

MODELING, ANALYSIS, IDENTIFICATION AND CONTROL OF MULTIVARIABLE SYSTEMS

Antonio Sala
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Documents and audio/video compilation

DOCUMENT VERSION:

Date and time: 05/08/2025 [13:30]
Number of learning objects: 165
Total video time: 42:08:16

[Own (165): 42:08:16; from third parties (0): 00:00:00]

Website with last versión: [HTML] [PDF]

©2025, Antonio Sala Piqueras, DISA-UPV.

Subscribe to companion **YouTube** channels: [**@asalacontrol, Español**] [**@ASalaControlEN, English**]

Chapter I

Presentation

This is a compilation of my materials in English and associated videos. You have many more in Spanish but, well, as there are many more choices in English (competition is fierce), I started recording videos in English quite late compared to Spanish. Sorry for the lack of content and disconnection between videos on the same topic, compared to the Spanish version.

Antonio Sala

Departamento de Ingeniería de Sistemas y Automática
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Note on originality

These materials deal with content which is already “settled” in the scientific community, many of them already “classic”, developed twenty or more years ago. Most of the ideas and algorithms are thus well known to those skilled in these topics, and thus no “research” originality is claimed thereon.

The only “originality” is in the “didactic” arena: the selection criteria and prioritization of the concepts, the organization of the materials and the way of presenting them in the transparencies and examples provided here. Therefore, this document or those linked to it should not be cited in a “research” context as an authoritative source for engineering recommendations or mathematical results/demonstrations, as they are not original/seminal. Expository considerations, thinking of the target audience (undergraduate/master students), intentionally make simplifications, variations, or assertions not entirely justified.

To avoid distracting the reader’s attention, the profusion of citations that is common in research texts has been avoided. However, it is explicitly stated here that there are abundant monographs that develop (in many cases, with much more depth and formalism) practically all of the concepts discussed here. Once familiar with the basic ideas, the student who wants more detail of the ideas in this text should consult his professor or TFM/Thesis supervisor, so that he can be referred to the corresponding authoritative references or seminal contributions.

Usage Rights

Regarding material under my authorship, all rights of use, copy, and intellectual property of the materials in this document and the audiovisual materials/PDF/linked code from it are reserved, both with respect to their *verbatim* usage or as a basis for derivative works. Nevertheless, use for *personal study*, by students and professors of *public* universities in official degrees, and the recommendation by their professors of this document is excepted from this reservation and expressly authorized. In another context (paid courses, academies, publications, commercial websites, ...), do contact me if you intend to get *money* or *merit* from these or derivative works and let me in.

Contents

I	Presentation	1
1	Modelling and simulation	5
1.1	Basic Math review for newcomers	5
1.2	Introduction to modelling	7
1.3	Linearization	10
1.4	Simulation	10
1.4.1	Gravitational interaction of several bodies	12
1.4.2	Interactive simulation	12
1.5	Solving Ordinary Differential Equations	13
1.5.1	Matlab dsolve	13
1.5.2	Intuition on linear systems response	14
1.5.3	Laplace transform techniques for time response	15
1.6	Case studies	17
1.6.1	HVAC operating point thought experiment	17
1.6.2	DC motor plus gearbox	18
1.6.3	Tubular heater case study	18
1.6.4	Spring-mass case study	23
1.7	Phugoid mode of aircraft (longitudinal flight dynamics)	25
2	PID control	28
2.1	Intuitive trial-and-error approach	28
2.2	Double integrator (motion control) case study	30
3	Control structures	32
3.1	Manipulated and Controlled Variable selection, 2x3 process	32
3.2	Control with excess actuators (manipulated variables)	35
3.3	SVD decoupling	36
3.4	Case Study: mixer/heater process	37
3.5	SUMMARY: the BIG picture in day-to-day control in industry	39
4	Other isolated things for the moment being.	40
4.1	Linear matrix inequalities (LMI), Semidefinite programming (SDP)	40
4.1.1	LMI problems with ellipses	40
4.1.2	Geometry of generic LMI/SDP-representable sets	43
5	Robust Control	45
6	Manipulability and Force ellipsoids in Robotics	51
6.1	Manipulability ellipsoid: speed locus	51
6.2	Force ellipsoid	52

7	Statistics and data mining	54
7.1	Introduction to probability and recursive Bayes filter: ‘roaring tiger’ case study	54
7.2	Other statistics concepts	57
7.3	Stochastic processes (Gaussian processes)	62
7.3.1	Multi-output GP	65
7.3.2	Derivatives of a Gaussian process	67
7.3.3	The Matern kernel, intuition and limit cases	68
7.3.4	Stochastic processes in state-space form	69
7.4	Bayesian Optimization	70
7.4.1	Introduction and methodology outline	70
7.4.2	In-depth detail and examples	72

Chapter 1


Modelling and simulation

1.1 Basic Math review for newcomers

[1: derivsmLEN] [Jacobians, Partial and Total derivatives: Matlab example](#) ***  15:17
Materiales: [[CÓD.:](#) DerivadasMLEN.mlx] [[PDF](#)] [YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video presents Matlab examples of using `diff` and `jacobian`. In particular, illustrates how chain rule works (product of Jacobian matrices), and the particular case of total versus partial derivatives in expressions $q(y(t), t)$, i.e., $\frac{dq}{dt} = \frac{\partial q}{\partial y} \frac{dy}{dt} + \frac{\partial q}{\partial t}$.


[2: maprEN] [Projection Matrices: quick introduction](#) ***  15:13
Materiales: [matrproyEN.pdf] [YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video outlines the concepts of orthogonal and oblique projections, with a graphical 2D/3D example and later on generalising the ideas to an arbitrary vector space (finite-dimensional), where a linear transformation can be represented by a matrix P .

Such matrix P is a projection matrix if $P^2 = P$. Projection is orthogonal if $P^T(I - P) = 0$ and that is equivalent to P being a symmetric projection matrix. Also, it is shown that projection matrix eigenvalues can be either zero or one. A couple of examples illustrate the concepts (particularly, the pseudo-inverse one, cornerstone of least-squares techniques). Oblique projection matrix to column space of A in direction B is also presented (without proof).

Ellipsoids

[3: ellip1EN] [Ellipsoids, positive definite matrices \(1\): basic definitions and motivation](#) **  16:29
Materiales: [EllipsIntroEN.zip] [YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video presents motivation and basic definitions of: a sphere centered at the origin, generated by the quadratic form $x^T x \leq 1$, an ellipsoid aligned with coordinate axes $x^T D x \leq 1$ (D diagonal) and, last, generic ellipsoids $x^T P x \leq 1$ (P symmetric positive definite), introducing rotation matrices (orthogonal matrices) and diagonalization to explain

the reasons leading to that last definition (the idea is discussed in more depth in the video [4]).


Of course, 2D ellipses are a particular case of generic N-dimensional ellipsoids.

It presents Matlab examples with `fimplicit`, and motivates its use in least squares problems (geometry of spheres), weighted least squares (geometry of ellipses) and other control, robotics, statistics, material science, etc. contexts.

The video [4], following this one, discusses in detail the interpretation of diagonalization, the ellipsoids not centered at the origin, and three equivalent ways to represent a given ellipsoid.

[4: ellip2EN] Ellipsoids (2): geometric interpretation of diagonalization, alternate representations of ellipsoids

Materiales: [EllipsIntroEN.zip]

**  14:59

[YouTube ▶]

* ENLACE A SPANISH VERSION


This video is a continuation of [3]. Here, we insist on the meaning of the diagonalization of a defined positive matrix, to determine semiaxes and their directions (recovering the diagonal scalings and rotations that generate it).

The second part of the video discusses alternative ways of representing ellipsoids: as a quadratic form $x^T P x \leq 1$, as $x^T Q^{-1} x \leq 1$, or as the image of the unit sphere by a matrix L (linear map). It also presents the ellipsoid not centered at the origin.

The video [5], a continuation of this one, presents 3D examples and degenerate cases of non-invertible P or Q .

[5: ellip3EN] Ellipsoids (3): 3D examples, degenerate cases, hyperboloids

Materiales: [EllipsIntroEN.zip]

**  08:28

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video presents examples of 3D ellipsoids, in order to complete the examples in the videos [3] and [4].

It also presents degenerate cases when P or Q are not invertible matrices in quadratic forms $x^T P x$ or $x^T Q^{-1} x$ appearing in ellipsoid expressions. In this case (if the non-zero eigenvalues are positive) cylinders or ellipsoids of lesser dimension are the result.

The last part of the video discusses the case where P has negative eigenvalues, which generate hyperboloids.

[6: ellip4EN] Ellipsoids (4): equivalence of representations

Materiales: [ElipsFullEN.zip]

***  14:25

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video presents a discussion on the equivalence of representations of ellipsoids: (1) direct $x^T P x \leq 1$, (2) inverse $x^T Q^{-1} x \leq 1$ and (3) as a linear transformation of a sphere $x = Lu$, $u^T u \leq 1$.

This video could be considered optional if you're just getting started with all of this, as it discusses refinements that might not be necessary for a first approach to these topics.

The topics discussed here complete the ideas presented in the video [4], to be watched prior to this one. Specifically, we discuss how to transform representation (3) to representation (2) when L is not invertible, either because it has excess columns or excess rows.

Then, the final part of the video discusses the transformation from representation (2) to (3) by factoring $Q = LL^T$ either by Choleski's method (L triangular) or by diagonalization ($L = V\sqrt{D}V^T$ symmetric from the diagonalization of $Q = VDV^T$). This L is not unique because we have one “rotation” degree of freedom as sphere $u^T u \leq 1$ renders the same sphere when rotated, so many L representing the same ellipsoid exist (square root of a matrix is not unique).

[7: ellip5EN] Ellipsoids (5): inclusion, inscribed/circumscribed sphere
Materiales: [ElipsFull.zip]


****  18:03
 [YouTube ▶]

*ENLACE A SPANISH VERSION

This video develops the conditions for an ellipsoid to be included into another. Basically, when written in direct form, $x^T P_1 x \leq 1$ is included in $x^T P_2 x \leq 1$ if and only if $P_1 - P_2 \succeq 0$. Conversely, in inverted form, $x^T Q_1^{-1} x \leq 1$ is included in $x^T Q_2^{-1} x \leq 1$ if and only if $Q_2 - Q_1 \succeq 0$.

The video dedicates two thirds of its duration to the proof of the above conditions. As a particular case, if one of the two ellipsoids is a sphere of radius ρ , that is, $x^T \rho^{-2} x \leq 1$, the application of the previous conditions results in conditions on the eigenvalues of P or Q that give the minimum radius and maximum radius of the ellipsoid (radius of the inscribed and circumscribed spheres, respectively).

[8: ellip6EN] Ellipsoids (6): cuts/transformations/projections
Materiales: [ElipsFull.zip]

***  09:51
 [YouTube ▶]

*ENLACE A SPANISH VERSION

This video, part of a series exploring the properties of ellipsoids (videos [3], [4], and so on), discusses cutting an ellipsoid with a plane or subspace, linear transformation of an ellipsoid, and, as particular cases, the cuts with the coordinate planes, and the projection on them.

[9: ellip7EN] Ellipsoids (7): relationship with singular value decomposition (SVD)


****  09:41
Materiales: [ElipsFull.zip] [YouTube ▶]

This video discusses the relationship between the singular value decomposition $L = USV^T$ of a matrix L with the geometry of the linear transformation with said matrix L of a unit sphere $x = L\nu$, $\|\nu\|_2 \leq 1$.

It is shown that the directions of the semiaxes are the columns of U , that the lengths of the semiaxes are the singular values (diagonal of S), and that the columns of V (input directions) are the points of the unit sphere that multiplied by L become the endpoints of the semiaxes of the ellipsoid.

1.2 Introduction to modelling

[10: modelsintroEN] Mathematical and computational (digital twin) models of physical systems: motivation, practical use
 15:57

** 
 [YouTube ▶]

*ENLACE A SPANISH VERSION

This introductory video presents the concepts of models of physical systems in continuous time and their main applications, as a motivation for further detailed study of them (in other materials).

Apart from reduced “physical” models (prototypes), engineering uses mathematical models in the form of differential equations, and computational models (sometimes named as digital twin, in today’s marketing lexicon ‘full of hype wording’ for a 70 year old concept) are also used as tools to get things to work as intended, prescribed in some technical specifications in an engineering project.


Specifically, the models can be used in the project phase to design the system and perform computer simulations of its behavior before its construction.

Once the system has been built, a computational model can be executed to check the effect of certain maneuvers, before performing them on the real physical system at the cost of materials, energy or risk to people or goods. It can also be used to simulate the signal evolution in time for signals for which actual physical sensors are not available (this application is called an *observer* in control theory).

Lastly, it can be used to adjust constant parameter values so that the simulation looks as close to the experiment as possible, and if those values are abnormal, then a fault has been detected.

Finally, the video describes the basic ideas of control system design based on these models.

Obviously, the details of all the above concepts and use cases constitute the core of bachelor’s and master’s courses in the areas of Systems Engineering, Control, Robotics, etc., which this video gives a glimpse of.

[11: pinionmEN]	Pinion-rack dynamical model: Newton equations, equivalent mass/moment of inertia, reaction force Materiales: [RackPinion.pdf]	**  15:31 [YouTube ▶]
-----------------	---	---

* ENLACE A SPANISH VERSION


This video discusses the dynamic modeling of a rack-and-pinion mechanism with one mechanical degree of freedom.

Newton’s equations (force balances, including reaction force) and the kinematic constraint linking linear and angular displacements (and, subsequently, velocities and accelerations) are presented.

With that complete model, two equivalent simplified models are then presented, either seen from the gear side (equivalent inertia) or seen from the rack (equivalent mass) side. Indeed, since the model only has 1 degree of freedom, everything can be reduced to an equation of order 2 of equivalent mass or inertia times linear or angular acceleration, respectively.

The final part of the video calculates the reaction force between the rack and pinion, both in equilibrium (zero acceleration) and out of it.

An equivalent approach based on Euler-Lagrange with constraints formalism appears in video [12].

[12: pinionELEN] 17:13	Pinion-rack dynamical model: Euler-Lagrange equations with kinematic constraints Materiales: [RackPinionEL.pdf]	****  [YouTube ▶]
---------------------------	---	---

This video discusses the Euler-Lagrange modeling of a 1 degree of freedom rack-and-pinion mechanism.

The mechanism is identical to the one seen in the video [11], where Newton's equations were used, with force and torque balances where reaction forces intervened in the contact. The Euler-Lagrange equations do not involve reaction forces, but rather kinetic and potential energies (the latter is zero in this specific simple example).

The video uses the Euler-Lagrange formalism in two ways:

- Expressing the Lagrangian $L = T - V$ as a function of a single coordinate (1 degree of freedom model), obtaining a model with (inertia/equivalent mass) times acceleration equal to a certain balance of torques or generalized forces.
- Expressing the Lagrangian in terms of two generalized coordinates (2 degree of freedom model) and introducing the mechanical link between both, $f(q) = q_1 \rho - q_2 = 0$, through a Lagrange multiplier. Due to the way it is written, the multiplier ends up being equal to the reaction force, but that need not happen in a more complex problem setup, in general.

As a result of both developments, exactly the same models are obtained as these in the video referred to above (which used Newton's mechanics), obviously, given that they are equivalent.

[13: tiovELEN] 2DoF dynamics of a carousel (rotational pendulum):
Euler-Lagrange equations

Materiales: [[CÓD.:](#) tiovivo2GLEulerEN.mlx] [[PDF](#)]

**** 19:50

[YouTube ▶]

This video discusses how to model a merry-go-round (well, really with just one element hanging the mechanism is often called a “rotational pendulum”) using the Euler-Lagrange formalism.

The first ten minutes review the kinematics (defining the coordinates that describe the motion plus deriving expressions of positions, speeds and accelerations of the elements of the system based on said coordinates). The `jacobian` command is widely used to apply the “chain rule”. If the manipulations are unfamiliar to you, perhaps you should preview the video [1].

Next, the Euler Lagrange equations are expressed with the Symbolic toolbox and the meaning of the different terms is analyzed, grouping them in the usual way $M(q)\ddot{q} = \tau + C(q, \dot{q})\dot{q} + G(q)$, i.e., with mass matrix and Coriolis terms.

Matlab code for simulations will be discussed in next video [17], and further digression on particular cases of interest will be discussed in video [14].

[14: tiovEL2EN] 2DoF dynamics of a carousel via Euler-Lagrange equations:
particular cases

Materiales: [[CÓD.:](#) tiovivo2GLEulerEN.mlx] [[PDF](#)]

*** 13:15

[YouTube ▶]

This video presents particular cases of the equations of motion of a merry-go-round carousel (with a single passenger) with two degrees of freedom, which were developed in the video [??].

In particular, the video discusses three situations:

- Zero Suspended Length
- Spherical pendulum (without top platform, load hanging from the center)
- Analysis of lateral dynamics only (this is not actually a particular case, but simply studying one of the two equations of motion in isolation).

Motivated by the latter case, the final part of the video discusses the idea of the *computed torque* necessary for the top platform to follow a certain prescribed trajectory, used in dynamic control of robotic mechanisms.

1.3 Linearization

[15: linmiso1EN] Linearization of function of 2 variables: Matlab example (symbolic toolbox), Taylor series revision

Materiales: [CÓD.: LinearizeMISOexample.mlx] [PDF]

** 16:57

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video develops a Matlab example of the linearization of a function of 2 variables $f(x_1, x_2) = x_1^2/2 + x_2^2 + \cos(1.1x_1 + e^{-x_2}) + 0.25$ at an (arbitrarily chosen) operating point.

You can consult material dedicated to the theory in the video [??], with a “geometric” (tangent subspace) approach, that you may wish to watch alongside this video, to complement the Taylor series approach that we discuss here.

The function is first defined and represented; later on, a theoretical review of the multi-variable Taylor series and the Jacobian and Hessian definitions is made, and it is applied to the actual example under study, to linearize it using the jacobian command.

The final part of the video superimposes the (3D) graphics of the original function and its linearized approximation, verifying that it is indeed the tangent plane in absolute coordinates.

The optional video [16] will evaluate the accuracy of the approximation at a specific numeric point away from the tangency point, and also plot the approximation error (with a contour plot), discussing the relationship with the Hessian.

[16: linmiso2EN] Linearization of function of 2 variables, Matlab: linearization error analysis, Hessian eigenvalues

Materiales: [CÓD.: LinearizeMISOexample.mlx] [PDF]

*** 09:04

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video is a continuation of the video [15]. Here, the accuracy of the linearization is evaluated numerically at a certain point (incremented from the operating point), and then a map (contour) of the approximation error is drawn.

Obviously, near the origin, the difference between the original function and the linearized approximation is dominated by the degree 2 terms of the Taylor expansion, so the properties of the Hessian (matrix of second derivatives) allow us to estimate the linearization error and its properties near the operating point: all eigenvalues of the same sign indicate “ellipsoidal” contours of the error, eigenvalues of different sign would indicate “hyperbolic” type contours (not the case in the example here).

1.4 Simulation

[17: tiouvELsimEN] 2DoF dynamics of a carousel: simulation and animation

12:43

Materiales: [CÓD.: tiouv2glanimv2.m] [PDF]

*** 09:04


[YouTube ▶]

This video animates the equations of motion of a merry-go-round (with a single hanging element) that were derived in the video [13] using the Euler-Lagrange methodology.

To do this, the results of the aforementioned video that used the Symbolic Toolbox are transformed/compiled into “numeric” code (i.e., non-symbolic ordinary Matlab functions), using matlabFunction, and everything is prepared in the required format to execute ode45. Different plot and plot3 are used to draw each animation frame.

[18: bcode45EN]

Simulation with ode45 of a closed-loop system (Matlab):
inverted pendulum with PD controller

***  18:14

Materiales: [[CÓD.](#): SimulaBucleCerradoPenduloODE45English.mlx] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video presents code to compute the state derivatives in a closed-loop block diagram, without the use of Simulink's GUI, just with ordinary Matlab code. The closed-loop state is actually the vertical stacking of process and controller states. First, we obtain state-space representations of both the controlled process (a pendulum in its upright unstable equilibrium) and the controller (a linear PD with noise filter).


Last, the above open-loop state and output equations are used in a function which computes the time derivative of both process and controller states.

With `ode45`, the resulting closed-loop system is simulated and compared with the linear simulation carried out with Control Systems Toolbox (`feedback` and `initial` commands). Simulations are basically identical if initial conditions are close to the operating point, as expected, and both linear and nonlinear simulations are stable in closed loop. If initial conditions are far away from the operating point, nonlinear simulation shows that there is not enough torque to lift the pendulum, so stability is only "local" (of course, linearised simulation cannot capture such a behaviour).

[19: ode45vs15sEN]

Numerical integration: comparison ode45 versus ode15s in stiff/
non-stiff ODEs (Matlab example)

10:46

** 

Materiales: [[CÓD.](#): Stiffoder45vs15sTSTEN.mlx] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)


The present video discusses numerical integration of ordinary differential equations (in first-order state-space representation), comparing performance of `ode45` versus `ode15s` in two different settings:

- In cases where all time constants are similar (non-stiff ODE); in such cases, `ode45` performs faster than `ode15s`
- In *stiff* ODEs, where there are very different time constants; in these latter cases, `ode15s` is the solver which performs significantly faster.

In the video [45] a tubular heater with transport delay is simulated; in that case, `ode15s` also rendered faster than `ode45` but, as shown in this material, each case needs to be individually studied in search for the best compromise between simulation accuracy and memory/time footprint.

[20: aproxretEN]

Transport Delay Approximation: simulation and comparison of
options (PDE, Padé, finite element)

***  17:38

Materiales: [[CÓD.](#): aproxrettemaPADE2ENG.mlx] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video discusses what transport delay is, with transfer function $\exp(-ds)$, and proposes three approximations of it to a finite order N :

- Pade approximation of order N
- Approximation of N finite elements of first order (each of them approximated with Pade of order 1)
- N finite element approximation with only poles.

The initial part reviews the transport partial differential equation (PDE) and its solution by Laplace (just a fast sketch, not the main objective; development is similar to the video [40]). You can skip it and fast forward to minute 3 if you are not interested in the transport EDP, don't worry.

The video compares the temporal (three cases: step, ramp and 1st order filtered inputs) and frequency responses with different values of N and draws conclusions about the validity conditions of the approximations.

1.4.1 Gravitational interaction of several bodies

[21: nbdsimcEN] N-body simulation of gravitational interaction: Matlab code/animation ode113

Materiales: [[CÓD.:](#) NbodySim.m] [[PDF](#)]

*** 16:42

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses the Matlab simulation code for the gravitational interaction of N bodies (just 2D planar motion, for simplicity). The actual ODE simulator is `ode113`, and the state equation is that the derivative of the position is the velocity and the derivative of the velocity is the acceleration of gravity (Newton formula). Each body has position 'x' and position 'y' as well as velocities, so the total order of the system of differential equations is $4N$. All the loop code that calculates the forces for all pairs of bodies and the resulting acceleration for each body is explained.

The final part of the video briefly discusses the code for animating object motions, simply with `plot`.

Actual examples of simulations with several number of bodies, Kepler laws and escape velocity are illustrated in video [22].

[22: nbdsim1EN] N-body simulation of gravitational interaction examples: ellipses, chaos, escape velocity

Materiales: [[CÓD.:](#) NbodySim.m] [[PDF](#)]

** 16:15

[YouTube ▶]

This video presents examples simulating the motion of several bodies under gravitational interaction. The video uses the code detailed in video [21], briefly summarised in the first two minutes. Then, examples with 3, 4, and 5 bodies appear and emphasis is placed on the simulation of problems with two bodies, to illustrate elliptical orbits, barycenter motion (conservation of momentum), and escape velocity of a light body from the attraction of a (much) heavier body, via mechanical energy conservation arguments.

1.4.2 Interactive simulation

[23: interactankEN] Interactive simulation of a liquid tank: Matlab class, source code detail

Materiales: [[CÓD.:](#) TankAnimInteractive.zip] [[PDF](#)]

**** 14:56

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video details the Matlab code for interactive animation (responding to keyboard input) of filling or emptying a liquid storage tank. This animation was used in previously published introductory motivational videos for systems engineering ([??] and [10] videos).

Here we discuss line by line the class (`classdef`) that handles the animation data and has the following methods:

- *constructor*, to initialize the data;
- “frame” drawing for animation, 20 times per second;
- simulation of the differential equation $\frac{dh}{dt} = \frac{1}{S}(u - \kappa\sqrt{h})$, approximating by Euler’s method. If the sampling period (frame to frame, 1/20 s) were larger, another numerical integration method might be needed, such as [ode45](#). This method is the most conceptually important from a systems theory point of view (numerical integration).
- Response to ‘key press’ event to perform different actions depending on the key that the user had actually pressed. The actions are raising/lowering inlet flow or stopping the simulation.

Last, the code of a script that is in charge of creating the animation, simulating for a certain time and presenting input and output plots once the simulation is finished is, too, detailed.


1.5 Solving Ordinary Differential Equations

1.5.1 Matlab dsolve

[24: masmusym1EN] Solving ordinary differential equations (ODE) with Matlab (dsolve): mass-spring free response

10:16

Materials: [[CÓD.:](#) testEDOsymEN.mlx] [[PDF](#)]

** 

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video presents the differential equation of a mass-spring system with friction $M\frac{d^2y}{dt^2} = -ky - b\frac{dy}{dt}$. Then, substituting the symbolic constant parameters for their numerical values obtains the general solution (with two constants of integration) using [dsolve](#). This solution is analyzed, but, in order to better understand the ODE it is again solved, also with the same command, giving prescribed initial position and velocity values. The solution is graphically represented and animated, to understand the type of movement obtained (damped oscillations).


The `dsolve` command allows you to obtain an explicit expression of the solution formula, instead of just a set of points like the [ode45](#) numerical simulations.

Additional simulations by varying parameters are covered in the video [25], and the animation Matlab code is detailed in the video [26].

[25: masmusym2EN] Solving ordinary differential equations (ODE) with Matlab (dsolve): mass-spring state-space form

10:08

Materials: [[CÓD.:](#) testEDOsymEN.mlx] [[PDF](#)]

** 


[YouTube ▶]

*ENLACE A SPANISH VERSION

This video obtains the free time response to non-equilibrium initial conditions of a mass-spring system, addressing exactly the same second-order problem as the video [24] $M\ddot{y} = -ky - b\dot{y}$. In this video, however, it is addressed by posing the equations in normalized internal representation $\frac{dx}{dt} = Ax$, with $A = \begin{bmatrix} 0 & 1 \\ -k/M & -b/M \end{bmatrix}$. The [dsolve](#) command does not have any problem with this representation and obviously obtains identical results to that of the referred video. The animation code is detailed in the video [26].

[26: masmuAnimEN]
12:35

Animating the time response of a mass-spring system: Matlab
 animation code and effect of damping

*** 

Materiales: [[CÓD.:](#) `simulspringdamperEN.m`] [[PDF](#)]

[\[YouTube ▶\]](#)

* [ENLACE A SPANISH VERSION](#)

This video presents animation results of the free response of a mass-spring system (with frictional damping) obtained in the videos [24] and [25] by means of `dsolve`. The basic drawing commands are `patch`, `xline`, `yline`, `plot`, inside a loop that generates each “frame” (not in real time, this is not a videogame).


Actually, the detail of the code is merely “cosmetic” with drawing functions, it does not address any conceptually important issues of solving differential equations.

We also check what happens when decreasing and increasing the coefficient of friction.

An example of code, similar but more laborious, for animation of the vibrations of a system with three masses and four springs is described in the video [51].

[27: masmusymForzEN]
13:41

Solving ordinary differential equations (ODE) with Matlab
 (dsolve): mass-spring forced response

*** 

Materiales: [[CÓD.:](#) `testEDOsymForcedEN.mlx`] [[PDF](#)]

[\[YouTube ▶\]](#)

* [ENLACE A SPANISH VERSION](#)

This video presents the solution of the mass-spring-damper ODE $M\ddot{y} = -ky - b\dot{y} + F(t)$ when subject to an input force with constant and sinusoidal components. The ‘free’ unforced response was discussed in video [24], which you might wish to watch prior to the one here.


It is verified that terms “similar” to the input do appear: constants indicating the new equilibrium if the force has a constant component, and vibrations of the same frequency as the input (sinusoidal stationary regime); also, terms with components akin to those of the free response, with decay rate and fixed proper frequency of oscillations that do not depend on the characteristics of the input $F(t)$.

The result is plotted and animated, using the same code as explained in the video [26] (animation details omitted here, please watch said video).

1.5.2 Intuition on linear systems response

[28: linregla3EN]

Intuition on linear dynamics: open loop control by sequence of
 steps (1st-order plant)

*  13:13

Materiales: [[CÓD.:](#) `LinearIntuitiveENG.mlx`] [[PDF](#)]

[\[YouTube ▶\]](#)

* [ENLACE A SPANISH VERSION](#)

This video proposes using a “step response” test obtained experimentally (or in simulation, but the detail of which equations constitute the model is irrelevant as long as they are linear –and approximately of the first order, without “inertia”–) to do straightforward proportionalities aimed at solving the following problems:

1. Achieve a certain setpoint (reference) by means of a single step increment.
2. Getting to said certain preset setpoint in a given settling time, and then staying there employing TWO steps.


To do this, the appropriate proportionalities are stated, and the performance achieved with the resulting step input profiles is checked in simulation. Since the simulated process is a first-order linear system, everything works perfectly as calculated. If the process were of a higher order (with “inertia”) then there could be overshoot and unexpected transients. An example of this fact (with code practically identical to the one explained here) is discussed in the video [29].

This could be considered a special case of what is called *open-loop control*: calculating an input profile without using sensors. In this simple case, only linearity and superposition are used to obtain that input profile, without needing any transfer function or representation in state variables as other more advanced methodologies do.

None of the computations have needed any theoretical knowledge about linearization or linear differential equations, they have simply used the proportionalities that characterize the response of linear systems (incremental around an operating point).

[29: linregla3ord2EN]
10:12

Intuition on linear dynamics: open loop control sequence of steps, 2nd order plant doesn't work well

*** 

Materiales: [[CÓD.:](#) LinearIntuitiveOrder2ENG.mlx] [[PDF](#)]

[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video uses the same methodology as the video [28] to calculate a two-step input signal that will bring a process up to a certain desired value in a desired time, and then hold it there, in equilibrium. You should watch the referred video before this one.


The only idea needed in both videos is linearity/superposition/time invariance, the model (e.g. transfer function $G(s)$ or similar) is not used at any time to calculate the input profile.

The previous video we were commenting on used a first-order process, in which the technique worked perfectly. Here, it is applied to a “secret” process that is actually of order 2, so that “inertia” means that when the input that maintains the desired equilibrium point is applied in the second step, there is an overshoot. The overshoot decreases if the first step (which accelerated faster) is made shorter, but then the desired place is not reached exactly. Refining this, either with an additional step of “braking”, or in a closed loop (PID control), is obviously not part of the objectives of this video, which only tries to reinforce the concept of “linearity” and “time invariance” of a dynamic system.

1.5.3 Laplace transform techniques for time response

[30: RCRmodEN]

Modelling series RC circuit with leakage resistance: state space + Laplace domain (Symbolic toolbox)

**  08:52

Materiales: [[CÓD.:](#) PulseSineRCREN.mlx] [[PDF](#)]


[YouTube ►]

This video models a serial Resistor-Capacitor circuit whose capacitor has some leakage conductance, modelled as a second resistor in parallel with the capacitor ($R + [C \parallel R]$).

First-principle equations and Kirchoff's laws are used. Then, a normalised state-space representation is built and, last, its Laplace transform is used to obtain the transfer function and initial condition term in the Laplace domain. Computation of the time response of this circuit given some input waveforms will be addressed in future videos, see [32] and [33].

[31: sinpulLEN]

Laplace transform of a sinusoidal pulse (semiperiod)

**  11:56

Materiales: [[CÓD.:](#) PulseSineRCREN.mlx] [[PDF](#)]

[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video presents two techniques for calculating the Laplace transform of a sinusoidal pulse (a half-period of the sine function).

Apart from calculating it with the `laplace` command of Matlab's Symbolic Toolbox, it proposes calculating it (a) as the superposition of a sinusoid and the same sinusoid delayed one semi-period using the lag operator e^{-ds} , or (b) directly by making the integral that defines the Laplace transform, integrating by parts twice.

Obviously, all results are identical.

NOTE: although these types of questions are typical in some exams, it turns out that by applying "superposition", the Laplace transform of the input calculated here is often NOT necessary to calculate the time response, since, for example, we may calculate the response to a sinusoid (complete, from $t = 0$ to infinity, transformed $\omega/(s^2 + \omega^2)$) and apply the same displacement+superposition on the output. In fact, this is done in quite a few of the examples with "piecewise" inputs in other videos.

For instance, time response to this sinusoidal pulse of an electric circuit is computed in videos [32] and [33].

[32: `sinpulRCREN`] Time response of RCR circuit to single sinusoidal pulse (1: Laplace, superposition)

Materiales: [[CÓD.:](#) PulseSineRCREN.mlx] [[PDF](#)]

** 11:49

[YouTube ►]

*ENLACE A SPANISH VERSION

This video computes the time response when subject to a (single) sinusoidal pulse input of a two resistor plus capacitor circuit whose model was obtained in video [30], to obtain the transfer function and initial condition term in the Laplace domain.

The first minute quickly discuss the modeling phase. The sinusoidal pulse and its transform are discussed in the video [31], whose viewing beforehand is recommended. However, this transform is NOT going to be explicitly used here, because the same idea of delay and superposition that was used in that video will be used here with the output to a full sinusoid.

The final part of the video discusses the response to non-zero initial conditions and the same input pulse. Obviously, it is a matter of simply adding an exponential (inverse transform of initial conditions term, free response) to the previously calculated solution.

An alternative "piecewise" way of calculating the response for this same circuit is discussed in the video [33] without resorting to "delay" concepts used here; also, the response to a train of sinusoidal pulses that repeat indefinitely is discussed in the video [34].

[33: `sinpulRCR2EN`] Time response of RCR circuit to single sinusoidal pulse (2: piecewise)

Materiales: [[CÓD.:](#) PulseSineRCREN.mlx] [[PDF](#)]

*** 10:59

[YouTube ►]

*ENLACE A SPANISH VERSION

This video calculates the time response of a circuit with 2 resistors and 1 capacitor to a sinusoidal pulse (semi-period) and non-zero initial conditions. The same problem was addressed in the video [32], where the idea of "linearity" and "temporal invariance" was used to be able to express the output by superposition of two sinusoid responses, one of them delayed. The statement of the problem and the summary of the aforementioned video are outlined in the first 2 and a half minutes of this material.

Here, the same problem is solved in a "piecewise" way, dispensing with the concepts of *delay* and *superposition* from the video referred to above. Two main concepts are used:

- *causality*: if the pulse and the full sine coincide during the first 0.01 seconds, their outputs will also coincide in that interval (the output does not depend on future changes in the input waveform).
- The *concept of state*: everything that happened in the past is summarized in the state (capacitor charge) at the moment when the input is no longer sinusoidal.

These two ideas allow us to solve two differential equation problems, one for the interval $[0, 0.01]$ and another for $t \geq 0.01$, with the initial condition of the second interval equal to the “final” condition of the first .

Repetition of this idea at many intervals will be used to calculate the response to a train of sinusoidal pulses in the video [34].

[34: trenpulRCREN] Time response of RCR circuit to a sinusoidal pulse train
16:44

**** 

Materiales: [[CÓD.:](#) TrainPulsesSineRCREN.mlx] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video obtains the response to a sinusoidal pulse train of a circuit with 2 resistors and 1 capacitor. This video is part of the case study of a circuit with sinusoidal pulse input, constituted by the videos [31], [32], [33] and this one.

The development is based on the superposition of fragments of response calculated by sections, by means of the concepts of causality and state, in a way analogous to that used in the video [33], whose previous visualization is recommended in order to better understand what it is done here.

First, it is justified that the Laplace transform of a periodic signal is that of the signal in a period divided by $(1 - e^{-Ts})$, with geometric progression sum formulas.

However, this Laplace transform of the pulse train is not used in the development, because its inverse is difficult to handle. Alternatively, solving is carried out in 10 ms fragment, using the final condition of the previous one as the initial condition of the next fragment.

The final part of the video discusses the steady state (not sinusoidal, because the input is not sinusoidal: there will be harmonics), computing it by solving the equation of the response that equates $y(t)$ with $y(t+0.01)$, being the initial condition an unknown variable.

1.6 Case studies

1.6.1 HVAC operating point thought experiment

[35: hvacop1EN] Operating point optimization: an air-conditioning (idealised) case study

***  22:40


[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video presents the decisions to make to optimize the setpoint in an air-conditioning system (HVAC). In fact, the problem statement is simplistic, idealized because, on one hand, nobody would use multicriteria optimization software to decide the temperature to input in the air-conditioning remote; on the other hand, comfort depends on multiple factors apart from temperature (moisture, dressing, radiation, ...) so an exhaustive study of the optimal operation point in, say, an office or manufacturing plant with dozens of workers would require specialized knowledge I am not an expert of.

However, this simplified air-conditioning thought experiment allows us to intuitively understand, without any formulae, the concepts of:

- Multicriteria optimization (trade-off between conflicting requirements),
- Dominated solutions,
- Cost indices, equal-preference curves,
- The meaning of convexity of the cost function in this particular case, so convexity is advisable,
- Final optimal operating point as a trade-off between comfort and power consumption once the physics of the actual process is plugged in.

[36: hvacop2EN]
Optimal selection of operating point and controlled variables:
air-conditioning thought experiment
****  17:14

[\[YouTube ►\]](#)

* ENLACE A SPANISH VERSION

This video is a continuation of [35], where the decisions to take when optimising an operating point were illustrated in an idealized air-conditioning problem setup.

Here, apart from multicriteria-optimization issues discussed in the cited video, we now consider which process variables should be chosen as “controlled variables”, i.e., variables for which a constant setpoint is set so closed-loop control tries to keep such a setpoint despite external disturbances trying to move away such variables.

The basic idea is that a given process variable is a good candidate to be a “controlled” variable if its optimal value (according to given design criteria and tradeoffs) remains constant even if exogenous factors (disturbances) do change.

Three “thought experiments” are presented, considering possible changes of outside temperature and different “importance” of energy-saving issues in the comfort vs. power cost trade-off.

In a first case, temperature will be a good option as a controlled variable; in a second case, operation at constant power would be the recommended option and, in a third case, neither constant temperature nor constant power are optimal choices, so a “setpoint optimization” block would be recommended to generate temperature and power operating points as a function of outside temperature (assumed measurable, i.e., this optimizer would amount to something very similar to a *feedforward* term, generalising the concept and the realm of application of basic feedforward ideas to complex systems).

NOTE: no one will do this in his living room (at least no one will be actually aware he is doing this); this video is a very complex way of telling that “prioritising comfort amounts to controlling temperature”, “prioritising energy saving amounts to setting a given cooling power when comfort is plainly unacceptable”.

1.6.2 DC motor plus gearbox

In ellaboration.

1.6.3 Tubular heater case study

[37: termrdmapEN]
Tubular heater case study: ROADMAP
*  06:51

[\[YouTube ►\]](#)

This video presents the roadmap of a case study on the behaviour of a tubular heater, which will be discussed in a series of videos.

The videos will discuss from simplistic first-order models to partial difference equations, and intermediate options such as first-order time dynamics with non-uniform longitudinal temperature profiles or finite-element ones.

The first video of the case study with actual content is [38], and the links in its description will let you through the complete case study.

[38: term1eEN]
Modelling the dynamics of a heating tank (1st order, perfect mixing, incompressible flow)

*** 12:52

Materials: [CÓD.: HeatExch1elementModelEN.mlx] [PDF]

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video derives a first-order model for a heating tank (perfectly stirred so everything is assumed to mix instantly, yielding an homogeneous temperature in all of the tank's volume). If it had non-perfect mixing (or, say, if it were a long pipe) we would need Partial Differential Equations (PDE) as done in most heat exchanger theory; the most complex cases would need full computational fluid-dynamics code; for simplicity, this will not be the case here and we'll just consider perfect mixing, 1st order dynamics.

A heating resistor providing thermal power Q is present, in order to heat a flow F of incompressible liquid, density ρ , specific heat c .

The final equation is $V\rho c\dot{T} = F\rho c(T_{in} - T) + Q - \kappa T$, and the video justifies it in terms of a power balance. Formally, as there are no pressure/volume changes, no work is done and increments of “energy” and “enthalpy” coincide.

There are other alternative approaches to build a model for a tubular heat exchanger; we have, of course, the PDE one (video [39]) and first-order approximations with nonconstant longitudinal temperature profile (video [41]).

[39: termedpEN]
Partial Differential Equation (PDE) modelling of a one-dimensional tubular heater for liquid fluids

16:16

Materials: [HeatExchModelEDPEnglish.pdf]

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video pursues modelling a tubular heat exchanger with at heating resistor alongside it, as a connection of multiple (infinitely many of them) elements where each of the elements is a mini-heating-tank modelled as a first-order dynamical systems; modelling is briefly outlined in the first two minutes of this video, detailed explanations appear at video [38].

Once the inputs and outputs of each element are suitably named, the separation between elements will be set to “ dx ”. Conceptually, if one element's temperature is $T(x)$, that of the next element will be $T(x+dx)$. As we are analysing the time evolution of temperature, we actually pursue findin the equations governing the evolution of $T(x,t)$ at different positions and time instants.

Taking limits when $dx \rightarrow 0$, the following PDE (partial differential equation) is obtained:

$$\frac{\partial T}{\partial t} = -\frac{1}{S}F\frac{\partial T}{\partial x} - \frac{\bar{\kappa}}{S\rho c}T + \frac{\bar{Q}}{S\rho c}$$

The final part of the video discusses two frequent particular cases, well studied:

- The first one is transport delay (no heating, perfect insulation), i.e., the transport PDE: $\frac{\partial T}{\partial t} = -v\frac{\partial T}{\partial x}$ being $v = F/S$ the linear speed of the fluid.

- The second particular case is the steady-state heat exchanger (enforcing equilibrium condition $\frac{\partial T}{\partial t} = 0$), considered for simplicity under no heating, i.e., $\bar{Q} = 0$. In that case, we obtain an ordinary differential equation (ODE) in the longitudinal position variable $\frac{\partial T_{eq}}{\partial x} = -\frac{\bar{\kappa}}{F\rho C_e} \cdot T_{eq}$ which obviously has an exponential solution which is widely used and explained in depth in heat-exchanger textbooks.

Numerical simulation of PDE requires, in a general nonlinear case, the approximation (discretization) to a finite number of non-infinitesimal elements, as considered in the starting phases of the modelling procedure in this material. Nevertheless, in this particular case, for constant flow, there is a Laplace-transform solution (with delay) that can be simulated with the control system toolbox in Matlab; details on how to solve the PDE are in video [40], the step response of such solution is detailed in video [42].

[40: termedpsolEN]
19:57

Partial Differential Equations tubular heat exchanger: PDE solution via Laplace transform (transfer function)

Materials: [HeatExchModelEDPsolFlujoCteENGLISH.pdf]

[YouTube ►]

*ENLACE A SPANISH VERSION

This video details how to obtain the solution to the partial differential equations describing the dynamics of a tubular heat exchanger with a resistor inside. Solution is obtained via Laplace transform techniques on the "temporal" side. Further detail on how to obtain the PDE model can be found in the video [39]. Given the tubular geometry, PDE is one-dimensional in space, for simplicity. The PDE is linear if flow traversing the heat exchanger is constant, so it will be assumed as such.

Really, we don't obtain the actual "solution" understood as $T(x, t)$, but a transfer function representation of the outlet temperature as a function of the heating power increments or those from variations of inlet temperature. Actual solutions and simulations will be the objective of future materials.

The resulting expression contains terms arising from the transport delay phenomena that arise in the underlying physics. It has the form:

$$T_{out}(s) = \frac{b \cdot (1 - e^{-\phi s} e^{-\phi a})}{s + a} Q(s) + e^{-\phi s} e^{-\phi a} T_{in}(s)$$

The step response of these transfer functions is simulated in the video [42], continuation of this one.

[41: term1expEN]
22:42

Transient modelling of tubular heat exchanger: 1st order, exponential temperature profile assumption

Materials: [EXPprofileHeatExch1elementModeladoENGLISH.pdf]

[YouTube ►]

*ENLACE A SPANISH VERSION

This video models a tubular heat exchanger element with a heating resistor alongside it.

Intentionally, we seek a first-order dynamics as a simplification compared to dealing with the "true" partial differential equations arising in a more detailed modelling (see video [39] for PDE modelling and [40] for the Laplace transform solution).

A first approach to a simplified model was addressed in video [38], under the assumptions of short and perfectly-stirred fluid in the heat exchanger.

Nevertheless, given that the heat exchanger PDE have a well-known stationary solution with exponential terms in the spatial coordinate, this video proposes using such exponential profile when computing the "mean" temperature used in the energy balance, while keeping first order temporal dynamics.

With this exponential spatial profile the match of our simplified setup with the steady-state outlet temperature predictions of the PDE will be better than with other simpler

approximations such as the “uniform” interior temperature which was discussed in the video [38]. In fact, the said result will be the “very long” limit case of the present model; also, the “very short” model will end up being that obtained under assumption of a linear (in space) temperature profile (this model is briefly reviewed here, too).

The relationship of the approach to the firmly established LMTD (Logarithmic mean temperature difference) formulae is briefly outlined; we are a bit more general as we incorporate time transients and the possibility of internal heating.

As a result, transport delay physical phenomena appear as non-minimum-phase components in the transfer function associated to increments of inlet temperature.

Of course, if the accuracy of a 1st-order model were not deemed enough for a particular application, then the actual PDE or finite-element approximations of it would need to be used.

[42: termedpstepEN]
12:47

Step response (inlet temp, heating power) of the PDE solution of a tubular heater transient dynamics

Materials: [[CÓD.:](#) HeatExchModelEDPsimulaENGLISH.mlx] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video briefly reviews the solution to the partial differential equations governing a tubular heat exchanger (details in video [39] –modelling–, [40] –Laplace Transform solution in transfer function form–). Then, it simulates its step response to increments of inlet temperature as well as increments of heating power. Simulations are carried out in two scenarios: first, one with longitudinal cooling due to conduction to the outside environment, and a second scenario with perfect thermal insulation.

[43: term1evsedpEN]
17:56

Tubular heat exchanger: Comparison between exact EDP solution and 1st, 3rd order approximations

Materials: [[CÓD.:](#) HeatExchModelEDPsimulayOrdenRedENGLISH.mlx] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video compares the exact solution of the partial differential equation of a tubular heater (details of the PDE modelling in video [39]; solutions were obtained in transfer function form in the video [40]) with finite (1st and 3rd) order approximations of them.

Erratum: there is an error in the video at line 5 of code where a power unit conversion factor from W to kW is wrongly propagated. The MLX and PDF have been corrected, changing the heat transfer constant \bar{k} so that the numerical results of the video remain correct.

Once the typo has been clarified, let’s continue with the description.

Actually, an approximation of the exponential in the Laplace transfer functions via a certain parameter β , first order models are derived, which, luckily, are identical to those obtained based on physical insight with a non-uniform longitudinal temperature profile in video [41]. This is important, because models from first-principle insights can be simulated when input flow changes, whereas approximation in the Laplace domain cannot (transfer functions were obtained under a constant flow assumption).

A particular case $\beta = 1$ renders the perfectly-stirred first-order approximation in video [38]; $\beta = 0.5$ renders a linear temperature profile, see [41] for details.

Additionally, Pade approximations of the delay are compared to the above models (first and third order Pade approximations, via Matlab’s [pade](#)).


Comparisons are made in both steady state (DC gain) and transient response to steps (in inlet temperature and in resistor heating power).

A few of these low-order approximations may be cascade-connected to yield finite-element setups, discussed in other materials, such as the video [44].

[44: tubulFE1EN]

Tubular heater/exchanger: finite element modelling (1)

15:06



Materialies:

[

CÓD.:

HeatExchModelElemDiscreENGLISH.mlx


]

[

PDF

]

[YouTube



*ENLACE A SPANISH VERSION

This video addresses the finite element modeling of the tubular heat exchanger/heater discussed in the videos starting with [38] and continuing in the links/references in the video description.

The first seven minutes review the many models discussed in this case study, ranging from:

- Model as a partial differential equation, PDE, [39]
- Static (stationary) models derived from said PDE (very common in practice), setting time derivatives to zero
- First-order models based on energy balance (see video [41]) over a mean temperature.

This review may be of interest in itself, to understand/unify the concepts of all those previous videos.


Next, it is proposed to subdivide the exchanger into N equal pieces (finite elements). Thus, we discuss how to interconnect the elements, both in a block-diagram form and with the state and output equations of each of the elements (which will be assumed first order, as mentioned above), so that they form a model, of order N , that approximates the “exact” solution of the PDE (which would be the limit when making the elements *infinitesimal*). Obviously, forming the N state equations with very large N is done using a loop and the Symbolic Math Toolbox, and explaining the code that does this is the goal of the final part of the video.

Actual simulation of this finite-element models will be discussed in video [45], which will be the next in the case study.

[45: tubulFEsim1EN]

Tubular heater/exchanger, finite elements: numerical simulation (1)

13:59



Materialies:

[

CÓD.:

HeatExchModelElemDiscreENGLISH.mlx


]

[

PDF

]

[YouTube



*ENLACE A SPANISH VERSION

This video is a continuation of the video [44], where it was shown how to introduce a (symbolic) finite element model of a tubular heater in Matlab, the heater is the object of a case study throughout multiple videos.

In this video, apart from briefly summarizing the video referred to in the first three minutes, we detail how to translate the problem from symbolic to numerical, how to define the inputs and how to integrate it with a numerical integrator. The execution times of `ode15s` and `ode45` are compared, the former being advantageous in this case when the number of elements is large, see [19] for other comparisons of the two integrators (no one is better than the other in all cases).


The simulations seem generally satisfactory, although there are certain artifacts (high frequency oscillations during the delay time) when using the coefficient $\beta \approx 0.5$ in the model of each element, related to the Padé approximation of the delay in each of them.

The detail about these oscillations, and the influence of the number of elements and the value of β on the simulation result are addressed in the video [46], continuation of this one. It would also be interesting to watch the video [20] on delay approximation (outside the case study) because its conclusions apply here, and they are similar to those obtained in the video [46].

[46: tubulFEsim2EN]

12:25

Tubular heater/exchanger, finite elements: numerical simulation (2)



Materials: [[CÓD.: HeatExchModelElemDiscreENGLISH.mlx](#)] [[PDF](#)]

[\[YouTube ▶\]](#)

* [ENLACE A SPANISH VERSION](#)


This video, a continuation of [45], presents simulations with a different number of elements and with a different value of the interpolation coefficient β , different from the theoretically necessary one to accurately replicate the energy balance in a steady state (equilibrium). The changes of β allow to reduce the artifacts in the step response simulation... although if the inputs do not change suddenly, they may not be a problem: the discussion is parallel to the video [??] where the approximation is discussed of transport delay specifically.

This ends the case study on the tubular heater that began in the video [38]; a summary and final conclusions about the case study in general will be addressed in the video [47].

[47: tubulconcEN]

16:31

Tubular heater/exchanger case study: recap and guidelines on model complexity choice



Materials: [IntercConclusionsEnglish.pdf]

[\[YouTube ▶\]](#)

* [ENLACE A SPANISH VERSION](#)

This video is, in a way, the conclusion of the tubular heat exchanger/heater case study discussed in the videos starting with [38] and continuing in the links/references therein.

It recapitulates the many ways to give a model, from purely “static” (simplest) to nonlinear partial differential equations, through first-order models based on energy balance of the *mean* temperature.


Excessive model complexity results in computational cost issues, and possibly too complex controllers. Besides, that will never solve problems of unmodeled dynamics (conduction in the metallic piping, turbulence) or parametric errors in heat transfer coefficients, or bandwidth limitations of the instrumentation of a specific application. Therefore, depending on the ratio between the residence time of the fluid and the settling time that is sought in each specific application, the best engineering choice will be a certain “sensible” complexity of the model: more complexity isn’t always better in ‘practical engineering’.

1.6.4 Spring-mass case study

[48: moll3modEN]

First-principle modelling of a 3 mass, 4 spring mechanical system (state-space internal representation)

**



10:00

Materials: [Model3muellEnglish.pdf]

[\[YouTube ▶\]](#)


* [ENLACE A SPANISH VERSION](#)

This video models the state-space differential equations of a mechanical system, consisting on three masses connected to each other by means of springs, and also connected both to a fixed wall and to a mobile end, whose position will be the input to the system. Although initially the equations are proposed in “absolute” coordinates, including the natural length of the springs, we quickly switch to “incremental” coordinates (with respect to the equilibrium position).

Six first-order differential equations are obtained that constitute the normalized internal representation (state variables are three positions and three velocities). As it is linear, it could be expressed as $\dot{x} = Ax + Bu$, although, for brevity and convenience, that step will be carried out in following videos, where the model will be simulated and animations of it will be made.

[49: moll3mod2EN]

First-principle modelling of a 3 mass, 4 spring mechanical system: recap and normalised matrix form
Materiales: [3M4SModelSummaryEN.pdf]

**  14:00
 [YouTube ►]

*ENLACE A SPANISH VERSION

This video is a continuation/complement of the video [48], which details how to obtain each equation of motion of a system with 3 masses and 4 springs, and the signs of the force balances.


This additional video reviews the concept of a model in 'absolute' units (with natural lengths of springs) and its equilibrium point (zero derivatives). Then the model is presented in incremental units and its expression in normalized form $\dot{x} = Ax + Bu$, $y = Cx + Du$ calculating the matrices A and B ; regarding output equation, do note that C and D matrices depend on each application, here $C = I$, $D = 0$ are proposed as an example (full information output).

The simulation of the above mass-spring system (with a generic `ode45` solver and also a specialised linear system simulator `lsim`) is discussed in video [50], which continues this case study.

Eigenvalues of the A matrix are key in analysing the properties of the free response (zero input), as discussed in the video [52].

[50: moll3sim1EN]

First-principle model of a 3 mass, 4 spring mechanical system: ode45 vs lsim simulation
Materiales: [Cód.: simul3M4Sstandalone.m] [PDF]

**  12:54
 [YouTube ►]

*ENLACE A SPANISH VERSION

This video continues the case study of a 3-mass, 4-spring system whose first-principle modeling was covered in the videos [48] and [49] with force balances and Newton's equations.


Here, the equations $\dot{x} = Ax + Bu$ derived in the referred materials are entered in matrix form, simply by entering the constant matrices A and B by copying them from the theoretical modeling slides. The output equation $y = Cx + Du$ is implemented so that only the three positions are part of the vector y .

Once the model has been typed onto a Matlab script, a simulation is performed under certain initial conditions and a sinusoidal input profile (arbitrarily chosen). To compare the results and the computation time, the numerical integration is performed both with `ode45` (a generic simulator, which can also simulate nonlinear systems) and with `lsim` (a simulator specialized in linear systems). The second case is, at least in this example, several times faster to execute; both simulations are basically coincident, apart from numerical tolerances to the fifth or sixth decimal place.

Animation in order to better apprehend how this system moves and vibrates will be discussed in the forthcoming video [51]. The free response (zero input) properties are discussed in detail in the video [52].

[51: moll3aniEN]

3 mass, 4 spring mechanical system: animation of free and forced response examples
Materiales: [Cód.: simul3M4Sstandalone.m] [PDF]

**  12:58
 [YouTube ►]

*ENLACE A SPANISH VERSION

This video presents the code that animates (drawing moving rectangles) the simulation results of a 3-mass 4-spring system whose model was developed in video [49] and whose simulation details (comparing `ode45` and `lsim`) were discussed in video [50].

A simpler animation of the oscillations of a single mass is presented at video [26]; maybe you should view that one prior to this if you are not familiar with Matlab.

Here, we animate the vibration in both longitudinal and transverse form, to better understand how the system vibrates; examples of free response and forced response (three resonant vibration modes, whose resonant frequency computation is not in the scope of this video) are provided.

[52: moll3freeEN] 3 mass, 4 spring mechanical system: modal analysis of free response oscillations
*** 15:10

Materiales: [[CÓD.:](#) FreeModes3Mass4SpringCode.zip] [[PDF](#)]
[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video presents an analysis of the free response modes of a 3-mass, 4-spring system whose modeling was detailed in the videos [48] and [49].

The Matlab code to visualize an animation of the movement of said masses was detailed in the video [51]; this code will be used (with minor variations) also in this video to present the results, but its detail will not be explained since it was already done in the aforementioned video.

Basically, the modes of the free response are given by eigenvalues and eigenvectors of the matrix A in a representation $\frac{dx}{dt} = Ax + Bu$, where, obviously, the term Bu has no relevance in the free response studied here.

In this case, A has complex eigenvalues and eigenvectors. The real part of the eigenvalues (decay rate) allows approximating the duration of the transient (time until the oscillations disappear); it is the same in all modes, although it need not be in a general case. The imaginary part is the frequency of the oscillations in radians per second.

In this system, the free response has three oscillatory modes that last 65 seconds to disappear, of different frequencies and with the masses either in phase or in phase opposition.

Eigenvalues and eigenvectors are analyzed, and the real part of the eigenvectors is loaded as initial condition to simulate each mode, and to display an animation of it, so that the meaning of the mode is better understood. These modes are similar but not the same as "resonance" modes (forced response to sinusoidal input).

1.7 Phugoid mode of aircraft (longitudinal flight dynamics)

[53: intrid1EN] Dynamics of planar movement in intrinsic (Tangent, Normal) coordinates (2 DoF point mass)
** 16:40

Materiales: [FrenetCdGDynamicsENG.pdf]
[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video discusses how to obtain the equations in Frenet intrinsic coordinates (tangent, normal) of the movement of a point mass subjected to forces. If the direction of the velocity vector is $\vec{T} := (\cos \theta, \sin \theta)$, then $\vec{v} = \nu \vec{T}$, where ν is the velocity of the "speedometer" in, say, a car (i.e., ν is a scalar). Then $\frac{d\nu}{dt} = \frac{F_T}{m}$, and $\frac{d\theta}{dt} = \frac{F_{NL}}{m\nu}$ where F_T is the tangential force, and F_{NL} is the normal force (positive if it points to the left, counterclockwise, of the trajectory). The proof of these expressions and the discussion of their meaning are the objectives of the video.


The final part recalls that a rigid body has an extra degree of freedom (angular) and, therefore, does not have to be oriented with the center-of-mass path angle θ , using airplane or sports car maneuvers as an example. The equations with that additional degree of

freedom are not covered, for brevity, in this introductory video, only devoted to “point mass” dynamics: we assume “path frame” and “body frame” are identical.

These equations will be used, for instance, in video [54] to model an aircraft/glider phugoid flight mode.

[54: fugoid1EN] Longitudinal phugoid mode dynamics of an aircraft/glider
(simplified 2nd order ODE)

Materiales: [PhugoidIntroEnglish.pdf]

***  16:13

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video presents the modeling of the so-called “phugoid” flight mode of an aircraft. Under certain assumptions (planar motion, negligible rotational inertia, large radius of curvature of the CoG trajectory), the plane can be considered as a point mass subject to tangential and normal forces; thus, we may write the equations of motion in intrinsic coordinates, discussed in the video videorefintrid1EN.

Once these equations of motion are taken into account, it is simply a question of substituting in said equations the thrust of the engine and the aerodynamic resistance in the tangential direction, the lift in the normal direction, and of breaking down the weight into its tangential and normal components.


With these ideas, it is possible to write the so-called Zhukovski/Lanchester equations that approximate this mode of flight dynamics called “fugoid”. Of course, this is a crude first approach, and more complex models are routinely developed in most flight dynamics textbooks.

Note: engine thrust has been assumed to be a constant input but depending on the propulsion system, it may have airspeed-dependent terms (for instance in constant-power propeller engines); these issues (and many others) have not been considered here, for simplicity.

The study of fugoid dynamics continues in video [55] (simulation), and [56] (equilibrium points and linearization).

[55: fugsimEN] Aircraft phugoid mode (simplified equations): discussion on
simulation/animation examples

Materiales: [Cód.: animaaavion.m] [PDF]

***  12:38

[YouTube ▶]

* ENLACE A SPANISH VERSION


This video presents simulations and animations with different lift/drag coefficients and thrust values of the simplified fugoid dynamics model whose derivation was discussed in the video [54]. The detail of the Matlab code to run these examples is presented in the video [57].

The chosen examples are: gliding down (no thrust), level flight, climb, and unstable climb, that occurs if thrust rises above a certain value (see video [56] for details).

Use of these equations is made in, for instance, the derivation of the phugoid aircraft flight mode, see video [54].

[56: fugeqlinEN] Phugoid aircraft dynamics: equilibrium, linearization, stability
(simplified 2nd order equations)

Materiales: [Cód.: FugoidLinearizENG.zip] [PDF]

***  17:59


[YouTube ▶]

* ENLACE A SPANISH VERSION

This video discusses calculation of equilibrium points, their linearization and the local stability analysis, of the simplified phugoid dynamics models (2nd order) of an aircraft (also ‘fugoid mode’). The detail of obtaining the model based on principles of Physics is discussed in the video [54], and different simulations with `ode45` and animations are covered in the video [55]. Some more animations are, nevertheless, shown here to illustrate concepts.

In this video, we discuss the relationships between angle, airspeed, and thrust needed to achieve equilibrium, as well as equilibrium point stability. Emphasis is placed on gliding (glider $u = 0$) and horizontal level flight ($\theta = 0$) as particular cases.

Stability is analyzed by obtaining a normalized state variable representation $\dot{x} = Ax$ for constant thrust, and evaluating the real part of the eigenvalues of A , solutions of the characteristic equation $\det(sI - A) = 0$. Of course, as with any linearized system, its stability only proves “local” stability of the original non-linear system.

<p>[57: fugsimcodEN] 12:53</p>	<p>Aircraft phugoid glide (simplified equations): <code>ode45</code> simulation and animation code (Matlab)</p> <p>Materials: [CÓD.: animaavion.m] [PDF]</p>	<p>*** </p> <p>[YouTube ▶]</p>
------------------------------------	---	---

*ENLACE A SPANISH VERSION

This video presents the detail of the Matlab code necessary to make simulations and animations of the 2D phugoid flight mode (longitudinal dynamics) of an aircraft.

The first two minutes review the state equations to be simulated (further details in video [54]). A theoretical analysis of the equilibrium points and their linearisation is carried out in [56].

Then, we discuss how to write the model as Matlab code, how to call `ode45` correctly, and in the final part, we discuss how to represent the response as either:

1. a graph of positions, velocities, accelerations, etc. as a function of time; or
2. an animation of the moving plane.

This video merely presents the source code. Actual runs of various simulations with different aerodynamic parameters, initial speeds or thrust is covered in detail in the [55] video.

Chapter 2

PID control

In preparation, I'm recording/editing a subset of the videos I have in my Spanish channel.

2.1 Intuitive trial-and-error approach

[58: dintpid1motEN] Double-integrator and its control (1): motivation
17:54

** 

[YouTube ▶]

This video discusses the relevance in engineering practice of studying the control of a ‘double integrator’ system, transfer function $1/s^2$.

Indeed, the motion of a mass $M \frac{d^2 p}{dt^2} = F$ has such dynamics, if there is no friction... and, well, in technological applications of motion control one should try to make such friction as small as possible, at least in principle. Angular motion also has double integrator dynamics, changing masses for moments of inertia and forces for torques, obviously.

The transfer function $1/s^2$ is an ‘abstraction’: in many technological solutions, directly manipulating a force or torque is not possible, and an internal electronic system for ‘torque’ or ‘pressure’ control will be required, for example. This is called *cascade control* and is VERY frequently used to simplify and divide the complexity of a practical control problem into different ‘hierarchical levels’ where each level assumes that the lower slave controllers behave correctly so that they follow their references for force, torque, pressure, etc.

As a final example, in lateral position control of a drone, the non-vertical component of the force is approximately the weight times the sine of the roll angle; lateral attitude control of a drone may assume an A/s^2 model if there is internally a control system ‘much faster’ than the attitude control system that achieves a given pitch/roll angle using the appropriate accelerometers and gyroscopes.

All of this video is about the concept of a double integrator (low-friction motion control) in the abstract sense, although the ‘real technology’ of ‘motion control’ applications requires many more implementation details that are not the goal of a first introductory course in control theory, obviously.

In applications with high friction, gearboxes, etc., it is sometimes more appropriate to think of my manipulated variable as the ‘velocity of motion’, that is, to control position thinking of the transfer function $1/s$ (simple integrator); this type of abstraction is called ‘kinematic control’, in materials on robotics, for example. This is not considered in the present video.

[59: dintpid2tunEN] Double integrator and its control: trial and error controller
tuning [1: PD]
18:37

*** 

Materials: [[CÓD.:](#) PIDcontrolAPP1.0.zip]

[YouTube ▶]

This video discusses the tuning of a PD controller for a ‘double integrator’ type process (mass motion with small, negligible friction). The motivation for studying this process has been addressed in the video [??], since motion control is an important problem in practical applications.

Here, the PID control problem of this process is starting to be discussed. This will be done through purely ‘intuitive’ considerations, common sense. The theory will be discussed in the video [61] and following. Actually, for brevity, the present video will discuss only the *PD* proportional-derivative tuning, and integral action plus final discussion will be left for video [60], a continuation of this one.

The basic intuitive consideration is that a proportional control is ‘analogous’ to a spring on the moving mass. Therefore, as can be seen in simulation, without friction, the response will show sustained oscillations: the P control does not succeed in stabilizing.


To stabilize, ‘damping’, friction... is needed and that is the analogue of the derivative action: a ‘braking’ proportional to the speed. Braking can be proportional to speed or proportional to the ‘error speed’, which gives rise to two possible implementations of the derivative action, parameterized by a certain coefficient ‘c’, which are tested in simulation. In the end, the important thing is that the derivative action does stabilize. Note that the ‘ideal’ derivative action is not feasible: the high-frequency noise is infinitely amplified and, therefore, a noise filter is necessary in all practical applications, as shown in simulations here.

A summary of the ideas concludes the video. Final value justification in intuitive terms, disturbance rejection and integral action are left for a sequel video.

[60: dintpid2tunBEN]
11:25

Double integrator and its control: trial and error PID tuning [2, PID; 3., advanced tweaks]

Materiales: [[CÓD.:](#) PIDcontrolAPP1.0.zip]

*** 

[YouTube ▶]

This video constitutes ‘Part 2’ of the trial-and-error heuristic tuning of a PID controller for a double-integrator (motion control) plant; PD control was discussed in video [59], and in here, we’ll discuss integral action for zero offset in disturbance rejection, antiwindup, and a final summary, conclusions and remarks.

First, it is intuitively justified that the final value in response to reference changes does achieve a stationary position error equal to zero (justified by the fact that when one is in place and at zero speed, then zero force is required to stay there, which is exactly what a PD regulator does). However, in response to disturbances there is a constant position error... the explanation is by analogy with a tilted platform: then at the reference point zero force is not the required value to achieve equilibrium, so PD has an offset when equilibrium is reached. This error is corrected by the integral action.

The final part of the video discusses that integral action to optimize the response to disturbances (unmodeled input forces) worsens the response to reference and that, to avoid overshoot due to error accumulation, the proportional part can be ‘fooled’ by lowering a certain parameter ‘b’.... Saturation also requires antiwindup (to compensate for excessive integral accumulation in the initial phases of the transient).

With all this, a PID regulator with derivative filter and with ‘2 degrees of freedom’ and ‘antiwindup’ modifications is finally designed (in a completely ‘math-free’ way) that achieves a satisfactory response to step, ramp and unmodeled step force changes (input disturbances).

‘Intuitive’ tuning strategies are extremely important in practice, because, in simple processes, it could be a quick option to solve a problem by a technician not specialized in control theory.

2.2 Double integrator (motion control) case study

[61: dintteostEN] Double integrator, PD control: stability

***  20:59

Materiales: [[CÓD.:](#) DINTJustifTheoryPandPI.mlx] [[PDF](#)]

[YouTube ▶]

This video is the first in a series to present the theory that justifies the behaviour we had found by ‘trial and error’ when tuning a PID controller for a ‘double integrator’ process $2/s^2$, see video [59] for such hand-tuning ideas.


In this first video on the underlying theory, the generic closed-loop equations $e(s) = \frac{1}{1+GK}r(s) - \frac{G}{1+GK}d_u(s)$ are obtained, and $G = 2/s^2$ is substituted; the controller is also substituted in there, several versions actually, as follows.

First, a proportional controller is tested, checking that the loop CANNOT be stabilized, having sustained oscillations (i.e., they do not decay to zero) with $K_p > 0$.

Then, a PD controller is tested checking that, if both proportional and derivative constants are positive, then it CAN be stabilized. With ‘little’ derivative action there are oscillations, but if the intensity of the derivative action is increased, then there are a pair of real poles (overdamped response).

The analysis continues in the video [62], where the steady-state error is analyzed with PD and PID controls. The design of controllers by pole assignment will be discussed in later videos of the case study.

[62: dintteoerrEN] Double integrator, PD control: position and velocity errors (setpoint tracking and disturbance rejection)

*** 

13:51

Materiales: [[CÓD.:](#) DINTJustifTheoryPandPI.mlx] [[PDF](#)]

[YouTube ▶]

This video is a continuation of the video [61]; in that video the closed loop poles of a double integrator with PD control were analyzed (it was stable, oscillatory or not depending on the amount of derivative action); in the present video, the error in steady state is analyzed.


Applying the final value theorem with a control law $K_P e + K_D \frac{de}{dt}$ and $G(s) = 2/s^2$, the following situations are analyzed:

- Error with step (e_{pos}) and ramp (e_{vel}) setpoint changes. Both are zero with the control $K_P + K_D s$.
- Error with input disturbance step (non-zero final deviation) and ramp (infinite final deviation) increments.

The results are verified in simulation.

Note: if the derivative action were implemented as $K_D \frac{dy}{dt}$ (output derivative instead of error derivative), then the position error with step reference change would be zero, but not the velocity error (stationary error with ramp reference); this fact is stated, without proof, for brevity.

[63: dintPDplaceEN] double-integrator PD design via pole placement

*** 

23:38

[YouTube ▶]

This video designs a PD controller for the double-integrator plant $G(s) = 2/s^2$ to meet certain static (zero position error with reference step change, limited steady-state drift when subject to input disturbances) and dynamic (settling time less than or equal to 2 seconds, damping coefficient χ greater than or equal to 0.7) specifications.

The design is based on the final value considerations from the video [62] and on equating the characteristic equation $0 = s^2 + 2K_d s + 2K_p$ to that from a certain polynomial $0 = s^2 + 2\chi\omega_n s + \omega_n^2$ that meets the specifications.

The final part of the video simulates the calculated controller, with the correct result being verified when there is no saturation or measurement noise, which are the prior assumptions under which the theoretical calculations operated.

When saturation and measurement noise are switched on, the results are very different: derivative action noise needs to be filtered quite intensely and the transient needs to be slowed down so that the differences between 'theoretical calculations' and 'simulations closer to practice' are smaller (smaller gains both amplify noise less and saturate less, obviously). This idea is simply stated as a guideline; a redesign following it is not carried out, for brevity.

[64: dintPIDplaceEN] double-integrator PID design via pole placement

21:49



Materiales: [[CÓD.:](#) DINTfullPIDplace.mlx] [[PDF](#)]

[[YouTube](#) ►]


This video designs a full-fledged PID to meet zero position error specifications both under reference changes and under input disturbances (step) of a double integrator process. This concludes the case study that began with purely intuitive tuning in the video [60], and that the immediately preceding video [63] solved by calculating a PD because zero error was not required under input disturbances.

The final part of the video simulates the achieved performance in Matlab and incorporates saturation and measurement noise... This worsens the performance and a noise filter is tuned 'by hand', plus a 2-degree-of-freedom strategy to avoid 'derivative kick', and an antiwindup scheme is activated to avoid saturation problems in integral-action regulators. All of this is not usually the objective of a first introductory control course, but it indeed is of a 'second' course in which additional considerations are addressed so that PIDs work better 'in practice'.

Chapter 3

Control structures

3.1 Manipulated and Controlled Variable selection, 2x3 process

[65: sacerf1EN] Controlled/manipulated variable selection: setpoint tracking, SVD, Matlab example (1) ***  14:43


Materiales: [[CÓD.:](#) solParc231P1v2part1ENG.mlx] [[PDF](#)] [YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video, the first of a case study with a total of 7 videos, poses the problem of determining if a process $y = Gu$ has properly chosen controlled variables y and manipulated variables u , in order to increase y a minimum amount required for a given application (setpoint tracking). Only the steady state case (static gain matrix) is discussed.

In this first video, the problem is presented, and its resolution is addressed using SVD methodology (geometry of ellipses, or spheres of radius 1 once scaled). The scaling process is reviewed and, subsequently, the scaled transfer matrix and the minimum gain (smallest singular value) are calculated, checking that minimum gain is greater than 1 and that the conditioning of 2.9 is also satisfactory, so the problem is considered as feasible.

This brief reasoning is what is required in the pen-and-paper exams in my M.Sc. courses. Additional discussion and graphical representation of the ellipses underlying this methodology is discussed in the video [66], a continuation of this one.

[66: sacerf2EN] Controlled/manipulated variable selection: setpoint tracking, full SVD, reachable ellipsoid, Matlab (2) ****  14:09

Materiales: [[CÓD.:](#) solParc231P1v2part1ENG.mlx] [[PDF](#)] [YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video continues the case study that began in the video [65], which discussed the problem statement of selection/sizing of manipulated and controlled variables for setpoint tracking tasks.

In said video, minimum gain and conditioning were scaled and checked. In this video we will discuss additional concepts such as:

- Complete U,S,V analysis of the SVD decomposition. Easy and difficult maneuvers in input and output directions are analysed; the null space of the gain matrix is also pinpointed (direction of u without effect in steady state increment).
- Since the nullspace is highly aligned with actuator 2, the analysis is repeated assuming that actuator 2 is not incremented; It is verified that the gains and conditioning barely change, so the setpoint tracking problem continues to be feasible.

- The reachable output ellipsoid achievable with the unit sphere in actuators, as well as the circle of radius 1 and the circle of minimum gain, are graphically represented in the (scaled) output domain. It is verified that, since the minimum gain is $\geq \sqrt{2}$ then the input unit sphere sweeps all the square of vertices ± 1 in the output domain, guaranteeing the feasibility of the problem with geometry of polyhedra (which will be discussed in depth in the video [67], a continuation of this case study).

[67: sacerf3EN] Controlled/manipulated variable selection: setpoint tracking, polyhedra (linprog) Matlab example (3)

Materiales: [[CÓD.:](#) solParc231P1v2part1ENG.mlx] [[PDF](#)]

**** 18:40

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video addresses the same setpoint tracking problem as in the previous videos [65] and [66] of this case study, which are briefly reviewed.

Here, we address how to determine if there is a u within limits that verifies $Gu = v_i$, being v_i each one of the four extreme vertices of the rectangle of desired incremental outputs. It is done with [quadprog](#), justifying that since we are only interested in feasibility, the cost index can be anything, for example, minimize $\|u\|^2$.

The second half of the video discusses a couple of observations about the geometry of polyhedra:

First, in order to approximate the pseudoinverse (scaling) problem, the H matrix of the quadprog command should be $inv(E_u)^2$; thus, if the pseudoinverse/SVD solution is feasible, it would match the result of [quadprog](#).

Second, by “zooming out” the desired output increment vertices until the problem is no longer feasible, we will obtain an excess power margin with an analogous interpretation to the minimum gain (above 1) of the SVD.

The final part of the video draws, in the output space, the polyhedra achievable with the available input increments and the desired one. Since the achievable is greater than the desired, the problem is feasible. The last few minutes of the video superimpose the results of the SVD approach and the multifaceted approach of the same problem, in order to get an intuitive idea of their similarities and differences.

[68: sacerf4EN] Controlled/manipulated variable selection: total disturbance rejection Matlab example (4)

Materiales: [[CÓD.:](#) solParc231P1v2part1ENG.mlx] [[PDF](#)]

**** 17:55

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video discusses the same case study that started in video [65]. Here, we consider the effect of disturbances in $y = Gu + Hd$, and we seek to determine if the actuator limits are sufficient to fully compensate the effect of d on the controlled outputs y , rendering zero steady-state error.

To do this, we first analyze the singular values of $-pinv(G) * H$, appropriately scaled.

In the second half of the video, the problem is stated in terms of polyhedra geometry, using [quadprog](#) to check its feasibility.

The final part of the video graphically represents the meaning of the different options considered to address the problem.

Videos [66] and [67] addressed the setpoint-tracking problem in the same context.

[69: sacerf5EN] Controlled/manipulated variable selection: partial disturbance rejection, polyhedra quadprog (5)

Materiales: [[CÓD.:](#) solParc231P1Part1and2ENG.mlx] [[PDF](#)]

**** 13:46

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

In this fifth video of the case study that began in the videos [65], [66], [67] and [68], the partial cancellation of steady state disturbances is addressed, with the geometry of polyhedra.

First, the “admissible residual error” vertices are generated from the problem specifications and, with them, for each of them, the feasibility of a quadratic programming problem is addressed, which we will test by executing `quadprog`. The most appropriate cost index to minimize is briefly discussed, although this discussion would only be relevant if the real-time controller on the process were to use `quadprog` (predictive control, usually) and the disturbances were measurable or observable. Since we are only looking for feasibility, really $H = I$, $f = 0$ would work for us (and that is the only thing that would be required to my students in a ‘pen-and-paper’ exam).

In the second half of the video, optional additional considerations are discussed; in particular, the margin of increase of disturbances so that the problem remains feasible, and the graphical representation of the achievable polyhedra in the space of the outputs y for a better understanding of the concepts (the code is not analyzed in detail, which is left to the reader).

[70: sacerf6EN]
Controlled/manipulated variable selection: partial disturbance rejection, SVD (6)
**** 16:51

Materiales: [[CÓD.:](#) solParc231P1Part1and2ENG.mlx] [[PDF](#)]
[YouTube]

* ENLACE A SPANISH VERSION

This is the sixth video of the case study that started in video [65]. Here we discuss partial cancellation of steady state disturbances with SVD techniques; the ‘exact’ geometry of polyhedra (`quadprog`) was discussed in the video [69].

Basically, keeping the scalings of manipulated variables and disturbances from previous problems, a new output scaling based on the maximum tolerable error threshold is now discussed. With this new scaling, the problem is that of calculating an optimal u such that $\|e_{esc}\|^2 + \|u_{esc}\|^2 \leq 1$ when $\|d\| \leq 1$. Thus, each of the two terms being added will obviously be less than 1.

Since this is a least-squares problem, it is solved with pseudoinverse, and the SVD of the resulting matrix is checked.

The case study concludes in the video [71], which analyzes the transient case.

[71: sacerf7EN]
Controlled/manipulated variable selection: transient analysis, sigma plot (7)
*** 14:55

Materiales: [[CÓD.:](#) solParc231P1Part1and2ENG.mlx] [[PDF](#)]
[YouTube]

* ENLACE A SPANISH VERSION

This video concludes the case study that began with the video [65]. Here we discuss, in transient regime (frequency response, `sigma`), the reference tracking problem and the total disturbance cancellation problem.


The final part of the video discusses the possibility of simultaneous reference increments and disturbances, comparing with the initial results of only one of the two problems.

Partial disturbance rejection with frequency-dependent error bounds should be handled, possibly, with full-fledged \mathcal{H}_∞ optimal control; the singular-value plots in here are just a first easy approximation to such a problem.

3.2 Control with excess actuators (manipulated variables)

[72: dosact1EN] Control with excess actuators (1): load balancing, split range, override

Materiales: [TwoActuatorsSummaryENG.pdf]

***  18:58

[YouTube ►]

* ENLACE A SPANISH VERSION


This video discusses strategies for control structures with one “controlled” and two “manipulated” variables. The first three minutes motivate the problem by introducing practical examples of where these ideas may apply (exchanger with bypass, neutralization, heating in stages, ...), and proposes that there are two major divisions of the strategies to solve the problem: (a) those with a single controller in operation, and (b) those where a secondary setpoint for a secondary controlled variable is generated and, hence, there are two active controllers.

In this first video, the first option (a single active controller) is discussed, and the structures of load balancing, split range, and override (two controllers, but only one active in emergencies) are explained. Really, override is a bit ‘in the middle’ and, depending on opinions we could say that there is one loop or two loops

A second video [73], a continuation of this one, explains the strategies with secondary loop (cascade, gradual, ...).

[73: dosact2EN] Control with excess actuators (2): cascade extra actuator, gradual, double cascade

Materiales: [TwoActuatorsSummaryENG.pdf]

****  19:40

[YouTube ►]

* ENLACE A SPANISH VERSION

This video continues the discussion on strategies for control structures with 1 controlled and 2 manipulated variables. The discussion was started on video [72].

There were two main divisions of strategies to solve the problem: those with only one controller in operation and those where a secondary reference is generated, and there are two active controllers.


The first three minutes of the video review strategies with an active controller: load balancing, split range, and override; These strategies were discussed in detail in the video [72], prior to this one.

Continuation of this exposition, here strategies with a secondary reference are explained. The proposed ones are:

- *cascade control with extra actuator*, where the secondary reference is actually an operating point of one of the manipulated variables.
- *gradual control* where, if an intermediate sensor is available, heating, neutralization, etc., can be considered to be carried out by *stages*, with a secondary reference for that intermediate sensor, that a kind of multiloop control should maintain.
- *double cascade control*, where if there is an extra actuator and also an extra sensor, then the reference for the intermediate ‘gradual’ sensor is generated by the master ‘extra actuator cascade’ structure.

These cascade structures work best (ease of tuning) when there is ‘time-scale separation’, whereas the structures in the first video work best when both manipulated variables have comparable bandwidths, similar dynamics.

3.3 SVD decoupling


[74: dsvdintu1EN] 17:40	SVD decoupling and principal maneuvers in process control (1): theory outline	*** 
Materiales: [SVDdecouplingTSTtheoryENG.pdf]		[YouTube ▶]

*ENLACE A SPANISH VERSION

This video provides a brief theoretical review of the SVD decoupling technique in multi-variable control.

The video reviews the concept of “principal maneuvers”, input and output directions and null space. Then, we propose a change of variable so that $y^{SVD} = U^T y$, $u^{SVD} = V^T u$. With that change of variable $y^{SVD} = S u^{SVD}$ when $y = (U S V^T) u$ is the SVD decomposition of the static gain matrix. In this way, the apparent behavior between ‘virtual’ SVD variables is diagonal S , and regulators (slow ones, since the decoupling is at zero frequency) can be designed exploiting that idea, resulting in $u = V K(s) U^T e$, with a diagonal $K(s)$ separately controlling each principal maneuver.

If some principal maneuvers are not controlled (because of low gain or/and poor numerical conditioning), to avoid frequent saturation and sensitivity to modeling errors, then $u = V K(s) U^T e$ would be constructed with only a subset of columns of U and V (a sort of so-called ‘economy size’ SVD).

[75: dsvdintu2EN] 17:09	SVD decoupling case study (2): one controlled variable, two manipulated variables	*** 
Materiales: [Cód.: SVDdecouplingTSTeng.mlx] [PDF]		[YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses the concept of principal maneuvers and SVD decoupling in a ‘1 controlled output, 2 inputs’ process. To understand the “intuitive” meaning of what is proposed, it is illustrated with a physical example of heating a piece of material (a thermocouple gives the reading of the variable to be controlled) with both a resistor and a blower (cool air being blown by the latter).

Obviously, there are infinitely many solutions to $y = Gu$, and with appropriate scaling so that 1 means ‘100% effort up to saturation’, we would want to use, say, the one that minimizes $u_1^2 + u_2^2$, i.e., the least-squares criterion.

Specifically, the singular value decomposition $G = U S V^T$ of a matrix that is a *row* vector has $U = 1$, $S = \text{norm}(G)$, and $V = G^T / \text{norm}(G)$, i.e., a norm 1 scaling of the original matrix; directions orthogonal to G form the null space. The pseudoinverse is $G^T / \text{norm}(G)^2$. If the theory is not familiar to you, check the video [74], prior to this one.

In fact, nothing is too interesting about SVD in a one-output process, in the sense that, actually, the input directions with non-zero effect are simply proportional to the gain of each actuator: this is just a first motivational toy example.


But the objective is to capture the intuition behind the fact that least squares solutions distribute the control effort proportionally to the gain and, therefore, we could think of the only controlled variable being controlled by a “virtual” actuator that moves the physical actuators in a coordinated manner, replicating the same actuator command, but multiplied by the gain.

The video [??] continues with the “dual” case of several outputs to be controlled with a single manipulated input.

3.4 Case Study: mixer/heater process

[76: estrx2aEN]

Two tank mixing + heating system (1): basic multiloop control, case study

***  12:45

Materials: [ttceENG.pdf]

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video presents the problem of a control case study of a system with two tanks and two heating elements, built with the purpose of diluting a certain reagent in water; the reagent must be supplied downstream at a certain temperature. The demanded downstream flow will be considered to be a disturbance. The point of operation will be set to diluting 1 liter of reagent in 5 of water (concentration 1/6 at output).

First, the selection of primary references is discussed: obviously, temperature and concentration in tank 2 are in the set, because that is what the process is built for. A primary level reference is also placed, for safety and so that the residence time of the fluids has less variability (but, well, residence time will depend on flow rates, so it will not keep constant anyway).

Second, the presence of four manipulated variables is discussed: reagent valve, solvent valve, exchanger in the inlet pipe, and heater in the outlet tank.


There are seven sensors, but in this video, we first consider choosing as sensors only those of the primary references (temperature, concentration, level) and thinking about a multi-loop strategy. By “common sense”, it is argued that the water flow will be used for level control, the reagent flow will be used for concentration, and the heater in the outlet tank will be used for temperature.

If you want model-based rather than “common sense” detail on multiloop pairings in the level and mixing subsystem, refer to video [78] detailing modeling, linearization, and relative gain array computation for this case study.

The excess of manipulated variables and sensors can be used to improve performance in cascade, ratio and feedforward structures, discussed in the video [77], a continuation of this one.

[77: estrx2bEN]

Two tank mixing + heating system (2): advanced cascade feedforward ratio control case study

****  21:45

Materials: [ttceENG.pdf]

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video continues the case study discussed in the video [76] of level, temperature, and concentration control in a two-tank liquid system. In said video, the problem was stated, and the multi-loop proposal from common-sense argumentations (justified with calculations in the video [78]) was reviewed as the most basic option that, if it worked, would not require worrying about anything else.

Assuming that it doesn't work well and that additional instrumentation is acquired, as depicted in a schematic of the process, this video proposes to use:

- Extra sensor cascade control for inlet flow control (and also in temperature loop),
- Feedforward with outlet flow sensor,
- Extra actuator cascade control to take advantage of two heating elements in temperature control,
- Ratio control for the concentration loop
- Decoupling to reduce the effect of level control on temperature.

With this, a final structure is designed: it uses 7 sensors and 4 actuators to control 3 primary references.

The video [??] addresses an optional discussion on the relationships between inverse decoupling, ratio control, and direct decoupling. In any case, the discussion is optional, specialized, and it is not necessary to understand what is proposed here.

[78: mzrgaEN] static mixing + flow control: multiloop control via relative gain array (RGA), theory

Materiales: [ttceENGDecRtAll2.pdf]

*** 12:02

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video studies in more detail the possible pairing in a “static” mixing for decentralised control of a process where controlled variables are total flow rate and mix concentration through modeling, linearization, and relative gain array (RGA) calculation of said process. It is motivated, for example, in the case study of the video [76], where conclusions were drawn by “common sense” argumentations, instead of based on models, which is what we do here.

[79: mzzratdc1EN] static mixing + flow control: ratio control, linearization and relation to multiloop/decoupling

14:13

Materiales: [ttceENGDecRtAll2.pdf]

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video continues the model-based theoretical analysis of static mixing at the inlet of a 2-tank process, which began in the video [78]. There, we modelled and linearized the static mixing stage, and derived the recommended pairing for multi-loop control based on the relative gain array (RGA).

Here, we discuss the *ratio* control strategy for the concentration loop, starting with a merely intuitive justification (similar to the model-free argumentation made in the video [77]). But the main objective of this video is the theoretical analysis of said ratio control, specifically its linearization: it can be observed that the linearization of the ratio control can be understood as a multi-loop + decoupling (inverse form) of the effect of the level loop on the concentration one. With this interpretation, it is justified that the multi-loop concentration control would be obtained from the ratio control one just by multiplying it by the nominal water flow.

The relationship with decoupling is discussed in more depth in the video [80], the sequel to this one whose watching is recommended.

[80: mzzratdc2EN] static mixing + flow control: decoupling in direct/inverted form, linear and nonlinear

18:54

Materiales: [ttceENGDecRtAll2.pdf]

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video complements the video [79] in which a *ratio control* structure was proposed, and its linearization could be interpreted as (partial) decoupling in inverted form.

Here, inspired by this, the complete inverted-form decoupling is proposed (an algebraic or parasite loop appears, of course) in linearized form and, subsequently, the ratio structure of the previous video is also modified to incorporate this complete inverse decoupling in NONlinear form.

The last part of the video discusses decoupling in a “direct” form (without looping back as the inverted-form structure), by using matrix inverse in the linear case, or by some straightforward calculations with proportions in the non-linear case.

Reverse decoupling creates a parasitic loop that needs to be broken or stability-checked but, on the other hand, it protects against saturation in some way (tracking-mode anti-windup would be needed, possibly); direct decoupling has no parasitic loops, but in case of saturation the ratios could drift away from the desired ones.

The generation of the “inversion” of the mixing equations to a “dynamic” case is called feedback linearization, see video [82], for instance.

3.5 SUMMARY: the BIG picture in day-to-day control in industry

Chapter 4

Other isolated things for the moment being.

[81: imcQcteEN] IMC control of a fast process: constant Q *** 09:21
Materiales: [[CÓD.:](#) IMCQctEnglish.mlx] [[PDF](#)] [YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses IMC control of a “fast” process where its open-loop settling time is already fast enough for a given application, and we don’t wish to speed it up in closed-loop to, for example, avoid unnecessary actuator oversizing.

In this case, the explicit IMC is easy, it is simply a question of proposing $Q=1/\text{dcgain}(G)$. When transformed to a conventional regulator (implicit IMC) with $K = Q/(1 - GK)$, it results in a closed-loop integral action controller, as the theory predicts. So, in the event of input and output disturbances and setpoint changes, it exhibits a closed-loop dynamics that achieves zero error with closed-loop poles in the same place as the open-loop ones, which was our ultimate goal.

[82: fbLin1EN] Feedback linearization and decoupling: two-input two-output **** 14:36
example (hand-made, no Matlab)
Materiales: [FbLinEx1ENG.pdf] [YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses a simple academic exercise to calculate the decoupling and feedback linearization equations of a 2nd order system, with 2 inputs and 2 outputs, relative degree 1 in both of them (inputs appear with first derivatives of outputs). The system also has no singularities or “zero dynamics” to analyze.

Please review the video [1] if you have forgotten the manipulations with partial derivatives and the multivariable chain rule.

4.1 Linear matrix inequalities (LMI), Semidefinite programming (SDP)

4.1.1 LMI problems with ellipses

[83: lmielout1EN] LMIs: Ellipsoid containing other ellipsoids/polyhedra, ****
Yalmip/Sedumi/Matlab (1): minimum major axis 18:43
Materiales: [[CÓD.:](#) ElipsLMIout.mlx] [[PDF](#)] [YouTube ▶]

This video presents how to obtain, via linear matrix inequalities (LMI) or *semidefinite programming* (SDP), according to other texts, the “smallest” ellipse that contains a given polygon and another ellipse.

The approach can be generalized to more ellipses and more polyhedra in more dimensions; the LMI code would be practically identical.

The first part of the video discusses the conditions for an ellipse $x^T P x \leq 1$ to contain a polyhedron with vertices v_i (that is, $v_i^T P v_i \leq 1$) and to contain another ellipse $x^T M x \leq 1$, that is stated as $M - P \succeq 0$; The second case is detailed and proven in the video [7].

The second part of the video discusses what needs to be “optimized” (objective function): in this first video, it is decided to minimize the length of the semimajor axis of the ellipse, that is, we wish to minimize the radius of the smallest sphere that contains the sought ellipse. This is done by minimizing ρ , adding the constraint $P - \rho^{-2} I \succeq 0$. Actually, to make the objective function and constraints linear in the decision variables, we change to maximize β subject to $P - \beta I \succeq 0$.

The video details all the YALMIP code that must be entered to solve the problem in a Matlab environment.

Note: by minimizing the size of the semimajor axis of the ellipse, the minor semi-axis is “free” and the solution is not unique. Discussing this topic as well as the minimization of the volume (area in a 2D case) of the desired ellipsoid is the objective of the video [84], a continuation of this one.

[84: lmielout2EN] 09:39

LMIs: Ellipsoid containing other ellipsoids/polyhedra,
Yalmip/Sedumi/Matlab (2): minimum area

Materiales: [[CÓD.:](#) ElipsLMIout.mlx] [[PDF](#)]

[YouTube ▶]

This video is a continuation of the video [83]. In said video, the LMI restrictions were proposed so that an ellipse included a polyhedron and another given ellipse. Then the radius of the circumscribed circumference of said ellipse was minimized, thereby minimizing the semimajor axis of the searched ellipse.

In this video it is discussed that, since the minor axis of the result is not unique, you can try to minimize it with 10,000 times less “force”... adding $1e-4$ times the trace of the matrix to the objective function, you get something similar to the ellipse with “minimum minor axis” among the many ellipses with “minimum major axis” feasible as an optimal solution to the problem we are considering.

The second part of the video discusses how to truly obtain the minimum area ellipse, using the Yalmip geomean command. This would obtain the minimum-volume ellipsoid in three or more dimensions.

[85: lmielin1EN]

LMI demos: largest ellipse inside other poliedra/ellipses (1)

Materiales: [[CÓD.:](#) ElipsLMIlin.mlx] [[PDF](#)]

**** 16:49

[YouTube ▶]

This video discusses how to obtain using linear matrix inequalities (LMI) the largest ellipse within a polygon (well, it also supports sides that are an ellipse segment).

The presentation is divided into two videos.

In this first video the problem is posed, and we consider how to obtain the matrix P such that $x^T P x \leq 1$ is within the predetermined intersection set of polygons/ellipses.

The restrictions are coded as LMIs, and two possible cost indices are proposed:

- Minimize maximum eigenvalue of P , which is equivalent to maximizing semi-minor axis, that is, finding the circle with the largest radius contained in the search region.

- Minimize the trace of P ... minimize the sum of the inverse squares of the lengths of the semi-axes... This gives a ‘large’ ellipse as a result of the optimization but its meaning is not entirely clear. Indeed, searching for the ellipse with the largest area will be the objective of the video [86], a continuation of this one.

[86: lmielin2EN] LMI demos: largest ellipse inside other poliedra/ellipses (2), largest area (geomean)

16:18

Materiales: [[CÓD.:](#) ElipsLMIlin.mlx] [[PDF](#)]

[YouTube ▶]

This video discusses how to obtain, using linear matrix inequalities (LMI), the largest ellipse within a polygon (well, it also supports curved sides that are an ellipse segment). It is a continuation of [85] where the problem was posed and some possibilities were discussed with an ellipse in the form $x^T P x \leq 1$, P being a decision variable.

In this second video, Q is considered a decision variable that defines the ellipse $x^T Q^{-1} x \leq 1$. The ‘inverse’ form is used for convenience to express the optimization problem as a convex one. Indeed, *maximizing* the determinant of Q (well, actually its square root, geometric mean of the eigenvalues $\sqrt{\lambda_1 \lambda_2}$) can be done with the `geomean` operator of Yalmip.

Conveniently, the area of the ellipse is proportional to said geometric mean, so that we can solve the requested problem of maximum area. Furthermore, translating the restrictions to inverse form will require using ‘congruence’ and ‘Schur’s complement’ so that the video will serve to illustrate the application of these results, which are very useful in control theory developments with LMIs.

[87: distelliEN] Distance between ellipses: semidefinite programming (SDP/LMI, linear matrix inequalities)

Materiales: [[CÓD.:](#) DistanceEllipsesENG.mlx] [[PDF](#)]

**** 10:59

[YouTube ▶]

This video explains how to compute the distance between two ellipses (minimum distance between any pair of points in them), by optimizing a function subject to linear matrix inequalities (LMI), that is, semidefinite programming (SDP).

The basic idea is to express the ellipses $(x_i - c_i)^T Q_i^{-1} (x_i - c_i) \leq 1$, $i \in \{1, 2\}$, as LMI sets, using Schur’s formula. Also a distance bound $(x_1 - x_2)^T (x_1 - x_2) \leq d^2$ will be expressed as an LMI on decision variables x_1 and x_2 using the Schur formula, and $\xi \equiv d^2$ will be minimized, as the overall semidefinite programming setting is a particular case of convex optimization.

Results are checked numerically and plotted for three examples, including an example where the ellipses do intersect and, therefore, the distance is zero (although the numerical solver does not yield exactly zero as a result, due to the tolerances and finite precision of the code). The actual software used in this video is Matlab, jointly with the free SDP toolboxes YALMIP and SeDuMi.

[88: elinnccEN] Maximum volume ellipsoid inside polyhedron and other ellipsoids: 2D example, Matlab (LMI/SDP)

Materiales: [[CÓD.:](#) ElipINElipORPoly.mlx] [[PDF](#)]

***** 15:36

[YouTube ▶]

This video discusses the problem of obtaining the maximum volume ellipsoid inside a convex body defined as the intersection of polyhedra and ellipsoidal sets, not centered neither symmetric with respect to the origin as in other prior videos of the collection; well,

as we are detailing a 2D example, we will actually speak about polygons and maximum area ellipse, but the ideas efficiently generalise to higher dimensions in a trivial way.

We use Matlab+YALMIP+SeDuMi to solve the associated semidefinite programming, setting linear matrix inequalities and a ‘geomean’ objective function (used by Yalmip/SeDuMi instead of ‘logdet’).

4.1.2 Geometry of generic LMI/SDP-representable sets

[89: lmisets1EN] **LMI sets (SDP-representable sets, spectrahedra): definition, basic properties and 2D examples**

Materiales: [[CÓD.:](#) LMIsExamplesPart1.mlx] [[PDF](#)]

**** 17:42

[YouTube ▶]

This video defines an ‘LMI set’, also known as ‘SDP-representable’ set, as the set of feasible values of a linear matrix inequality $LMI(x) \succeq 0$, where $\succeq 0$ means being positive semidefinite. The name of ‘spectrahedron’ (plural spectrahedra) is, too, used to denote these sets of feasible values of a semidefinite program.

Convexity of the cone of positive semidefinite matrices implies that LMI sets are convex sets. As positive-semidefiniteness of a matrix is equivalent to all principal minors being non-negative, then, LMI sets are semialgebraic ones with polynomial boundary.

Examples of LMI sets are given, such as circles, ellipses, polyhedra, cones and a cubic egg-shaped set. All examples are in 2D, for ease of visualization, albeit of course LMI sets can be defined in any dimension.

The next video [90] will describe examples of the so-called ‘lifted’ LMI sets, projections on a reduced dimension of LMI sets in a higher-dimensional space.

[90: lmisets2EN] **Lifted LMI sets, 2D examples (Matlab, YALMIP)**

Materiales: [[CÓD.:](#) LMIsExamplesParts1and2.mlx] [[PDF](#)]

**** 13:56

[YouTube ▶]

This video presents examples of ‘lifted’ LMI sets (SDP-representable sets), i.e., projections of higher-dimensional LMI sets, involving additional ‘dummy’ decision variables. This video is a continuation of video [89], which introduced the concept of LMI set, its convexity and polynomial boundaries, and posed some non-lifted examples. The first three minutes of the present video review such concepts.

Then, several examples of lifted sets are posed:

- a square with rounded edges, a 4D LMI set projected in 2D,
- an ellipse defined by its foci, also a 4D to 2D lifted set.
- a 3-ellipse (guitar pick), generalising from 2 to 3 the number of foci, so it is represented as a 2D projection of a 5D LMI set,
- a generic, arbitrarily invented 4×4 LMI in 5 variables, giving a convex shape with 4th-degree polynomial boundary (well, an extra linear constraint is added so degree might be considered to be 5) in the higher dimension.

Video [91] will discuss the details on plotting code, plus geometric and optimization problems that may be solved with these sets.

[91: lmisets3EN] **LMI sets (3): plotting routine and applications**

Materiales: [[CÓD.:](#) LMIsExamplesParts1and2.mlx] [[PDF](#)]

**** 09:54

[YouTube ▶]


This third video closes the introduction to LMI sets (and lifter ones). In here, details of the plotting code used in previous videos [89] and [90] is presented (it's based on a ray-tracing idea, using YALMIP *optimizer* command, which is a faster pre-compiled version of *optimize* for repeated problems). Computation of the analytic center of an LMI set is also discussed.

The last part of the video discusses the kind of problems that can be addressed with SDP optimization software on LMI sets:

- Checking for intersection of several LMI sets (collision detection, say, in a planning problem)
- Optimization of linear functions on them (computation of tangent planes)
- Optimization of SDP-representable convex functions (as an example, the minimum distance between LMI sets; an explicit example of that is addressed in video [??] so it is omitted here).

Determining if one LMI set is contained into another may require, in a general case, other tools providing sufficient conditions via *Sum of Squares* (SOS) and Positivstellensatz multipliers; details on these extensions to LMIs is out of the scope of this introduction.

A later video with discussion on scaling, perspective cones and convex hull of LMI sets has been added, see [92].

<p>[92: lmisets4EN] 20:15</p>	<p>LMI sets (4): scaling, perspective cones, set interpolation, convex hull</p> <p>Materiales: [CÓD.: LMIsetExamplesPart3Scaling.mlx] [PDF]</p>	<p>***** </p> <p>[YouTube ▶]</p>
-----------------------------------	--	---

This video discusses some transformations and operations with LMI sets (SDP-representable sets), introduced in videos [89] and [90].

The basic ideas are the following two ones:


1. Even if we name LMIs as ‘Linear’ matrix inequalities, they are actually ‘affine’, with a constant term $LMI(0)$ plus a truly linear one $LMI_L(x)$ so $LMI_L(ax + by) = a \cdot LMI(x) + b \cdot LMI(y)$. The linear component is, henceforth, $LMI_L(x) = LMI(x) - LMI(0)$.
2. if x is in the set such that $LMI(x) \succeq 0$, the transformation $z = Tx + q$ converts it to an LMI set in variable z given by $LMI(T^{-1}(z - q)) \succeq 0$.

From these two ideas, in this video we discuss:

- Scaling an LMI set
- Forming the ‘perspective cone’ in one more dimension, so the sections are the LMI set, whose projection is the union of all scalings in an interval
- Changing the apex (scaling point)
- Convex hull of LMI sets
- Convex interpolation of LMI sets (direct sum of λ and $(1 - \lambda)$ scaled versions).

Chapter 5

Robust Control

[93: prth1EN] Robust performance: small-gain sufficient condition (h-infinity norm bound) ****  10:59

Materials: [RobustPerfEnglish.pdf] [YouTube ▶]


This video presents the main “small gain” condition to ensure robust performance, the fact that the infinity norm of a given generalised plant P is less than one is a sufficient condition for robust performance (under some scalings and assumptions).

The video proves the above assertion and also relates it with the small gain conditions for robust stability and nominal performance (arising, in both cases, from a “fragment” of P).

Erratum: in audio, I often say “less than” ($<$) but slides write “lower or equal than” (\leq); the slides are correct, I just kind of abbreviate the speech but just pay attention to the correctly written expressions in the slides. Likewise, I say “negative” instead of “non-positive”; excuse my imprecise speech.

This condition will be further relaxed in forthcoming videos addressing the “scaled small gain” modification, and it will give rise to H-infinity control synthesis methods for robust performance (with so-called “multipliers” or “scalings”) conforming the basis of the mu-synthesis techniques.

A robust performance Matlab case study appears in videos [95], [96], [97].


[94: stabmrgEN] Robust stability margins: robstab, wcgain, robgain example 2nd order system with uncertain damping 12:56 **** 

Materials: [Cód.: robstabwcgainOLEnglish.mlx] [PDF] [YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses the meaning of the robust stability (or robust performance) margins given by the **robstab** (robust stability margin), **wcgain** (worst case gain) and **robgain** (robust performance margin) commands of Matlab’s Robust Control Toolbox.

For simplicity, the case of a second order uncertain open loop system (mass-spring-damper with uncertain damping) is addressed, although the most common use case of the commands is, indeed, for problems of robust control design, i.e., h-infinity and mu-synthesis; but here the objective is to understand the command outputs with a system that we physically understand, to verify that it coincides with what was intuitively expected.

[95: cerp1EN] Robust performance case study (Matlab) 1: problem statement and simulation-based robustness validation ***  15:32

Materials: [Cód.: RPCE1English.mlx] [PDF] [YouTube ▶]

*ENLACE A SPANISH VERSION

This video starts the discussion of a robust performance analysis of a given PID controller for the process with transfer function $G(s) = 5/(s+1)^2$. This video simply introduces a PID designed by trial and error. Its nominal performance of loop error and reference to control action, both in time and frequency, are analyzed. They are considered satisfactory and the problem arises of how to guarantee, given a certain estimation of the modeling error (unstructured additive uncertainty, bound size 0.2), that the response of the system to the modeling error "resembles" the simulation of the nominal (guarantee robust performance).

In this first video, a simulation-based analysis of the achieved performance is pursued. Indeed, exhaustive simulation is appealing and frequent in practice, as we can simulate nonlinearity, very complex dynamics, faults, etc. However, maybe our simulations look well because of "good luck"? , thus, a theoretical analysis is also welcome. The theoretical approach to give robustness "margins" for this problem will be discussed in the video [96], a continuation of this one.

[96: cerp2EN] Robust performance case study (Matlab) 2: generalised plant, norm-based performance PID certification

Materiales: [[CÓD.:](#) RPCE1English.mlx] [[PDF](#)]

**** 16:20

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video continues the case study started in video [95], dealing with the discussion of a robust performance analysis of a given trial-and-error tuned PID controller for the process with transfer function $G(s) = 5/(s+1)^2$. In the first video, a simulation-based analysis of the achieved performance was pursued.

Now, we aim to obtain theoretical guarantees of robust performance. So, first a setpoint-to-error gain bound is defined in the frequency domain. Second, a generalised plant with uncertainty is built. Last, the robust performance certification via \mathcal{H}_∞ norm bound (video [93]) is used.

We see that expressing the "size 0.2 uncertainty" as either $0.2 \cdot \Delta$ or $\Delta \cdot 0.2$ gives different results. The second choice allows for proving robust performance, but that inspires expressing the uncertainty as $S^{-1} \cdot \Delta \cdot S \cdot 0.2$ and considering S to be an additional degree of freedom. Indeed, that's a scaling that allows to generalise the robust performance check to the so-called *scaled small gain* theorem. The idea is further explored in video [97], where \mathcal{H}_∞ optimal controllers are also pursued instead of a hand-tuned PID we were using up to this moment for simplicity in the illustration of key ideas.

[97: cerp3EN] Robust performance case study (Matlab) 3: scaled small gain and h-infinity optimization (approx. musyn)

Materiales: [[CÓD.:](#) RPCE1English.mlx] [[PDF](#)]

***** 13:31


[YouTube ▶]

*ENLACE A SPANISH VERSION

This video is the last in the case study started in video [95] and continued in video [96].

Robust performance bounds for a hand-tuned PID were proved in the referred videos. In here, we replace the PID by an **hinfsyn** controller. This allows us to both increase the modelling error bound and increase the closed-loop bandwidth. Scaled small gain is needed, playing with the multiplier S so a closed-loop infinity norm bound of 1 is achieved.

Of course, the procedure in this video is crudely simplistic, merely illustrative: a better-tuned PID might have achieved comparable performance, the plant cancellation by the h-infinity controller may result in bad input disturbance performance, and the manual search of the multiplier can be replaced by iterative-convex-optimization steps, as done by the command **musyn**, which actually is "the way to go", explored in other videos of this collection... in particular [98] and [99].

<p>[98: cerp4muEN] 11:27</p>	<p>Robust performance case study (Matlab) 4: mu-synthesis (building uncertain generalised plants)</p> <p>Materiales: [Cód.: RPCE1musynENGLISH.mlx] [PDF]</p>	<p>***** </p> <p>[YouTube ▶]</p>
----------------------------------	---	---


* ENLACE A SPANISH VERSION

This video continues the case study of the videos [95], [96] and [97]. In these videos, the problem of robust performance was stated, a generalized plant was formed with uncertainty, and multipliers (scalars) were searched for manually, by trial and error.

Although of interest for didactic purposes in order to better understand the scaling small gain theorem, you rarely have to do things as said in these videos, because all that detail is taken care of by the internals of `musyn` mu-synthesis command. This video compares the generalized plant that `musyn` transparently builds when it is executed (we can do it explicitly with the command `lftdata`) with the plant that we built manually in the video [96], verifying that they match.

Then, the multiplier lookup is done on one line with `musyn`; the multiplier for uncertainty LTI can be dynamic because if $\Delta(s)$ is SISO then $\Delta(j\omega) \cdot D(j\omega) = D(j\omega) \cdot \Delta(j\omega)$ and can be used in scaled small gain theorem. The `musyn` command with default options uses a scaling $D(s)$ of order 8 here, and produces a regulator of order 11.

For brevity, order reduction, closed-loop simulation, and comparison to musyn-PID results will be discussed in the video [99], a continuation of this one.

<p>[99: cerp5muEN] 16:15</p>	<p>Robust performance case study (Matlab) 5: mu-synthesis order reduction, PID tuning, loop simulations</p> <p>Materiales: [Cód.: RPCE1musynENGLISH.mlx] [PDF]</p>	<p>***** </p> <p>[YouTube ▶]</p>
----------------------------------	---	---


* ENLACE A SPANISH VERSION

This video continues the case study started in the video [95] and, in particular, it analyses the `musyn` command output which was introduced in the preceding video [98]. The first two and a half minutes here summarise the preceding videos.

The scaled small gain multiplier lookup is done on one line with `musyn`; the multiplier for uncertainty LTI can be dynamic because if $\Delta(s)$ is SISO then $\Delta(j\omega) \cdot D(j\omega) = D(j\omega) \cdot \Delta(j\omega)$ and can be used in scaled small gain theorem. The `musyn` command with default options uses a scaling $D(s)$ of order 8 here, and produces a regulator of order 11.

The initial part of the video discusses reducing the order of the mu-synthesis regulator (order 11) with `balred` and `freqsep` down to order 2. The result is very much like a PID, so in the second half of this video, `musyn` is used to directly optimize the parameters of a tunable PID, giving a very similar result.

Last, simulations in time and frequency domain of the PID and the musyn full-state-state (+ order reduction) regulator are shown; the simulations are satisfactory both in time and frequency domains, with random plants within the family with additive uncertainty for which everything has been designed. This concludes the case study.

<p>[100: gapm1EN]</p>	<p>Normalised factorisation uncertainty: nu-gap metric and its geometric interpretation (simplified, real, SISO) -1-</p> <p>Materiales: [nugapmetricrealEnglish.pdf]</p>	<p>*****  10:50</p> <p>[YouTube ▶]</p>
-----------------------	---	---

* ENLACE A SPANISH VERSION

This video discusses the geometric interpretation of the *normalised coprime factor uncertainty*, dealt with with matlab commands `ncfmargin`, `gapmetric`, `ncfsyn`.

Specifically, we consider a simplified SISO case in which we have two numbers p_1 and p_2 , real ones.

Each of them, say p , can be represented by an infinite number of pairs $(d, n) \in \mathbb{R}^2$ such that $n/d = p$, so p can be mapped to a straight line because $(d, n) \equiv (qd, qn)$ for any $q \neq 0$. Then, p is the slope of such line. The normalised representation is the intersection of the line with the unit radius circumference.

In this learning object, it is shown that the ν -gap (minimum coprime factor uncertainty) that transforms $p_1 \equiv (d_1, n_1)$ onto $p_2 \equiv (d_2, n_2)$ is the *sine* of the angle difference between the lines associated to each of the p_i .

Further details on numerical computations of the ν -gap and an alternate geometric interpretation appear on the video [101], a continuation of the present video.

[101: gapm1bEN]
14:16

Normalised coprime factorisation uncertainty: nu-gap metric and its geometric interpretation (simplified, real, SISO) -2-

**** ☰

Materials: [nugapmetricrealEnglish.pdf]

[YouTube ►]

This video is a continuation of the video [100], in which a number p was associated with an infinitude of fractions $p = n/d$ where (d, n) laid on a straight line with slope p . Normalised representation was the point of the line where $d^2 + n^2 = 1$. Also, the minimum-norm disturbance to p_1 so it rendered it equivalent to p_2 was the sine of the angle difference between the lines associated to p_1 and p_2 .

In this video, we show two formulae to obtain the numerical value of the *nu*-gap, either from the normalised description, or from the original (unnormalised) one.

Also, we illustrate a second interpretation as a chordal distance on a stereographic projection, which is the original interpretation proposed in the seminal works of Vinnicombe.

Of course, the full matrix-valued complex case is developed in the cited works, but it is not considered here as the focus is simplicity and intuitive understanding. A glimpse of the underlying ideas appears in video [102], continuation of this one.

[102: gapm2EN]

Normalised factorisation uncertainty: nu-gap metric geometric interpretation, general case (no proofs)

**** ☰ 11:35

Materials: [nugapmetric2English.pdf]

[YouTube ►]

*ENLACE A SPANISH VERSION

This video extends the discussion of the videos [100] and [101], which reviewed the geometric interpretation of the ν -gap in the real case. Here, the notions are generalized to the complex case of ν -gap distance between $P_1(j\omega) \in \mathbb{C}$ and $P_2(j\omega) \in \mathbb{C}$, also interpretable in the SISO case as chordal distance in stereographic projection. Transfer matrix formulae for MIMO cases are stated without proof, and briefly commented upon. Matlab's `gapmetric` command is related to the ideas and computations here described.

The usefulness of all this lies in the fact that, given a controller K , if the ν -gap between P_1 and P_2 is smaller than the normalised coprime factorisation robustness margin (`ncfmargin`) of the loop P_1, K then P_2 will be robustly stabilised by K .

[103: obsmu1EN]
13:26

LFT modelling of a system with uncertain damping for force estimation

**** ☰


Materials: [Cód.: MuvsKalvsHinfilerENG.mlx] [PDF]

[YouTube ►]

*ENLACE A SPANISH VERSION

This video presents the modeling of a mass-spring-damper system with an uncertain damping coefficient, subject to an unknown force input with limited bandwidth (that is, it will be the output of a first-order filter). The LFT modeling is detailed, leaving the uncertainty separated from the plant. The outputs that will be necessary to solve a problem of optimal design of force estimators will be also discussed. Here we insist on the concept of LFT uncertainty modeling, because I consider it an important didactic objective per se... but in this case of real parametric uncertainty, we can leave the job of setting up the LFT to Matlab and make everything easier, as seen for this same system in the video [94].

The actual design of the observers via `h2`, `hinf` or `mu` synthesis will be addressed in the video [104].

[104: obsmu2EN] 19:55	Force estimation in an uncertain mechanical system: <code>h2</code> , <code>hinf</code> , <code>musyn</code> Matlab example	***** 
Materiales: [Cód.: MuvsKalvsHinfilerENG.mlx] [PDF]		[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video addresses the design of an optimal estimator (in a certain sense) of the force acting on a mass-spring-damper system with uncertain damping coefficient.


Although it's quickly reviewed here, a more extensive explanation of the uncertain LFT modeling part is covered in the video [103], which you might want to watch, prior to this one, depending on your previous familiarity with these topics.

In this video, generalized plants and weighted generalized plants are posed to address the problem.

We review the `h2syn` and `hinfsyn` designs on the nominal plant, and the `musyn` design on the plant with uncertainty.

Performance is preliminarily discussed in terms of margins `robstab` (not relevant in "observer" problems), `wcgain` and `robgain`.

The result of `musyn` is a high-order estimator. Order reduction and simulation in time and frequency will be addressed, for brevity, in video [105], continuation of this one.


[105: obsmu3EN] 13:53	Force estimation in uncertain mechanical system: <code>h2-hinf-musyn</code> performance analysis, frequency domain	**** 
Materiales: [Cód.: MuvsKalvsHinfilerENG.mlx] [PDF]		[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video continues the case study of the design of an "estimator of an internal variable" (observer) of a system with an uncertain model, which began in the video [103] (modelling), continuing in the [104] (generalized plant and optimal design `h2`, `hinf`, `mu`).

Formally, the theory pursues guarantees in the frequency domain; thus, in this video the behavior of the observer in the frequency domain is discussed, both nominally and in the event of modeling error, comparing the `h2syn`, `hinfsyn` and `musyn` results.

Since what will be seen in a real application are signals in the time domain, we will dedicate a video ([106]) that will conclude the case study with simulations in the time domain incorporating measurement noise.

[106: obsmu4EN] 19:20	Force estimation in uncertain mechanical system: <code>h2-hinf-musyn</code> performance analysis, time domain	***** 
Materiales: [Cód.: MuvsKalvsHinfilerENG.mlx] [PDF]		[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video is the conclusion of the case study of the design of an estimator of an internal variable (observer) of a system with an uncertain model, which began in the video [103] (modelling), continuing in [104] (generalized plant and optimal design h_2 , h_{∞} , μ) and in [105] (frequency analysis of the results).

Although formally the theory just pursues guarantees in the frequency domain, in a real application we will actually see signals in the time domain. Thus, this video discusses the performances when subject to step and sinusoidal inputs with random measurement noise. It is argued that, since h_{∞} or μ do not optimize the variance of the output, it would be necessary to experimentally check or/and add high-frequency filtering to the limit templates to ensure that the variance when facing measurement noise is acceptable. The case in continuous time also has its peculiarities because white noise is an infinite variance signal... The objective of this video is not to go into those subtleties but at least approach how to craft a sensible simulation.

Chapter 6

Manipulability and Force ellipsoids in Robotics

6.1 Manipulability ellipsoid: speed locus

[107: elipm1EN] Manipulability of robotic arm (1): modelling, jacobian, singular value decomposition *** 16:33

Materiales: [Cód.: ElipsRobot1ENG.mlx] [PDF] [YouTube ▶]

This video introduces the concept of manipulability analysis, the discussion of which will continue in the video [108] (manipulability ellipsoid) and [109] (more examples and animations).

Specifically, the basic idea of the analysis is based on the robot Jacobian (the matrix that relates joint velocities to end-effector velocities) and its singular value decomposition $J = USV^T$, which gives rise to the “principal manoeuvres”. A theoretical analysis discussing the relationship between singular value decomposition (SVD) and ellipsoid geometry is further discussed in video [9].

This video makes a brief theoretical review of the Jacobian concept, models a robot with three articulated elements (planar one, no gravity), obtains said Jacobian, and analyzes its SVD. The case study continues in the videos referred to at the beginning of the description.

[108: elipm2EN] Manipulability ellipsoid of robot arm (2): theory, example, polyhedron approach **** 17:52

Materiales: [Cód.: ElipsRobot1ENG.mlx] [PDF] [YouTube ▶]

This video continues the manipulability discussion started in video [107], which is summarized here in the first two and a half minutes.

In this video emphasis is placed on defining what is called the *manipulability ellipsoid*: the locus of end-effector velocities when joint velocities are on the unit sphere (after some scaling/normalizing).


After a theoretical review, said ellipsoid is represented graphically and the relationship of the principal maneuvers of the SVD of the scaled Jacobian with said ellipsoid is illustrated.

The final part of the video superimposes the manipulability ellipsoid with the locus of end-effector velocities when joint velocities are in the *cube* of vertices ± 1 . The ellipsoid approximates it reasonably, and allows the SVD principal manoeuvre interpretation that the polyhedron approach does not easily allow. By contrast, the polyhedron is an exact representation of practical achievable speeds. Everything has its advantages and disadvantages.

Examples in other poses and animations will be illustrated in the video [109], a continuation of this one.

[109: elipm3EN]

Manipulability ellipsoid of robot (3): further examples and animations
Materiales: [[CÓD.: animrobot.m](#)] [[PDF](#)]


***  12:23
[\[YouTube ▶\]](#)

This video graphically illustrates the principal maneuvers and manipulability ellipsoids discussed in the video [108] in different poses. Starting at minute [06:40], we also represent the principal maneuvers with an animation, to better understand their meaning.

6.2 Force ellipsoid

[110: elipf1EN]

Jacobian: matrix gear ratio, effect on force/torque multiplication
Materiales: [ElipsRobot2Force.mlx]


****  20:22
[\[YouTube ▶\]](#)

This video discusses the role of the Jacobian matrix J in force/torque transmission in a robot arm (or in a generic mechanism, of course). In previous videos, chain rule justified that $\dot{r} = J\dot{q}$, being \dot{r} end effector speed and \dot{q} the joint speeds. Here, we discuss that $\tau = J^T f$ is the joint torque needed to compensate end-effector force f in order to keep the robot arm motionless.

We do it in two ways: a first one, sort of informal, via an energy (power) conservation argumentation, extracting intuition from a pinion-gear thought experiment; the second one is a more accurate account of what happens, deriving the same expression $\tau = J^T f$ through the Euler-Lagrange equations of motion.

[111: elipf2EN]

Force ellipsoid in a robot (1): definition and basic examples
Materiales: [ElipsRobot2Force.mlx]

***  11:53
[\[YouTube ▶\]](#)


This video defines the *force ellipsoid* of a robotic manipulator. Said ellipsoid is the ellipsoid of end-effector forces that could be compensated to keep the manipulator immobile if the joint torques lie on the unit sphere.

The basic theoretical formulas are justified in the video [110], which you should watch before this one. Indeed, the fundamental idea is that the torque vector τ that compensates a force f in the end effector is $\tau = J^T f$, so $\tau^T \tau = f^T \cdot (J \cdot J^T) \cdot f \leq 1$ defines an ellipsoid (said “force ellipsoid”). The video introduces the possible scaling and draws the ellipsoid in different poses.

Theoretical SVD analysis continues in video [112], and the relationship between manipulability ellipsoids (velocities) and forces is discussed in video [113].

[112: elipf3EN]

Force ellipsoid in a robot (2): principal maneuvers, Jacobian singular value decomposition
Materiales: [ElipsRobot2Force.mlx]

****  14:41
[\[YouTube ▶\]](#)


This video discusses the relationship of the *force ellipsoid* of a robotic manipulator (defined in the video [??]), with the singular value decomposition of the Jacobian (with an appropriate scaling).

The main maneuvers are obtained in the space of end-effector forces and normalized joint pairs.

The development is parallel, practically identical, to that of the principal maneuvers in the speed space discussed in the video [108]. In fact, it is so similar that (with unit scalings) the principal maneuvers are coincident (with inverse gain): the details of this important property are discussed in the video [113], which delves into the relationship between manipulability ellipsoids (velocities) and forces.

[113: elipf4EN]

Duality of force/manipulability ellipsoids in a robot (inverse behaviour)

***  14:43

Materials: [ElipsRobot2Force.mlx]

[YouTube ▶]

This video concludes the analysis of the force and manipulability ellipsoids (object of videos [107], [111] and successive ones) of a robot. In this last video, we explore their duality: the main speed and force maneuvers have “inverse” amplitudes: if a maneuver multiplies angular velocity by 7 to obtain end-effector linear velocity, then the same maneuver divides joint torque by 7 to give end-effector force.

For this to be correct, operations must be performed on NON-scaled compatible physical units (m/s, rad/s, N, Nm).

This generalizes to “matrix version” the basic idea of many mechanisms (lever, gear): multiplying speeds by 7 in a lever divides forces by 7, for instance.


The orthogonality of the main SVD maneuvers means that the Jacobian of a robot can be interpreted as an XY table, with two different gear ratios given by the singular values.

Chapter 7

Statistics and data mining

7.1 Introduction to probability and recursive Bayes filter: ‘roaring tiger’ case study

This case study will present a simple-to-understand ‘zookeeper’ task that will motivate and introduce a series of basic concepts in probability and statistics and end with a recursive Bayes filter (almost!, state transitions are not present).

[114: tiger1EN]	Hidden tiger (1): Conditional, prior and joint probability tables; problem description	*  22:26
Materiales: [CÓD.: Tiger1ENG.mlx] [PDF]		[YouTube ►]

* [ENLACE A SPANISH VERSION](#)


This video is the first of a case study of a “hidden tiger” problem that will allow us to review and improve the understanding of many concepts in Statistics and probability.

In this first video, the problem is outlined: we have a tiger that is in one of two cages (left or right), but whose roars are sometimes heard by a zookeeper from the wrong side because the tiger roars towards a passageway connecting the doors. The final goal of the problem is to find out where the tiger is (to “save the zookeeper’s life” by leaving through the other door)... but for that, you have to understand many previous concepts... starting with the “probability tables” we are introducing here.

Here we explain conditional probability tables, of “observations” (hearing roars left or right) conditioned on ‘location’ of the tiger (tiger left or right).

After that, we define the concept of “a priori” probability that the tiger is on one side or the other *before* hearing the first roar, and with this we calculate the joint probability $p(m \wedge t) = p(m|t) \cdot p(t)$ of the four possible “atomic” combinations of “roar hearing” and “tiger location”.

The video discusses the interpretation and relationships between these probability tables. Specifically, although everything can be calculated from the “joint” table, a continuation of this one), it turns out that dividing the problem into subproblems is conceptually simpler with the concept of conditional probability (we’ll delve further into these issues in the video [115]).

[115: tiger2EN]	Hidden tiger (2): from joint to marginal and conditional; direct/inverse conditional tables	*  20:34
Materiales: [CÓD.: Tiger1ENG.mlx] [PDF]		[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video is the second one of the “hidden tiger” case study, which began in the video [114] and is briefly reviewed here.

In the aforementioned video the problem was posed, and a probabilistic model was defined. It was based on a probability of a left/right tiger after closing a certain gate and a conditional probability of which side (left/right) it is heard roaring based on (conditioned to) which side the tiger remained trapped in. With this, the *joint* probability table was obtained.

In this video, the following probability tables are obtained from the joint probability one:


- Marginal of ‘tiger location’ (recovers the starting prior probability)
- Marginal probabilities of ‘side where roar is heard’
- Conditional from tiger location (condition) to roar hearing side (conditioned variable); We already knew this one, we are just tracing back what we did earlier on.
- Conditional from roar listening side (condition) to tiger location (conditioned variable)

The first marginal and conditional have the meaning discussed in the first video; The second marginal tells the “accumulated frequency” of left or right roars if the experiment were repeated many times.

The second conditional table has a different meaning; roughly, the arrow of causality goes “backwards”: given an observed effect (roar), the *posterior* probability of its two possible causes (tiger on the left or tiger on the right) is inferred. This idea is the basis of *Bayesian statistical inference*. The following video, [116], will further detail this last idea.

[116: tiger3EN] Hidden tiger (3): marginal, conditional, Bayes rule, graphical interpretation

Materiales: [[CÓD.: BayesNonRecursiveP1ENG.mlx](#)] [[PDF](#)]

**  21:20

[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video continues the discussion about marginal and conditional probabilities that we started in the video [115]. A quick recap of such video is made in the first minutes of the present one.

In particular, the relationship between marginals and conditionals will be graphically represented as a function of the a priori probability... If the a priori probability of the condition is 1, marginal and conditional coincide; if it is not 1, then the marginal is a kind of “interpolation” of the conditional.

The second part of the video discusses the Bayes formula to obtain the “backward” conditional probability (cause given effect) from the joint table or, better, as it is usually written, from the “forward” conditional (effect given cause) and the a priori probability of the cause.

The backward conditional probability of a cause given a certain effect is also called “a posteriori” if it is evaluated after actually observing said effect in a practical situation.

Various a priori and a posteriori probability plots are plotted for different conditional tables representing different levels of “precision” in the observation of tiger roaring noise.

Extensions to two roaring events are discussed in the continuation video [117], and to any number of roarings in [118].

[117: tiger4EN] Hidden tiger (4): conditional and Bayes rule for TWO roars

Materiales: [[CÓD.: BayesNonRecursiveENGPart2.mlx](#)] [[PDF](#)]

**  13:44

[YouTube ►]

* [ENLACE A SPANISH VERSION](#)

This video is part of the “tiger” case study that began in the video [114], which continues the inference using Bayes formula discussed in the video [116], the one immediately preceding this one in the recommended sequence.

This video addresses the problem of estimating the posterior probability of locating a “hidden tiger” after hearing TWO roaring events (without allowing the tiger to switch cages between roars), while in previous videos it was only assumed that a single roar could be heard.

The problem will be addressed by building the conditional probability table of the four possible observations (LL, LR, RL, RR) conditional on the position at the left or right cage of the hidden tiger.

To conform said conditional table, we will assume “conditional independence” which will allow us to multiply conditional probabilities; Its meaning is briefly discussed in the context of the tiger problem (if the condition that the tiger does not change sides is met, then successive roars are asumed independent), but please refer to the videos [123] and [124] for a detailed analysis of the concept and further examples.

Once this new conditional table has been formed, the application of the Bayes formula is identical to previous cases. The video [118] will discuss the general case of an arbitrary number of roars, and the video [119] will propose a recursive Bayes formula to solve the same problem.

[118: tiger5EN]

Hidden tiger (5): Bayes formula for any number of roars
(non-recursive)

Materiales: [[CÓD.:](#) BayesNonRecursiveENGPart2.mlx] [[PDF](#)]

*** 16:18

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video is part of the “hidden tiger” case study in the collection. Here we generalize the Bayes formula that was seen for the case of one roar in the video [116] and for two roars in [117].

Specifically, it is generalized to an arbitrary number of roars, say N . Two possible conditional probability tables are introduced: an expanded one where the order in which the roars are received does matter, and another compressed one where only the number of roars heard through each of the doors matters; The second table has a “binomial” distribution.

Applying the Bayes formula in both cases produces an identical result. The video ends with a numerical example in which it is shown that if we wait for, say, 15 roaring events, then we are pretty sure on which side the tiger lies.

[119: tiger6brEN]

Hidden tiger (6): recursive Bayes formula for any number of
roars (proof)

Materiales: [[CÓD.:](#) BayesRecursiveENGv2.mlx] [[PDF](#)]

**** 15:55

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video reviews the theory behind the recursive Bayes formula for the N hidden tiger roar problem. The NON-recursive formula was discussed in the video [118]. Here we verify that applying N times the Bayes formula with the ‘1 roar conditional table’ gives the same final posterior probability as applying the ‘ N roar table’ just one single time. In both cases, ‘conditional independence’ must be assumed for the results to be correct.

Ideas similar to the ones here are behind some ‘recursive Bayes filters’.

[120: tiger7rbsEN]

Hidden tiger (7): recursive Bayes formula (simulation, Matlab)

17:40

*** 

Materiales:

[[Cód.:](#) BayesRecursiveENGv2.mlx] [[PDF](#)]

[[YouTube](#) ►]

* [ENLACE A SPANISH VERSION](#)

This video presents a simulation of the recursive Bayes filter in the hidden tiger problem that we are analyzing in several previous videos. Specifically, the theory of the recursive filter was discussed in the video [119] (well, only adapted to this particular case study). Here, the code to carry it out is detailed and its results are discussed, obviously coinciding with those of the non-recursive formula in the video [118].


Several simulations are made with different “signal-to-noise” ratios for the observations (percentage of roars heard from the wrong side). The noisier the observations, the more roars are needed to be sure where the tiger is actually hiding. In all simulations, the tiger always remains on the same side, as it is required to be so according to the assumptions we posed to carry out the statistical inference.

This is almost a full-fledged “Bayes filter”, albeit the said full Bayes filter can incorporate the possibility of the tiger switching sides from time to time following a “Markov process” dynamics with an associate transition probability table. These possible expansions are not considered here, for brevity.

7.2 Other statistics concepts

[121: cfrc1EN]

2D Gaussian distribution: confidence rectangles and ellipses (Matlab example)

**  18:51

Materiales:

[[Cód.:](#) IntConfVSchi2ENG.mlx] [[PDF](#)]

[[YouTube](#) ►]

* [ENLACE A SPANISH VERSION](#)


This video illustrates the computation of the ‘maximum likelihood per unit area’ region for a 2D standard normal distribution (that is, a multivariate Gaussian probability distribution with identity covariance matrix), using the χ^2 formula. Such a region is, indeed, a ‘confidence circle’ given the circular contours of the density function.

We also compute ‘marginal’ confidence regions by studying each variable separately (e.g., $\pm 1.96\sigma$ would give the 95% confidence interval). Multiplying the probabilities of an interval would calculate the probabilities of a confidence ‘rectangle’ (square in this unit variance case). The final part of the video discusses the confidence ellipse that would come out with a covariance `diag([1 4])`, and the resulting rectangles. The calculations if the variance-covariance matrix were not diagonal would not be valid; this is discussed in more detail in the video [122], a continuation of this one.

[122: cfrcwrongEN]

2D Gaussian distribution: remark on confidence rectangles and ellipses with correlated variables (Matlab example)

08:29

*** 

Materiales:

[[Cód.:](#) IntConfVSchi2ENG.mlx] [[PDF](#)]

[[YouTube](#) ►]


* [ENLACE A SPANISH VERSION](#)

This video reviews the main ideas of the video [121], and raises the question of whether or not the confidence region computations in that video would still be correct with correlated Gaussian variables (non-diagonal covariance matrix).

Briefly, it is proposed that the ellipsoid will be correctly computed (although it will not be aligned with the axes, the relationship of the geometry of said ellipsoid with the eigenvalues and eigenvectors of the variance-covariance matrix is discussed). Also the marginal

confidence intervals will be correct (the marginal confidence intervals of a 2D normal are a 1D normal).

However, the 'rectangles' that were obtained by multiplying the probabilities of the confidence intervals would NOT give a correct probability result as marginals were not independent variables. To do this right, the rectangles should be aligned with the axes of the ellipse (principal components). This fact of transforming the variables of a multidimensional normal (rotate) so that the covariance matrix is diagonal (confidence ellipsoids aligned with the axes) is what motivates what is called principal component analysis in multivariate statistics.


[123: condin1EN] Conditional independence (I): definitions and basic examples ***  18:42
Materiales: [conditionalIndepEnglish.pdf] [YouTube ►]

*ENLACE A SPANISH VERSION

The ideas of statistical independence (videos [??], [??]) can be refined to discuss about the independence of conditional probability distributions (video [??]). With this, we can think of random variables that are “conditionally independent given a certain value of another third variable”. Although it may seem like a useless refinement of interest to very few people, this is not the case: there are very relevant “common sense” ideas behind it, and it is at the root of the ideas of “state” in Physics and Control Theory. and “Bayesian networks” in Artificial Intelligence.

This video discusses the theoretical definition and some “intuitive” examples to understand the concept (car brand versus fines, height versus academic qualifications, etc.). A second video [??], continuation of this one, discusses its relationship with dynamic systems models used in control and Bayesian networks, also used in other areas of computer science and Artificial Intelligence.

The idea is also discussed in the ‘hidden tiger’ case study that introduces some probability concepts, see video [117]. Further examples (well, actually particularized to the case of conditional correlation or lack thereof) are discussed in the video [128], recommending its viewing to consolidate the concepts.

[124: condin2EN] Conditional Independence (II): control-relevant examples (Markov hypothesis), Bayesian Networks ***  15:27
Materiales: [conditionalIndepEnglish.pdf] [YouTube ►]

*ENLACE A SPANISH VERSION

The video [123] defined the concept of random variables which were “conditionally independent given some value of another third variable”. Although it may seem like a useless refinement of interest to very few people, this is not the case: there are very relevant ideas behind it, and it is the cornerstone of the idea of “state”, fundamental in multivariable control theory, and “Bayesian networks” in Artificial Intelligence.


This video complements the “intuitive” examples in the above-mentioned video with more abstract examples on measurement and process noise, the concept of state, Markov hypothesis, and the associated Bayesian network diagrams. It is recommended to watch the video [??] for more discussions about the importance of the concept of state and Markov in the modeling of physical systems.

The video ends presenting some conclusions about the importance of the concept of conditional independence in scientific methods in general.

Further numerical examples (well, actually particularized to the case of conditional correlation or lack thereof) are discussed in the video [128], recommending its viewing to consolidate the concepts. The idea is also discussed in the ‘hidden tiger’ case study that introduces some probability concepts, see video [117].

[125: vcinv1EN]

Best linear prediction: from covariance to linear model with additive noise and vice-versa (theory)

***  09:20

Materiales:

[[CÓD.:](#) BestpredlinearversusmodelEnglish.mlx] [[PDF](#)]

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses how two related statistical computations are related:


First, the video discusses how to obtain a linear model with additive independent noise from a given covariance matrix of a pair of random variables (a, b) . Indeed, using the best-linear prediction formula, the prediction can be expressed asserting that, if we believe linearity, the conditional distribution of b given a may be written as $b = \theta a + \epsilon$, being $\theta = \Sigma_{ba} \Sigma_{aa}^{-1}$ and ϵ having variance $\Sigma_{bb} - \Sigma_{ba} \Sigma_{aa}^{-1} \Sigma_{ba}^T$. This does not mean that the actual underlying physics is linear but, well, it explains the observed variance.

Second, the inverse process is illustrated, i.e., how to obtain the associated covariance matrix from a linear model with additive noise $b = \theta a + \epsilon$.

Numerical examples are left for videos [126] and [127].

[126: vcinv1bEN]

Best linear prediction, inverse models in statistical sense: example (I)

***  15:46

Materiales:

[[CÓD.:](#) BestpredlinearversusmodelEnglish.mlx] [[PDF](#)]

[YouTube ▶]

A first numerical example on the ideas of video [??] is presented. Given a covariance matrix between a pair of random variables (a, b) , a linear model with additive noise $b = \theta a + \epsilon$ is built, as well as another one $a = \eta b + \epsilon'$.


Note that we might think that one model would be the algebraic inverse of the other, but it is not true, i.e., $\eta \neq \theta^{-1}$. A graphical representation of the idea is discussed.

Some information is lost when contaminating something with noise so we cannot revert back to where we started with... Unless you train a "difussion neural network" trying to do exactly that: recover a picture from a random seed... but we are NOT discussing that in the video!

A second example delving on the same idea appears in video [127].

[127: vcinv2EN]

Best linear prediction, inverse models in statistical sense: example (II)

***  10:10

Materiales:

[[CÓD.:](#) bestlinearpredversusmodelInverseEnglish.mlx] [[PDF](#)]


[YouTube ▶]

*ENLACE A SPANISH VERSION

This video illustrates the ideas in videos [125] and [126] with an additional short example. Again, it illustrates that inverse in statistical sense (minimum-variance predictor) is not the same as inverse in algebraic (deterministic) sense; they coincide when signal-to-noise ratio tends to infinitely good (noise variance tends to zero).

[128: condnocoEN]

Conditional Independence (III): conditionally UNcorrelated random variables (only multivariate normal distribution)

**** 

17:13

Materiales:

[[CÓD.:](#) conditionalnocorrelaexample1English.mlx] [[PDF](#)]

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video complements the discussion on conditional independence of the video [123], particularizing it to the case where only *linear* predictions are desired and, therefore, only the variance-covariance matrix must be analyzed.

Therefore, the absence of covariance (that is, the non-correlation) between the prediction errors of two random variables given a third one is what will define the conditional non-correlation... well, sort of..

NOTE: you should take “informally” the concepts released here, since they would only be valid if we are sure that the random variables have been generated with a multidimensional normal distribution. Otherwise, the “conditional” distribution will not be the one given by the best linear prediction formulae, but some “informal” conclusions can be drawn regarding capability of doing linear predictions. And, in fact, in the normal case, conditional uncorrelation would also mean conditional independence, so we are not discussing, in fact, any new concept from said independence if we wish to be more rigorous.

The point is that this conditional non-correlation allows making Bayesian networks that “chain” linear models with additive noise, as seen in the numerical examples.

[129: pcaissvdEN] Comparing PCA (statistics toolbox) with SVD (built-in, Matlab): identical results

Materiales: [[CÓD.:](#) pcavssvdEnglish.mlx] [[PDF](#)]

****** **03:59**

[YouTube ►]

*ENLACE A SPANISH VERSION

This very short video verifies that singular value decomposition ([svd](#)), which does not require any particular toolbox installed as it is a basic built-in Matlab command, produces the same result as the output of the Matlab Statistics and ML toolbox command [pca](#) for principal component analysis.

[130: tls51EN] Total Least Squares with 5 variables: Matlab example

Materiales: [[CÓD.:](#) TLS3x2p4English.mlx] [[PDF](#)]

******* **18:30**

[YouTube ►]

*ENLACE A SPANISH VERSION

This video presents an example of Total Least Squares (TLS) with five interrelated variables. The theory and simpler examples are discussed elsewhere.

When the data is generated, there is a model $y_{1 \times 2} = x_{1 \times 3} \Theta_{3 \times 2}$, that is, of two equations (we predict two variables from three other ones). This is not known to TLS, obviously, since it has to figure it out.

TLS is convenient because all the data, both X and Y , are corrupted with independent noise of certain standard deviation; indeed, the estimation of ordinary least squares with pseudoinverse is biased. Scaling so that all variables have noise 1, the two smallest singular values of the matrix $[Y \ X] = USV^T$ are, in effect, approximately 1 as predicted by theory. The last two columns of V constitute the implicit model.


To check that it is correct, the scaling is undone and $Y = X\Theta_{TLS}$ is derived to check that it matches the one used when generating the data (which, in a real application, would be unknown, evidently).

A last rather trivial procedure addresses how to modify the code if each sample of the data is a column instead of a row. The modifications are merely cosmetic, transposing things, and the result is obviously identical.

The video [131], a continuation of this one, asks what happens if I make a mistake in the scaling: unfortunately, I also get a skewed model in that case.

[131: tls52EN]

Total Least Squares with 5 variables: Matlab example 2, wrong scaling



11:10

Materiales:

[[CÓD.: TLS3x2p4English.mlx](#)] [[PDF](#)]

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video is a continuation of the video [130] where a total least squares analysis was made of the implicit equations that related five (random) variables.

The issue discussed here deals with analyzing what happens when the scaling, made prior to SVD, is incorrect. Unfortunately, the result is a biased model. In fact, the bias is even greater than in the classical least squares model that is often used to justify TLS in many classrooms (even mine)... if the scaling is poorly designed, the cure may be worse than the disease. This may be a problem in PCA/TLS applications.

Since the model is multivariable, and implicit (equation between five variables $\Theta[Y; X] = 0$, the command `subspace` is used to determine the angle between the subspaces in which the variables $[Y; X]$ can be there if they verify the model's equations (the null space of Θ), so the magnitude of the "bias" is reduced to a single number.

Classification (problem statement)

[132: clasifintr1EN]

Model fitting for classification (1): problem statement (deterministic version)

Materiales:

[ModelsClasif1ENG.pdf]

*ENLACE A SPANISH VERSION


This video poses the problem of fitting some parameters θ in a model $f(x, \theta)$ to fit pairs of training data samples (x_i, y_i) with $y_i \in \{0, 1\}$, that is, binary classification (supervised). The problem can be deterministic (dog/not-dog) or random (probability of having a certain genetic mutation when some marker appears in a blood test). This video discusses the deterministic problem, presenting the issues that arise when "perfect" classification is possible, and the cost functions for "imperfect" classification problems where false positives or false negatives are inevitable.

The probabilistic version of the statement of this type of problems is addressed in the video [133].

[133: clasifintr2EN]

Model fitting for classification (2): probabilistic version of problem statement

**



15:41

Materiales:

[ModelsClasif1ENG.pdf]

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video presents the "probabilistic" version of the problem of fitting θ in a model $f(x, \theta)$ to fit pairs of training data (x_i, y_i) with $y_i \in \{0, 1\}$, that is, binary classification. The "deterministic" version is discussed in the video [132], previous to this one, which you are advised to watch.

The objective here is for the model to estimate the probability of having a certain output at 1 given x ... the simplest version is to maximize the probability of the samples (maximum likelihood estimation) by adjusting θ . There may be more formal Bayesian approaches, out of the scope of this very short introduction to the topic.

[134: clasifNoLSEN]

Model fitting for classification (3): are Least Squares a sensible choice?



16:51

Materiales:

[ModelsClasif1ENG.pdf]

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video discusses the applicability (well, rather the NON applicability) of least squares in classification; The video is a continuation of the videos [132] and [133].

Although least squares for fitting a function $f(x, \theta)$ to training data $\{-1, +1\}$ might be reasonable as a first option to start with the problem, it suffers from several drawbacks:


- The cost is ‘symmetric’, but some problems fare better with an asymmetric goal,
- The only thing that really matters in the fit is the sign of $f(x, \theta)$ not its value,
- Probability distributions (or their logarithms) with binary results (Bernouilly, or binomial if we count repetitions) are not proportional to ‘squared error’ as actually is the logarithm of the normal distribution in least squares for fitting continuous data (under certain assumptions such as additive normal measurement noise).

Therefore, a ‘naive’ solution of applying ‘linear regression’ or, in general, optimizing a least-squares index for binary data, may not have the desired performance or may not have a formally adequate statistical interpretation.

7.3 Stochastic processes (Gaussian processes)

[135: estoc1EN] Stochastic processes (random functions) in engineering:
motivation, definitions, examples

Materiales: [estocas1ENG.pdf]

**  15:40

[YouTube ►]

*ENLACE A SPANISH VERSION

This video poses, in a very informal way, the concept of “stochastic process”, used in statistical analysis, as a formalization of what, in engineering terms, would be understood as a “signal corrupted by noise”.


Its definition as a random function is discussed, and examples in signal processing, audio-video, and vibrations are outlined. The overall goals of using stochastic processes in applications are also hinted.

At first glance, signal processing applications might be associated to only “functions of time” (time series, in statistical jargon). However, the ideas generalize to stochastic processes in “spatial” coordinates (say, colored noise in a photograph) or in both time+space (video signals).

Stochastic processes are also used in optimization, giving rise to the so-called *Bayesian optimization* techniques, see [156].

[136: gpsambEN] Sampling a Gaussian process (realizations), Matlab example

Materiales: [CÓD.: GPSSampletests2ENG.mlx] [PDF]

***  13:57

[YouTube ►]

*ENLACE A SPANISH VERSION

This video discusses how to compute ‘realizations’ (random functions $f(x)$) of a Gaussian stochastic process with given mean function $\bar{f}(x)$ and covariance kernel $k(x_1, x_2)$. As the process is ‘continuous time’, samples of it will only be generated in a finite set of test points.

The video discusses the code to generate the covariance matrix in said set of test points, the meaning of the chosen stationary covariance kernel (quadratic exponential, but it can be any other), the band-type structure of the covariance matrix and the use of `mvnrnd` to generate the process realizations. The internal detail of the `mvnrnd` command requires carrying out diagonalization or Cholesky decomposition of the covariance matrix, but those implementation details are out of the scope of this video.

To better understand this type of processes, the correlation abscissa distance parameter is changed in the covariance function, also, in an additional example, white measurement noise is added (increasing the diagonal of the covariance), etc.

The video [137], a continuation of this one, will discuss the same problem (generating realizations) when observations are available at some ‘measurement’ points of the value of the process.

[137: gpsambpoEN] Sampling a Gaussian process with some observations (sampling the posterior), Matlab example

14:55

Material: [[CÓD.: GPSampletests2ENG.mlx](#)] [[PDF](#)]

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video discusses how to compute realizations of a stochastic process when you have observations of its value at a set of ‘measurement’ points. For example, the stochastic process may be the speed of each car at different points on a highway, and you want to ‘interpolate’ for a specific car the most likely trajectories knowing that, in two positions where there was a radar, its recorded speeds were 110 and 80 km/h.

The first part of the video reviews previous concepts and best linear prediction, and discusses the code to generate a posterior mean and covariance matrix on a set of test points.

The second part of the video plots the confidence intervals of that posterior and generates various realizations (random functions) to illustrate the concept, using `mvnrnd`. The development is quite parallel to the video [136], whose visualization is recommended prior to this one.

The final part of the video presents simulations of the results of changing the correlation distance in the covariance kernel.

[138: gpkh2EN] Karhunen-Loeve (PCA) components of a Gaussian Process: Matlab example (1)

Material: [[CÓD.: GPpcaHKtestENG.zip](#)] [[PDF](#)]

**** 18:47

[YouTube ▶]

* [ENLACE A SPANISH VERSION](#)

This video presents how to obtain the principal components (Karhunen-Loeve eigenfunctions in the continuous limit case) of a stochastic Gaussian process, with a ‘quadratic exponential’ covariance kernel (said kernel choice was arbitrary). For simplicity and for graphical representations, a ‘discrete’ set of 241 test points will be considered in order to analyze the structure of the covariance $K_{241 \times 241}$.

The principal components, obtained from the diagonalization $K = VDV^T$, are one of the many ‘matrix square roots’ $K = QQ^T$ that can be devised. Specifically, they are an ‘orthogonal’ square root in the sense that the columns of $Q = V\sqrt{D}$ are orthogonal to each other (uncorrelated components).

The video discusses various conceptual issues related to matrix square roots and principal components, and ends by graphically representing some of them.


The video [139], a continuation of this one, will animate the reconstruction of a realization from standard normal ‘latent’ principal components, and will also do so with a process for which observations are available at certain points.

As the matrix square root Q of the covariance matrix is not unique, there are other causal, anticausal and bilateral representations of the Gaussian process, which will be analysed in videos [140] and [141].

[139: gpkh2pEN]

16:44

Karhunen-Loeve (PCA) components of a Gaussian Process:
Matlab example (2), animation and posterior



Materiales: [[CÓD.:](#) GPpcaHKtestENG.zip] [[PDF](#)]


[\[YouTube ▶\]](#)

*ENLACE A SPANISH VERSION

This video is a continuation of the video [138], where the meaning of a matrix square root of the covariance $K = QQ^T$ and its continuous limit (Karhunen-Loève eigenfunctions) were discussed. In this video, realizations are generated by adding component by component multiplied by a standard normal variable, in a Matlab animation. The second part of the video computes the posterior assuming that the process value has been observed at some points (see video [137] for details). Then, diagonalizing the covariance, the shape of the principal components (scaled eigenvectors) is analyzed and realizations of the posterior are constructed from them.

[140: gpcholEN]

Gaussian process: Cholesky factor of covariance, spectral factor
(Matlab example)



19:34

Materiales: [[CÓD.:](#) GPcholENG.zip] [[PDF](#)]

[\[YouTube ▶\]](#)

This video presents the interpretation of the Cholesky factorization (lower triangular) of the covariance matrix of a Gaussian stochastic process (in discrete time, since we approximate a continuous process through a finite number of test abscissa points). An initial part reviews the basic ideas from the video [138], about the square roots of a covariance matrix and the principal components. Then, it focuses on the main objective, which is to interpret the meaning of the triangular structure of the Cholesky factor... it is given a 'causal' interpretation: the latent variable ' j ' has an effect on the outputs in the $j + 1, \dots, 241$ positions. Thus, a realization of the process 'from left to right' is constructed.


It is observed that the columns of Q converge to a single sequence, which moves downwards (when the steady state is reached and the finite window effects have disappeared). The video justifies why this gives rise to a *convolution* formula to calculate the effect of the latent variables on the outputs; The convolution kernel can be interpreted as the impulse response of a time-invariant linear system that is called 'causal spectral factor' in literature.

As square roots are not unique, there are other representations (anticausal, bilateral) of interest, which will be discussed in the video [141], a continuation of this one.

[141: gpanticaEN]

13:48

Gaussian process: anticausal and bilateral representations
(Matlab example)



Materiales: [[CÓD.:](#) GPcholENG.zip] [[PDF](#)]

[\[YouTube ▶\]](#)

This video presents the 'anticausal' and 'bilateral' (mixture of causal and anticausal) representations of a Gaussian process, in a practical way via a Matlab example. The representations come from using a particular square root of the covariance matrix, since such matrix square root is not unique. The first three minutes review the orthogonal square root that gives rise to the Karhunen-Loeve principal components (video [138]) and the lower triangular Cholesky factorization that gives rise to the causal representation (video [140]).

By permuting rows and columns of the Cholesky decomposition, equivalent to a rearrangement of the random variables, we obtain an 'anti-causal' representation where each latent variable influences the output of the process in the same place and on the previous ones (past), following the way we decided to order the abscissas from the Gaussian process

(which we could interpret as 'time'). This gives rise to an animation where a realization of the Gaussian process is built 'from right to left'.

The second part of the video discusses the symmetric square root $Q = V \cdot \sqrt{D} \cdot V^T$. In that case $K = QQ^T = Q^2$. The result is a 'mixture' of causal and anticausal (symmetric) action of the latent variables on the output, except effects of initial conditions or finite window; this will give rise to what would be a *bilateral convolution kernel*. An intuitive interpretation is that each row of Q (well, just the 'stationary' ones) is like a 'hammer blow' that deforms symmetrically on either side of the impact point, and that the intensity of the impact is the standard normally distributed latent variable.

7.3.1 Multi-output GP

[142: mimoGP1EN] (1/5) Multi-Output Gaussian Processes: Motivation
15:34

*** 

Materiales: [[CÓD.:](#) GPmultioutputpart1.mlx] [[PDF](#)]

[YouTube ►]


This video introduces and motivates multi-output stochastic processes, based on two basic examples:

- Temperature and humidity readings at weather stations in different positions, from which you want to 'interpolate'
- The statistical relationship between the position and velocity of a mass subject to random accelerations.

In these examples, we will have a variable, say $y_1(x)$, and another one, $y_2(x)$, so that apart from the 'marginal autocovariances', say, $E[y_1(x_a)y_1(x_b)]$, $E[y_2(x_a)y_2(x_b)]$ (we assume zero mean for notational simplicity), we could have a covariance between the signals $E[y_1(x_a)y_2(x_b)]$ that would allow us to interpolate humidity with temperature data or estimate speed with position data, for example.

How are these covariances generated? Well, the theory of stochastic differential equations and filtering justifies the second case (position, velocity)... The former case (temperature, humidity) could be more 'descriptive' or assuming certain components (underlying latent variables) to be present in the data-generating mechanism. The 'intuitive' details of this case are discussed in the video [143], a continuation of this one. Stochastic differential equations are a topic of greater complexity (but of greater theoretical interest, too), discussed in other materials.

[143: mimogp2EN] (2/5) Multi-Output Gaussian Processes: representation; code
21:58 for realizations

**** 

Materiales: [[CÓD.:](#) GPmultioutputpart1.mlx] [[PDF](#)]

[YouTube ►]

This video, a continuation of the video [142], discusses the representation of multi-output stochastic processes in two ways:


- as a process that, indeed, has an auto-covariance function $k(x_1, x_2)$ whose result is a matrix of size according to the number of outputs
- as well as a stochastic process with one more input, $k(\xi_1, \xi_2)$, with $\xi = (x, j)$ being an augmented input, a pair of abscissa + index (say, 1 or 2 if we are in a two-output case) indicating which output we are referring to, henceforth with $k(\xi_1, \xi_2)$ being a scalar.

Actually, both representations are equivalent, as one would intuitively expect, and it is just a matter of preference when writing code in which one is used.

Although it is not the only option, generating these multi-output covariances is convenient and simple if one considers that the process is $y = Cu$, with C being a matrix and u a vector of “components”. If we assume that they are independent, they will each have an autocovariance independent of the rest, and $E[u(x_1)u(x_2)^T]$ will be a diagonal matrix.

The last part of the video presents a Matlab example of said processes (2 outputs) and generates realizations of it.

[144: mimogp3pcaEN] (3/5) Multi-output Gaussian Processes: principal component analysis, Karhunen-Loeve eigenfunctions

 09:46

Materiales: [[CÓD.:](#) GPmultioutputpart1.mlx] [[PDF](#)]


[YouTube ▶]

This video presents the PCA (principal component analysis) of the covariance matrix (at a finite set of points) of a 2-output gaussian process whose motivation and covariance structure details were discussed in previous videos [142] and [143]. When the number of points is infinity, in the continuous-input case principal components are Karhunen-Loeve (KL) eigenfunctions. See bottom note for references to one-output KL analysis.

In here, we exploit the special structure of 2-output Gaussian process to better understand such eigenfunctions via suitable plots, seeing that some eigenfunctions are equal in both outputs (strong common component) and only at higher frequencies eigenfunctions appear explaining the difference between the two outputs.

A simpler single-output Gaussian process PCA (KL) is studied in videos [138] and [139].

[145: mimogp4predAEN] (4/5) Multi-output Gaussian Processes: joint prediction

 17:42

Materiales: [[CÓD.:](#) GPmultioutput.mlx] [[PDF](#)]


[YouTube ▶]

This video discusses prediction (Gaussian Process regression) in multi-output stochastic Gaussian processes, see video [142] for motivation and introduction.

Actually, with the representation as a process of a single output with an input domain expanded to $f(x, i)$ with i being a categorical ‘index’ or ‘label’ variable, prediction in an abstract sense is identical to the usual GP prediction, so there is nothing new from a theoretical point of view. Therefore, this video simply has a didactic objective: to understand how by measuring one variable another can be better estimated in close abscissa, complementing the case study of the videos [143] and [144].

If the ‘latent’ factors u in the statistical model had a physical meaning so its estimation seemed of interest in any particular application, that estimation is carried out in video [146], which closes the case study.

[146: mimogp5predUEN] (5/5) Multi-output Gaussian Processes: latent factor estimation

 22:55

Materiales: [[CÓD.:](#) GPmultioutput.mlx] [[PDF](#)]


[YouTube ▶]

This video discusses the prediction of the value of ‘internal’ hidden components or ‘latent factors’ of a 2-output Gaussian process where the statistical model is $y = Cu$, with y of size 2×1 being the observable outputs and u of 3×1 being the latent components, not directly

‘measurable’. Obviously, when the components are the ‘states’ of differential equations what we are doing would be analogous to Kalman filters or RTS/Wiener smoothing, but those techniques are not within the scope of this video.

Basically the covariance between the components and the outputs is $E[u(x_1)y(x_2)^T] = K(x_1, x_2)C^T$ so that using this covariance, the usual prediction formulas can be applied, illustrated in the video by a numerical example based on the case study of the previous videos, in particular [145].

7.3.2 Derivatives of a Gaussian process

[147: gradgpEN] 17:54	Mean and covariance of the partial derivatives (gradient, velocity) of a stochastic process	***** 
Materiales: [estimgradientofGPTheory.pdf]		[YouTube ▶]

* ENLACE A SPANISH VERSION


Certain stochastic processes (“second order” or higher, whatever that means, we’ll skip it for the moment being) are differentiable... for example, the speed of a moving mass subjected to random accelerations can be characterised: that speed is a stochastic process. Position is the integral of velocity, but velocity is the derivative of position.

Formally defining the derivative of a stochastic process requires some rigour (convergence in probability or in mean square) but, well, if we make a “leap of faith” and believe that we understand what the speed mentioned above is, then, from the statistical characteristics of the position in mean and variance (covariance function $k(position1, position2)$) the speed characteristics in terms of mean and variance, and speed-position covariance can be calculated.

In a multivariable stochastic process $f(x)$ (not a time series $f(t)$ as seemed to be implicitly assumed before), the idea is generalized to estimating partial derivatives of said process $f: \mathbb{R}^n \rightarrow \mathbb{R}$ (if they exist), from covariance function $k(x, x')$.


This video proves that the gradient of the covariance $k(x, x')$ is the covariance between the process and its derivative, and that the Hessian (second derivatives) of the covariance gives us the covariance between partial derivatives at different points.

Of course, in time series, operations could also be done with associated stochastic differential equations (for example, Kalman filtering) or with the power spectral density. These issues are not considered here.

[148: gradgpstEN] 13:54	Derivatives of a stochastic process via Jacobian/Hessian of covariance: stationary case	***** 
Materiales: [estimgradientofGPStationary.pdf]		[YouTube ▶]

* ENLACE A SPANISH VERSION

This video applies the formulae in video [147], about the derivatives of a stochastic process, to a stationary case. In that case, covariance $k(x, x')$ only depends on the difference between the points $k(x, x') = \kappa(x' - x)$. Actually, the results are a direct application of the formulae in the referred video but, given that the stationary case is very frequent in applications, it is worth explicitly obtaining the expressions for such a popular use case.

[149: gpvel1EN]	Gaussian Processes: covariance between position and speed example (1)	*****  14:27
Materiales: [CÓD.: estimgradientofGPex.mlx] [PDF]		[YouTube ▶]

* ENLACE A SPANISH VERSION

This video discusses a specific case of statistical relationship between ‘position’ and ‘velocity’ of a 1D Gaussian process. This video briefly states the theoretical formulas for calculating covariance between position and speed and autocovariance of speed. The detailed discussion and proof of these formulas is discussed in the videos [147] and [148].

Then, these formulas are applied and the position/velocity covariance and velocity autocovariance are plotted. The video briefly discusses their ‘physical’ interpretation. Its use for prediction with position and velocity ‘sensors’ is discussed in the video [150].

[150: gpvel2EN]

Gaussian Processes: covariance between position and speed example (2), prediction

Materiales: [[CÓD.:](#) `estimgradientofGPex.mlx`] [[PDF](#)]

**** 19:57

[YouTube ▶]

* ENLACE A SPANISH VERSION

This video uses position/velocity joint covariances from a Gaussian process to make velocity predictions given position measurements in a numerical Matlab example. The second part of this example discusses how to incorporate mixed measurements of position and velocity to make statistical predictions of position and velocity (interpolation). The variances and covariances themselves are calculated with the first and second derivatives of the covariance kernel between two ‘positions’, as detailed in the video [149], which you are advised to watch prior to this one.

7.3.3 The Matérn kernel, intuition and limit cases

[151: maternEN]

Matérn kernel and squared-exponential one: intuition as a filter with repeated real poles

Materiales: [[CÓD.:](#) `MaternKernelTST.mlx`] [[PDF](#)]

**** 14:35

[YouTube ▶]

This video presents the relationship of the ‘Matérn’ type kernel with the result of filtering a continuous-time white noise with a filter (spectral factor) of type $F(s) = \frac{A}{(\tau \cdot s + 1)^m}$. It is verified that the power spectral density and autocovariance (Fourier inverse of the power density) coincide with the Wikipedia formulas based on the Gamma and Bessel functions for said Matérn kernel.

The limiting case (quadratic exponential kernel, Gaussian filter) is also discussed.

The study is done without demonstrations, based only on Matlab plots in time and frequency.

[152: maternr1EN]

Matérn kernel regression (1): basic setup, 1st order filter example, linear interpolation

Materiales: [[CÓD.:](#) `SemipmaternReg.mlx`] [[PDF](#)]

*** 16:40

[YouTube ▶]

This video presents an example of GP regression using a Gaussian process with autocovariance given by a Matérn kernel. The definition of it was discussed in the video [??], of which this is a continuation. To illustrate other concepts making the example richer, a ‘parametric’ component is incorporated saying that the stochastic process could not have zero mean and wishing to estimate that mean from a prior with a normal distribution, more details in other materials.


First, the example setup is presented and, then, a demonstration of interpolation with a spectral factor of order 1 is made. If the correlation distance ρ is small, with distant

samples interpolation (mean) falls exponentially to the estimated “constant mean”. If the correlation distance is large, then interpolation between nearby samples tends toward a piecewise linear interpolation.

Cases of higher order (and its relationship with splines), fractional order and infinite order (exponential-quadratic kernel) are discussed in the video [153], continuation of this one.

[153: maternr2EN] Matérn kernel regression (2): 2nd 3rd and fractional order, splines, Gaussian filter examples

17:25

***** 

Materiales: [[CÓD.](#): SemipmaternReg.mlx] [[PDF](#)]

[YouTube ▶]

This video, a continuation of [152], discusses the Matérn cases of spectral factor of order 2 and 3. When the correlation distance ρ increases, the interpolated mean tends to be a cubic spline (in case of order 2 spectral factor) and a quintic spline (in case of order 3), respectively.

The final part of the video presents an example of fractional order ($n = 1.5$) of the spectral factor (this motivates definitions of ‘fractional derivatives’ in literature, which are not the subject of this material), and an example with a quadratic exponential kernel, which is the Matérn limit when the order of the spectral factor tends to infinity (whose interpolation would be infinitely differentiable), as discussed in video [151].

7.3.4 Stochastic processes in state-space form

[154: ekfteoEN] Extended Kalman filter for nonlinear state estimation (theory)

Materiales: [FiltrKalmanExtEnglish.pdf]

*****  17:28

[YouTube ▶]

*ENLACE A SPANISH VERSION

This video outlines some theoretical considerations that give rise to what we know as *extended Kalman filter*. Such a filter is the generalization of Kalman filter for linear time invariant systems, to nonlinear systems, based on the time-varying linearization around the estimated state trajectory.


A quick recap on the non-stationary Kalman filter equations for linear time invariant systems is carried out in the first four minutes of the video. Next, the nonlinear extension is described.

Basically, if “true” state x_k is close to the estimated state \hat{x}_k , then the linearisation of the state and output equations will allow to estimate the next state or the measurement in both mean and variance. With that, we have the information to set up a standard Kalman filter. The result is a non-stationary filter with model process matrices that change as the state and input change.

The basic drawback is that it is an approximation: with strongly nonlinear systems or inaccurate initial conditions for the state estimate, the filter ceases to be optimal and, in fact, it can even be unstable. However, it is widely used in applications due to its simplicity, efficient computational implementation if Jacobians are available, and reasonably good behaviour in “almost linear” systems.

[155: vanloanEN] Van Loan’s expm formula for variance discretization in linear stochastic ODEs

15:58

***** 

Materiales: [DiscretizRuidoVanLoanEN.pdf]

[YouTube ▶]

*ENLACE A SPANISH VERSION

This material develops the particularization of Van Loan's formulae (paper <https://doi.org/10.1109/TAC.1978.1101>) to numerically compute, with a single matrix exponential, the integral of exponentials of matrices resulting from the solution of variance equations in linear processes subject to white noise.


The presented formula (or its dual transposed one) is used in Matlab's commands `kalmd` and `lqrd` in discrete-time Kalman filter synthesis for LTI systems.

7.4 Bayesian Optimization

7.4.1 Introduction and methodology outline

[156: BOfot1EN]

Bayesian Optimization motivation (1/4): problem statement and model classes

**  15:40

Materials: [BOIntroTheoENG.pdf]

[YouTube ▶]

*ENLACE A SPANISH VERSION


This introductory video raises the need for statistical methods in optimization in some classes of applications (say, experimental optimization, data with additive random noise, etc.) where there is no model with 'perfectly known equations' to optimize (i.e., under the realm of deterministic optimization).

A set of algorithms that use statistical models in optimization are usually grouped under the name *Bayesian optimization* in literature. This video states the problem, raises the concept of 'random function' and the need for a covariance kernel to prove, among other things, continuity of the function and, in general, introduces the idea of stochastic processes and Gaussian processes as the basic tool that most works on Bayesian optimization exploit.

Other descriptions and applications of stochastic processes are covered in the introductory video [135]; forthcoming videos will develop the particular Bayesian optimization problem in greater depth, starting with [157].

[157: BOfot2EN]

Bayesian Optimization motivation (2/4): methodology outline

***  11:20

Materials: [BOIntroTheoENG.pdf]

[YouTube ▶]

*ENLACE A SPANISH VERSION


This video continues with the introduction and motivation to Bayesian Optimization problems which was started in video [156].

In here, we review said video in the first four minutes, and then we discuss the generic goal of Bayesian Optimization: it's actually a type of 'experiment design' trying to choose samples that have a high likelihood of giving me a value close to the optimal (exploitation) or, well, maybe we wish to sample currently likely sub-optimal ones in exchange for them giving a lot of information on where the actual optimum will be for future samples (exploration).

So, BO algorithms end up comprising the following steps:

- Set up a prior statistical model
- Acquire samples (this may be a costly procedure in some relevant application cases)
- Compute a posterior (conditional to the measurements)
- Carry out a statistical analysis on the posterior, to decide next sample, and go to step 2, unless some termination criteria (sample budget, quality of the last sample, lack of progress) is met.

For brevity, a quick outline of the details of the third and fourth steps will be deferred to video [158], and application cases and concluding remarks in video [159] will conclude this brief motivation and presentation of the Bayesian optimization problem. A quick numerical example of the methodology appears in video [160], but you need to watch [158] first to better understand it.

<p>[158: B0mot3EN] 10:49</p>	<p>Bayesian Optimization motivation (3/4): computation of posterior and decision (acquisition) on next sample (outline)</p> <p>Materiales: [BOIntroTheoENG.pdf]</p>	<p>**** </p> <p>[YouTube ▶]</p>
----------------------------------	--	--

* ENLACE A SPANISH VERSION

This video continues with the introduction and motivation to Bayesian Optimization problems which was started in video [156].


In video [157] we concluded that BO algorithms end up comprising the following steps:

- Set up a prior statistical model
- Acquire samples (this may be a costly procedure in some relevant application cases)
- Compute a posterior (conditional to the measurements)
- Carry out a statistical analysis on the posterior to decide next sample, and go to step 2, unless some termination criteria is met.

This video expands on the third step of the methodology, which is carried out in a Gaussian process with the conditional formulae from multivariate normal distributions.

The last step is also outlined... next sample may be selected in order to optimize the expected value (EV), probability of improving (PI), expected improvement (EI), we may be risky and choose a lower confidence bound (we will not likely achieve it, but we would obtain a very good sample if we did), or optimize the information gain by sampling (entropy search). A very brief description of each of these options is presented, just to get a glimpse of the main ideas, but a detailed analysis is the topic of other materials that analyse BO in more depth, see videos [162] and [??].

A quick numerical example of the methodology appears in video [160]. Application cases and concluding remarks in video [159] will conclude this brief presentation of the Bayesian optimization procedures.

<p>[159: B0mot4EN]</p>	<p>Bayesian Optimization motivation (4/4): recommended application domains, remarks</p> <p>Materiales: [BOIntroTheoENG.pdf]</p>	<p>**  19:33</p> <p>[YouTube ▶]</p>
------------------------	--	--

* ENLACE A SPANISH VERSION

After a quick review of prior material, this video discusses the application domains where Bayesian optimization IS recommended (model-free experimental optimization, configuration of long computations in neural networks, fluids, ...), as well as some circumstances in which it is NOT recommended (simple functions, high-dimensional problems). If you did not watch videos [156], [157] y [158], I recommend doing it now to get a broader view.

The video concludes with a couple of remarks on, first, the fact that determinism is a limit case of statistics with uncertainty size tending to zero, so there are quite a few deterministic optimization problems that admit a 'statistical interpretation' (say, Least squares, assuming normally distributed random measurement noise); there are also semi-parametric Bayesian optimization options.


The second remark pinpoints the fact that gradient-based methods may get stuck on local minima; hence, some algorithms (directed random search) use 'rolling the dice' steps to

try to explore and get the global optimum; however, these issues do not change the fact that the underlying problem is deterministic and no uncertainty is assumed to be present in the objective function.

A summary and conclusions reviewing the ideas on this video and the preceding ones in the series that started with video [156] end the video and the 4-video set that motivated and briefly outlined the main features of Bayesian optimization.

[160: boloop1EN] Bayesian Optimization loop: a quick example of the methodology

Materiales: [[CÓD.:](#) BOIntroLOOPSImpleENG.mlx] [[PDF](#)]

***  12:29

[YouTube ▶]


This video presents an example of the Bayesian Optimization methodology which we outlined in videos [157] and [158]. Code issues are only sparingly mentioned, as later videos will delve into the details and add some refinements.

The video focuses on understanding the Bayesian optimization loop (acquire a sample, update posterior, evaluate acquisition function for next sample), via a simple example using ‘lower confidence bound’ as said acquisition function heuristic.

7.4.2 In-depth detail and examples

[161: boinopEN] Bayesian optimization: implicit information about the optimum in a Gaussian process

Materiales: [[CÓD.:](#) BOIntroOptimalSample.mlx] [[PDF](#)]

***  18:53

[YouTube ▶]

This video plots the (approximate) distribution of global minima of the posterior of a function modeled as a Gaussian process of which three samples have been observed.


First, the video reviews how to obtain said posterior. Next, we review how to obtain realizations of it, and the code to repeat them and obtain the minimum of each one (this is the basic of the “Thompson sampling” heuristic in Bayesian optimization). With this code, six thousand minimums are obtained and three histograms are plotted:

- Marginal of x^* , probability density that in a given x^* there is a global minimum of some realization;
- Marginal of y^* , probability density that a certain y^* is the optimal value (global minimum) that achieves some realization;
- “joint” histogram approximating the joint density function over (x^*, y^*) representing the probability that some realization has the global minimum at x^* and its value is precisely y^* .

This is the implicitly available information about the location of the optimum. The goal of Bayesian optimization is to refine that information through sampling (explore) and/or to hit the optimum of my unknown underlying function as quickly as possible (exploit).

[162: boPIEN] Bayesian Optimization: probability of improvement, example

Materiales: [[CÓD.:](#) BOIntroAcquisitionShortEN.mlx] [[PDF](#)]

***  20:43

[YouTube ▶]


This video will discuss the meaning of some acquisition functions in Bayesian Optimization by means of an example. In particular, this first video will review the basic methodology, and it will set an example up with some mean, covariance kernel and dataset to work on it. If you are familiar with these concepts (if you already watched [157], [158] and [161]), you may skip to [10:30] when basic review and example setup have been already explained.

Then, the meaning of the ‘probability of improvement’ heuristic (PI) for BO acquisition function is discussed. In particular, it is shown to lie on the ‘conservative side’ if good samples (below the mean) are present: if I want to improve with almost certainty, I must descend a ‘little step’ in the downward direction (sort of gradient search), and also do not drift too far away from good samples to avoid uncertainty piling up. I will improve with large probability but, possibly, I will not jump quickly to the global minimum.

Other acquisition function options take a more ‘risky’ stance: they may not improve, but when they do improve, they may improve quite a lot more than the PI sample proposal.

[163: boEIEN] Bayesian Optimization: expected improvement, example
(+LCB,PI,EV)

Materiales: [[CÓD.:](#) BOIntroAcquisitionShortEN.mlx] [[PDF](#)]

****  19:58

[YouTube ▶]

This video, a continuation of [162], will discuss the meaning of the *Expected Improvement* (EI) acquisition function in Bayesian optimization. Basically, Expected value (Gaussian mean) should be used if getting a bad sample penalises some “utility” in my particular application, and expected improvement should be used if bad samples do not incur any cost as we have our ‘best’ sample in our historical record to provide as our final solution to the optimization problem.

A final discussion on the comparison with probability of improvement and confidence bound acquisition functions is also provided.

All discussions are supported via an example that is coincident with that in the above-mentioned video this is a continuation thereof.

[164: boloop2EN] Bayesian optimization: PI, EI, LCB detailed example (Matlab)

15:59

Materiales: [[CÓD.:](#) BOIntroLOOPENG.mlx] [[PDF](#)]

**** 


[YouTube ▶]

This video presents an example of Bayesian optimization with probability of improvement, expected improvement, and confidence interval bound as acquisition functions. The development is similar to that of the video [160], but with greater detail in the presentation of the Matlab code.

[165: boloop3EN] Bayesian optimization: bad performance examples

17:17

Materiales: [[CÓD.:](#) BOIntroLOOPENG.mlx] [[PDF](#)]

***** 

[YouTube ▶]

This video continues video [164] with additional examples. The aforementioned video is reviewed in the first two minutes, and then the following examples are discussed:

- Example with one local minimum and the global minimum at an extreme of the search range (works OK)
- Example with “overexploitation” that proposes quite unreasonable samples in its final phase.

- Example with prior confidence interval too small: it gets stuck at a local minimum
- Example with too small a bandwidth of the prior (too large correlation distance): exploration is very bad, blindsided, and ends at a point that is not even a local minimum
- Example with too large prior bandwidth (too small correlation distance): it only extrapolates to a very short distance and, therefore, progress is very slow and, before approaching the true optimal zone, it explores in an almost regular mesh (at the distance where correlation vanishes).

With all these examples, we seek to depict a complete image of the advantages and disadvantages of Bayesian optimization, and the importance of a good choice of the prior... or, if the prior is not so good, detect overexploitation and low probability of the samples (given the prior) to adjust the (hyper)parameters of said prior as samples are acquired. These issues are not considered here.