CONEM 2016
CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
21 - 25 AGOSTO DE 2016
FORTALEZA - CEARÁ

CONTROL SYSTEM DESIGN AND COMPUTATIONAL INTELLIGENCE: A SURVEY AND BRIEF TUTORIAL ON DESIGN AUTOMATION

Fausto de Oliveira Ramos, faustofor@iae.cta.br[1]
[1]Instituto de Aeronáutica e Espaço (IAE), Praça Marechal Eduardo Gomes, 50, 12228-904 - São José dos Campos - São Paulo - Brasil

Abstract. *Conception and design of a control system require a designer and a Design Set, composed of architecture, specifications, technique and controller configuration. At the same time, an implicit but important factor of Control System Design is called resource: time, human and material. As theses resources are constrained and in order to make the design feasible, there is a strong trend to simplify the original problem and abbreviate the solution search process, achieving specifications compliance with few design iterations, without knowing the improvement degree one could reach (which we call here as "Internal Flexibility"). In order to restore the equilibrium between Consistency (specifications fulfilment) and Internal Flexibility (how far one can go into that fulfilment), Control System Design Automation (CSDA) area has emerged. In this work, CSDA environment and components are presented, with examples provided from a small survey mostly from the aerospace area (a demanding and complex one). Also, in order to evoke the CSDA benefits on control engineers, a MATLAB tutorial developed by the author and an example of application are provided.*
Keywords: *computational, intelligence, design, automation, control.*

1. INTRODUCTION

This work aims to evoke the benefits of Control System Design Automation (CSDA) in control engineers, also in a practical sense. The classic design procedure implies an human operator which defines a design set, composed of (a) one or more candidate architectures, (b) systems specifications to be achieved, and (c) one or more techniques to be applied. Finally, after some iterations, (d) a controller is selected among a set of candidates, representing the best compliance with the specifications. The last step of this process poses an important limitation regarding the required effort demanded from the human designer to explore a relevant subset of the search space. In other words, the more the number the input candidates selected (e.g., weighting matrices Q and R of the linear-quadratic technique), the more the time required to finish the design. Furthermore, the designer must not only produce the controllers, but evaluate them against stability and performance metrics (among others), individually and then globally, selecting the best one. Therefore, it is logic to suppose that the human designer will be prone to abbreviate the searching process, in order to satisfy project demands such as scheduling times and human resources allocation. It is also logic to suppose that an automated environment could explore much faster and deeper the same search space, and even without dependency of the human designer, which would be free to advance other duties. In this line, the Section 2 presents certain perspectives of control system design, which motivates such automation. A small survey comes in Section 3 to reinforce such motivation. Then, Section 4 describes a mechanism and components to achieve such automation, which is applied to a flexible beam problem (Section 5); the benefits found there are evident. The last section summarizes the main concepts and findings, bringing a special proposal about the MATLAB® code developed by the author.

2. SOME COMMENTS ON CONTROL SYSTEM DESIGN

Control system design and analysis is associated with a generic Design Set $\boldsymbol{\mathcal{D}} = \{\mathcal{A}, \mathcal{S}, \mathcal{T}, \mathcal{K}\}$, containing:

- a control system architecture, $\mathcal{A}$;
- system specifications to be achieved, $\mathcal{S}$;
- a controller technique to be chosen, $\mathcal{T}$;
- a controller to be designed, $\mathcal{K}$.

In other words, given $\mathcal{A}$, compute $\mathcal{K}$ using $\mathcal{T}$ in order to meet $\mathcal{S}$.

In order to $\mathcal{D}$ be **consistent**, its elements must be reciprocally **compatible**, that is, they must comply to each other. Note that consistency is only a matter of $\mathcal{S}$ fulfilment if the specifications are perfectly and completely defined. Otherwise, a given design could satisfy such compliance at a cost – *e.g.* – of an actuation effort higher than $\mathcal{A}$ could allow.

Compatibility is assured while consistency is not affected by the interactions between the elements of $\mathcal{D}$. The compatibility degree is the **flexibility** to change the current element of $\mathcal{D}$, without a significant impact on the remaining ones. For example, in an advanced phase of the design process, substituting a PID technique for an $\mathcal{H}_\infty$ one ($\mathcal{T}$) could have serious impact on $\mathcal{K}$ (algorithm incompatibility to a higher order controller), $\mathcal{A}$ (on-board computer burden) and $\mathcal{S}$ (insufficiency of robustness specifications).

2.1 Consistency

At this point, it is worth defining a key concept:

Internal Flexibility A characteristic of a Design Set $\mathcal{D}$ related to the allowed margins for interactions between all its internal elements (or global compatibility) while still keeping its *Consistency*.

Among other factors, the knowledge of the **Internal Flexibility** of $\mathcal{D}$ is built upon:

- the architecture and constraints on $\mathcal{A}$ (physical, electrical, redundancies, environmental factors, etc.);
- the formulation and completeness of $\mathcal{S}$: as a set of objectives, how are dealt opposite ones (e.g. rise time and overshoot)? all have the same relevance? what is the trade-off between them?
- the degree of exploration of $\mathcal{T}$ and $\mathcal{K}$;
- somehow on the designer's perception. Replacing a controller $\mathbf{K}_1$ by $\mathbf{K}_2$ may result in a better accomplishment of $\mathcal{S}$: smaller settling times, higher gain margins, and so on. However, if $\mathbf{K}_1$ is already $\mathcal{S}$-compliant, **why** find $\mathbf{K}_2$?

Knowledge on *Internal Flexibility* is essential for complex and demanding areas such as aerospace control engineering, where functionality[1] is required not only for nominal conditions but also for time-varying, uncertain, perturbed, noisy and even faulty ones, for multiple operating points.

2.2 Consistency $\times$ Internal Flexibility

An implicit but important factor to control system design is called resource: time, human and material. As theses resources are constrained and in order to make the design feasible, there is a strong trend to simplify the original problem and abbreviate the solution search process. This trend assigns the main focus of $\mathcal{D}$ on its *Consistency* rather than its *Internal Flexibility*, similarly as emphasizing absolute stability (a necessary condition) with detriment of relative stability (margins to keep that condition). An automated environment to explore Internal Flexibility is a consequent thought: Control System Design Automation (CSDA) area came to fulfil such needs.

3. CONTROL SYSTEM DESIGN AUTOMATION: A SMALL SURVEY

Control System Design Automation, or CSDA, emerged as complete environments covering requirements interpretation, modelling, design and analysis. Li *et al.* (2004) attempted to set a unified scene for linear time-invariant schemes, by formulating various design schemes under index-based optimal design, hence transforming the original CACSD (Computer-Aided Control System Design) into CAutoCSD (Computer-Automated Control System Design). In Maekawa and Pang (1998), a very comprehensive control system design environment for mechanics called CSDA[2] was presented, composed of a requirement interpretation block, a modelling block, a design and analysis block, a data base / knowledge base block, and a verification block.

CSDA is built with Computational Intelligence (CI, to be introduced in the next section). Solutions combining design and CI are spreading: for many examples, see Coello *et al.* (2007) regarding engineering, scientific and industrial applications. Besides, from the previous experience of this author with other Brazilian and international researchers, and in order to motivate the reader to develop a joint work in the CSDA area, one can mention the following applications:

1. In Ramos and Leite Filho (2007), a launch vehicle model and its respective control system are considered (proportional-integral plus velocity feedback). The original design methodology considers a closed-loop transfer function with frozen poles, obtained with the LQ technique applied for the flight instant when the aerodynamic load over the vehicle is maximum. Based on that frozen t.f., the remaining gain-scheduled controllers are computed. Therefore, the LQ optimisation feature is lost for those controllers. Then, a Linear Quadratic / Genetic Design (LQ/GD) approach is applied, taking into account not only stability and performance specifications, but also interpolation smoothness of the gain-scheduled controllers. The comparison is satisfactory, where the smoothness degree of the LQ/GD solution is superior. Besides, as previewed, the LQ optimization is fully extended to the flight time range considered, given that the maximum control effort is reduced, as proven by Hardware-in-the-Loop simulations.

[1] Not to mention other important aspects such as mass, power consumption, ruggedizing, cost, etc. .

[2] CSDA is used specifically here in connection with Maekawa and Pang (1998), but generically in the remaining of this work.

2. In Ramos (2011), a CI-based mechanism is combined with the Observer-Based Realization (OBR) technique of Alazard and Apkarian (1999), for the computation of an $\mathcal{H}_\infty$ controller. The OBR technique allows a controller with suitable order to function also as an observer. For this, the set of OBR model's closed-loop eigenvalues must be explored regarding its distribution as state-feedback, state-estimator and Youla parameter ones. Besides, the OBR model comprised the estimation of a bias fault at the angular velocity sensor output, related to the roll plane attitude control system of a launch vehicle (see also Ramos and Alazard (2010) on the estimations of attack and pitch angles simultaneously to a bias fault). In parallel, CSDA was employed to compute gain-scheduled controllers according requirements on stability, performance and smoothness. A non-linear digital simulation revealed an abrupt bias fault on the roll sensor output, as expected.

3. An interesting mass-spring-mass benchmark problem was proposed by Wie and Bernstein (1992) in a control periodical. Later on, Thompson (1995) presented its own solution with another eleven ones of previous contributions. Thompson's own contribution was ranked in the first place. In order to evaluate the power of CSDA, Ramos (2015) redesigned the worst ranked in Thompson's work with CI. The resultant control system brought improvements on most indexes (stability, performance and parameter variation), specially a 99% reduction of the original maximum control effort, plus a 180% increase of the original settling time, while still not violating the respective requirement. Finally, the new controller left the last position of the score, awarding the 3^{rd} place of 12 designs. It was even possible to achieve the 1^{st} place, but with a degraded performance, due to the replacement of the CSDA metric element by the original metric adopted in Thompson (1995). That metric was shown imperfect, as it promoted a poor design to a high score. Therefore, CSDA even found a defect in the original metric.

4. Ramos and Souza (2008) presented the embedding of CI in well-known techniques available for control system design, applied to satellite attitude control system by using a reaction wheel. It was shown that effective search and scoring procedures can replace human-performed trial-and-improvement actions for gain computation according with four design techniques (including linear-quadratic and H_2), producing performance indexes and torque levels compatible with real world specifications. A comparison of the controllers found with each technique is provided, based on the hypothetical example FireSat proposed by Souza (2006). Posteriorly, Ramos and Alazard (2009) combined this same approach with OBR in order to provide estimates of the satellite model state-space vector.

4. BASICS OF COMPUTATIONAL INTELLIGENCE

NOTE: this section is a brief summary; the reader may refer to Ramos (2011) for details.

Artificial Intelligence (AI) and Computational Intelligence (CI) are competing fields associated with respectively symbolic and sub-symbolic (*e.g.*, numeric) representations aimed for building intelligent systems. AI relies on a top-down strategy (that is, each system or ability is viewed as a black-box where its content is not primarily important) while CI is built from small components (as the GA chromosomes) without having a priori conception of the final system.

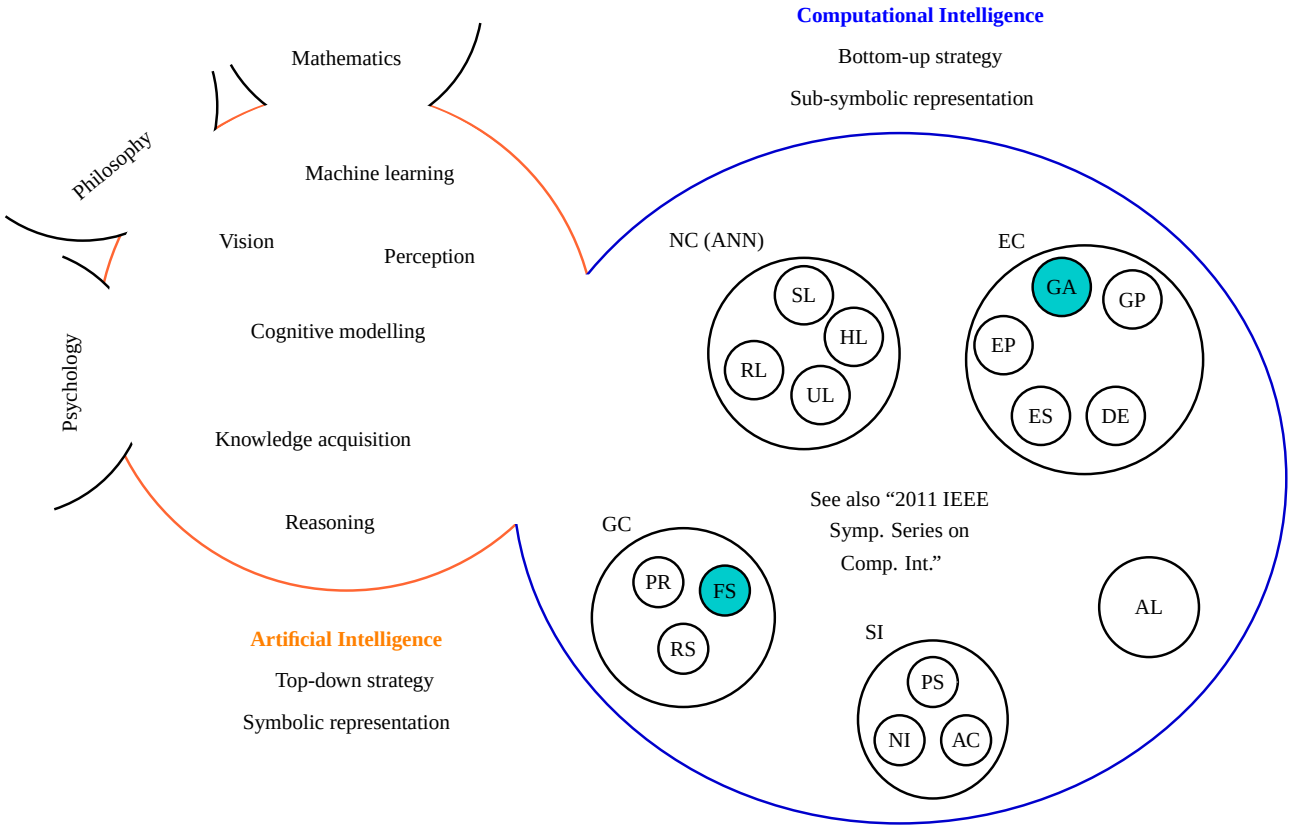
There is low agreement on the definitions of AI and CI, but a major consensus about an overlapping nature between them (see the Fig. 1). For terminology and discussions about it, see Konar (2005), Rudas and Fodor (2008) and Craenen and Eiben (2002). CI area is composed of many sub-areas (see the Fig. 1) such as Granular Computing (Fuzzy Systems (FS) included), Neuro-computing (Neural Networks (NN) included) and Evolutionary Computation (EC, Genetic Algorithms (GA) included) to name the most known, along with hybrid versions of them, as well. Two important elements are used in this work, namely GA and FS; thus, one presents them with more details.

4.1 Genetic algorithms

One of the most recognized pioneers in GAs is Bremermann (see Fogel and Anderson (2000)), who originally proposed them as general models of adaptive processes (although posteriorly applied popularly as optimization and search tools). GAs are basically the representation of the Darwinian theory of the life evolution on virtual populations or sets of individuals. Each individual is composed of chromosomes, which refer to independent entities forming the full system. By its turn, each chromosome is characterized by a set of genes. For example, the three-axis control system of a launch vehicle (the individual) could be devised as three chromosomes (each one representing the controller of the respective plane of manoeuvring), with 3 genes per chromosome (proportional, derivative and integral gains of each controller).

The GA operation cycle is basically composed of the following actions:

- **Reproduction.** Clones are produced with multiplicity proportional to the ratings of their parents.
- **Crossover.** Segments of genes are randomly exchanged between individuals.
- **Mutation.** Genes are changed randomly with a given probability.
- **Elite Reinsertion.** The individual of the original set with the best rating is preserved in the new set; that is necessary due to the mutation and crossover actions.



Evolutionary Computing (EC) = {Genetic Algorithms (GA), Genetic Programming (GP), Evol. Programming (EP), Evol. Strategies (ES), Differential Evolution (DE)}
Neuro-computing (NC) = Artificial Neural Networks (ANN) with {Superv. Learning (SL), Unsupervised Learning (UL), Hybrid Learning (HL) and Reinforcement Learning (RL)}
Granular Comp. (GC) = {Fuzzy Systems (FS), Probab. Reasoning (PR), Rough Sets (RS)}
Swarm Intelligence (SI) = {Particle Swarm (PS), Nature Inspired (NI), Ant Colony (AC)}
Artificial Life (AL)

Figura 1: Block diagram of the computational intelligence field.
Source: adapted from Konar (2005).

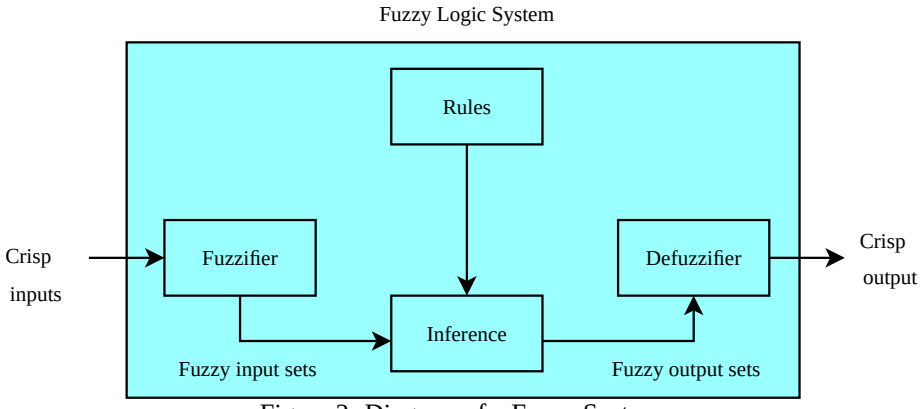


Figura 2: Diagram of a Fuzzy System.
Source: Mendel (1995).

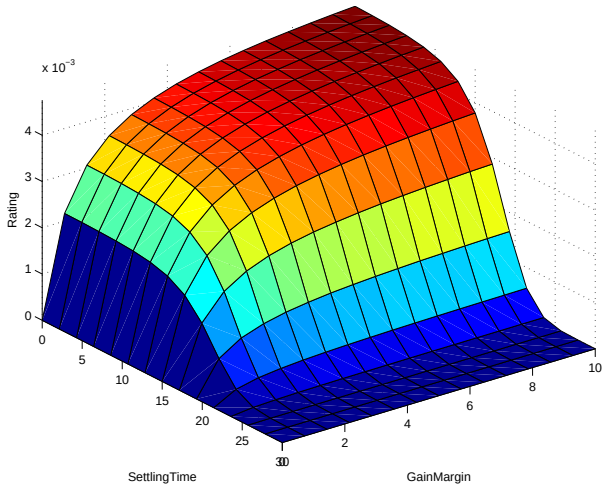


Figura 3: Example of a FS mapping.

4.2 Fuzzy Systems

Zadeh (1965) defined Fuzzy Sets as

> a class of objects with a continuum of grades of membership

opposing the usual mathematical sense (*e.g.*, the sets "*real numbers greater than one*" and "*real numbers MUCH greater than one*"). According to Mendel (1995), a Fuzzy System (FS, see Fig. 2) is a non-linear mapping of an input vector into a scalar output[3] (*e.g.* Fig. 3), composed of Linguistic Variables, Fuzzy Sentences and Fuzzy Rules. The Fuzzy Sentences adopted in this work are Mamdani ones, based on mathematical expressions such as Gaussian or polynomial functions[4], with engineering specifications as input Linguistic Variables (*e.g.*, "*Settling Time*"). The output Linguistic Variable could be "*Rating*". Each Linguistic Variable comprises the respective Fuzzy Sentences and an Universe of Discourse.

In formal language, a FS could be described with the following illustrative statements:

- the Linguistic Variable "*Overshoot*" is associated with the overshoot of the system step response, where its Universe of Discourse is [0, 100] [%]. The Fuzzy Sentence *"Overshoot is small"* is defined by a z-polynomial function (Eq. 1) and the pair $\langle a, b \rangle$, with $a = 0$ and $b = 20$;

$$f(x) = \begin{cases} 1, x \leq a \\ 1 - 2\left[(x-a)/(b-a)\right]^2, a < x \leq (a+b)/2 \\ 2\left[b - x/(b-a)\right]^2, (a+b)/2 < x \leq b \\ 0, x > b \end{cases} \tag{1}$$

- the Fuzzy System rules are logical combinations of Fuzzy Sentences (Eq. 2):

$$\begin{array}{l} R_1\text{: If (“}\textit{Actuation is Large}\text{”) then (“}\textit{Rating is Bad}\text{”)} \\ R_2\text{: If (“}\textit{Actuation is not Large}\text{”) and (“}\textit{Settling Time is Satisfactory}\text{”) then (“}\textit{Rating is Regular}\text{”)} \\ R_3\text{: If (“}\textit{Actuation is Satisfactory}\text{”) and (“}\textit{Settling Time is Satisfactory}\text{”) then (“}\textit{Rating is Good}\text{”)} \end{array} \tag{2}$$

The use of Fuzzy Systems as elements aimed for requirement evaluation in automated control system design was firstly proposed by this author. Such feature will be discussed in the next subsection.

4.3 CI Mechanism for CSDA

A model of a CSDA CI-based mechanism proposed by this author is shown in the Fig. 4, and its operation follows. Firstly, a GA generates, combines and mutates individuals, reinserting the best fitted one (the elite) of the last generation in the current one. The individual is a chromosome given by the matrix (or vector) $\mathbf{K}$, with its $K_{i,j}$ elements as genes. $\mathbf{K}$ is not always a gain, but can assume other forms, as the weighting values of the LQ design used in the example of the Sec. 5. Secondly, a design process takes these individuals and produces controllers. Thirdly, indexes (stability, performance, etc.) are extracted from the resulting control systems formed with the controllers. Finally, these indexes are evaluated with a cost function (in this case, a FS); then, clones of the best rated individuals compose a new generation and the process continues until a stop criterion is met. At this moment, an elite individual – $\mathbf{K}_{elite}$ – is delivered to external validation by means of digital and Hardware-in-the-Loop simulations[5]. Finally, a validated controller $\mathbb{K}$ is found.

[3] Refer to Mathworks (2010) for a lengthy and more didactic explanation of these concepts.
[4] Refer to the MATLAB® set of mathematical functions linked to the FS environment.
[5] It could be used simply as a baseline controller, if desired.

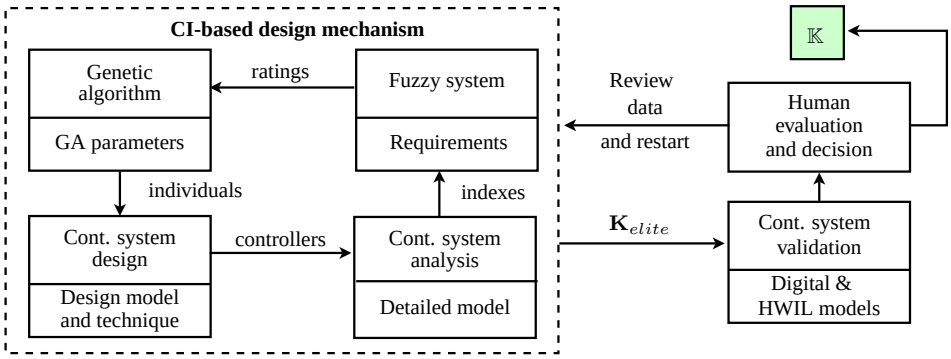


Figura 4: Example of a CI-based mechanism used for CSDA.

5. AN EXAMPLE AND TUTORIAL

In order to illustrate CSDA, one chose a CONEM 2010's paper where Guedes *et al.* (2010) computed a Linear-Quadratic (LQ) controller for a flexible beam problem. The same design was computed with CSDA code developed by this author in the MATLAB® environment. Then, both results were compared against qualitative performance specifications suggested by Guedes *et al.* (2010).

5.1 Design Set

The linear model used in the design (Eq. 3) is given by a flexible beam coupled to a QUANSER's SRV02-Series rotary servo base unit. For the sake of space, only the most relevant variables are named (angles in radians): θ is the servo load gear angle, α is the angle of the arm deflection, and V_m is the voltage applied to the electric motor of the servo unit.

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_{Stiff}}{J_{eq}} & \frac{B_{eq}\ R_m - \eta_m\ \eta_g K_t\ K_m\ K_g^2}{J_{eq}\ R_m} & 0 \\ 0 & -\frac{K_{Stiff}\ (J_{eq}+J_{Arm})}{J_{eq}} & \frac{B_{eq}\ R_m + \eta_m\ \eta_g K_t\ K_m\ K_g^2}{J_{eq}\ R_m} & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \frac{\eta_m\ \eta_g\ K_t\ K_g}{J_{eq}\ R_m} \\ -\frac{\eta_m\ \eta_g\ K_t\ K_g}{J_{eq}\ R_m} \end{bmatrix} \mathbf{V_m}, \text{ where: } \mathbf{x} = \begin{bmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix}, \tag{3}$$

An LQ controller $\mathbf{K}_G$ (Eq. 4) was found by Guedes *et al.* (2010) after trying somehow "*time consuming*" combinations for the main diagonal elements of the weighting matrix $\mathbf{Q}$, with $R = 1$. According to the author, "*the controlled system has a good response, (· · ·) presenting a small overshoot, a small rise time and also a small settling time.*". Those values were not presented, but it can be shown as being respectively 3.8337%, 0.2975s and 0.6431s (Fig. 5). However, a key feature of any design is the actuation level. The proposed controller, in this case, yielded a maximum voltage of 15.8697V, exceeding the nominal value specified for the servo electric motor (according Quanser (2014), 6V), meaning probably a **Consistency** problem of the Design Set (incomplete $\mathcal{S}$).

$$\mathbf{Q}_G = \begin{bmatrix} 255 & 0 & 0 & 0 \\ 0 & 3500 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ R_G = 1,\ \mathbf{K}_G = \begin{bmatrix} 15.9687 & -51.5602 & 2.1649 & 0.5598 \end{bmatrix} \tag{4}$$

5.2 Design Automation

With CSDA, the controller structure is based on a single chromosome with genes as binary n-bit numbers[6]. According to the Eq. 5 (where MSB is the Most Significant Bit of a gene), each pair of genes is associated to the Q_{ii} element of the $\mathbf{Q}$ main diagonal. Therefore, the four LQ parameters were translated to eight genes.

$$Q_{ii} = \frac{g_{k,1}}{g_{k,2}},\ \text{where: } 0 \leq g_{k,1} < 2^n,\ 1 \leq g_{k,2} < 2^n \tag{5}$$

The FS is defined by Fuzzy Sentences presented graphically in the Fig. 6 and Fuzzy Rules shown in the Eq. 2 (see again the Sec. 4.2).

[6] In this case, n = 8.

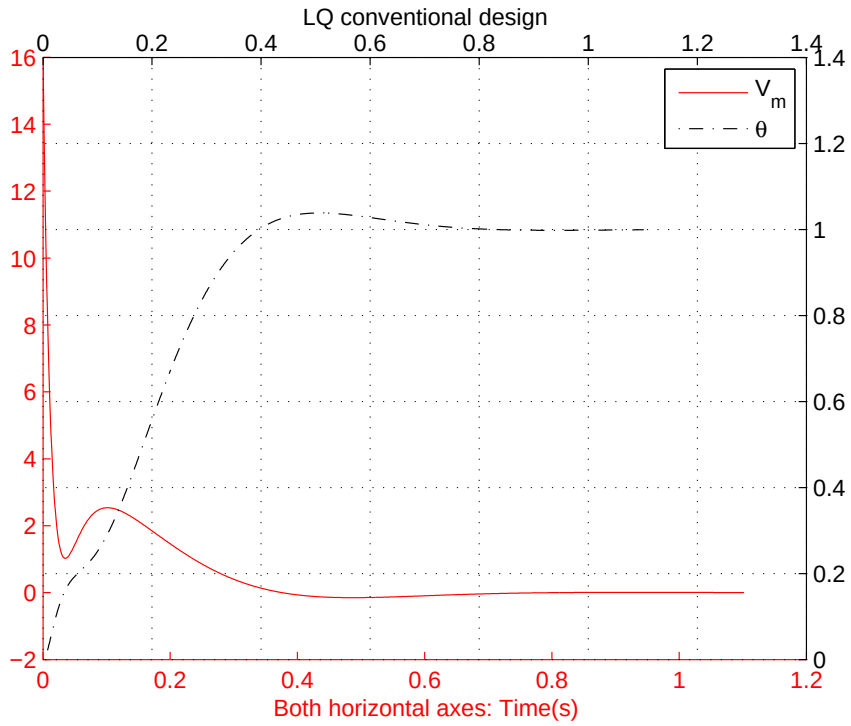


Figura 5: Angle θ (radians) and voltage V_m (volts) found in the conventional LQ design (Guedes *et al.* (2010)).

5.3 Results

Despite the huge search space ($2^{8\times2\times4}$ possible combinations), after testing 4000 individuals in less than 6 minutes with a Intel®Core™ i3-2310M CPU @ 2.10GHz, the CI elite controller was found (Eq. 6) yielding a settling time of 0.4391s and maximum voltage applied to the servo unit of 5V (near the nominal value). Compared with the conventional design, it brought 31.7213% reduction of the original settling time and 68.4934% reduction of the original maximum actuation voltage V_m (see Fig. 7). One can see the amount of **Internal Flexibility** remaining in the conventional design. Besides, no overshoot was produced with CSDA.

$$\mathbf{Q}_G = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 0.0042 & 0 & 0 \\ 0 & 0 & 0.0039 & 0 \\ 0 & 0 & 0 & 0.4539 \end{bmatrix}, \; R_G = 1, \; \mathbf{K}_G = \begin{bmatrix} 5.0000 & -5.8658 & 0.4145 & -0.0975 \end{bmatrix} \tag{6}$$

5.4 Comments

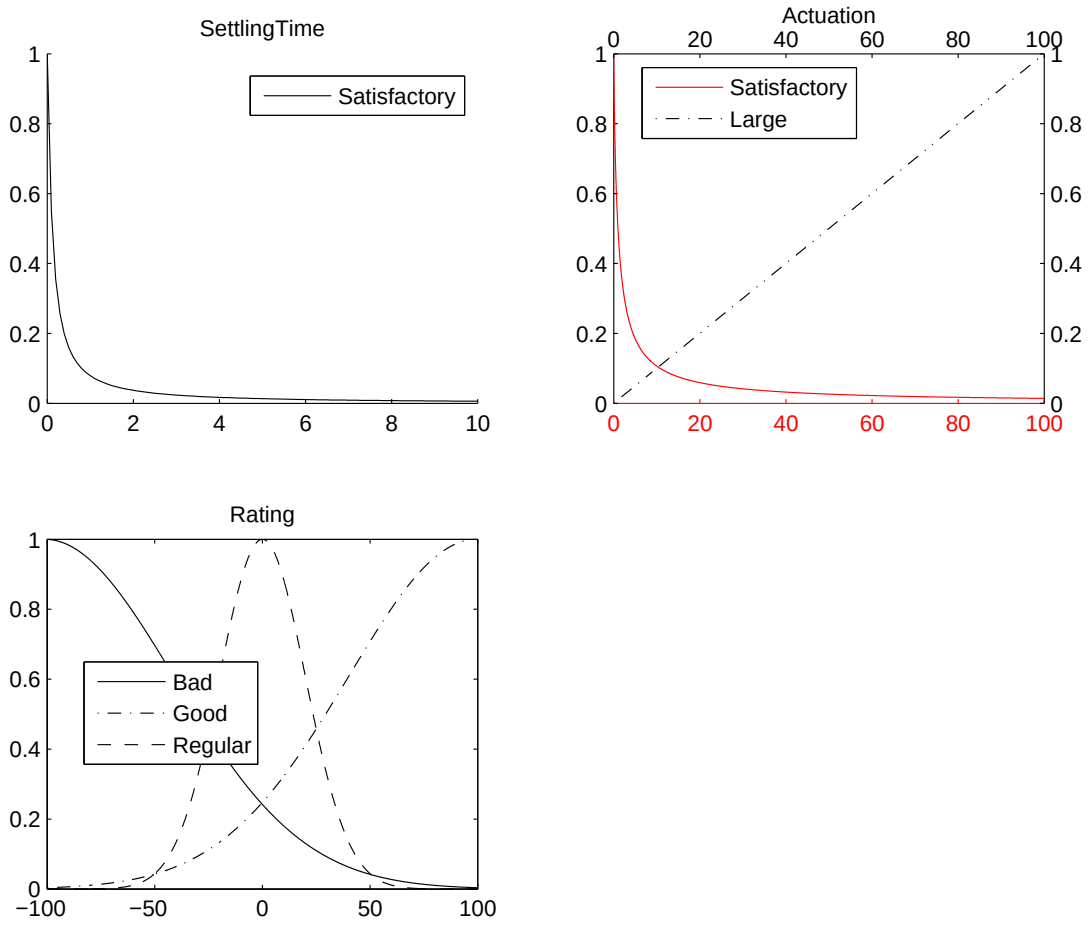
Regarding the Design Set $\boldsymbol{\mathcal{D}}$ of the flexible beam design, some possibilities arise:

- CSDA reveals the **Internal Flexibility** of $\boldsymbol{\mathcal{D}}$ and possible inconsistencies, if any;
- CSDA can explore a search space much higher than an human designer would, in a unachievable speed;
- CSDA can suggest a base design controller for a given $\boldsymbol{\mathcal{D}}$, which the designer can start with.

5.5 Tutorial

The results presented in this section were produced with a set of MATLAB® scripts developed by this author, named "*Evolutionary Design Box*" (EDB for short). In order to provide a glance about the CSDA process, a summarized procedure is presented, taking into account the following EDB scripts:

- `ci.m`, the main script;
- `data.m`, containing the input data required by the CSDA;
- `gen_alg.m`, where the high level functions of the CI mechanism are located;
- `world.m`, where the low level functions of the CI mechanism are located;
- `models.m`, containing the models used;



NOTES: (*i*) all vertical axes and "Rating" both ones present ratings; (*ii*) horizontal axes: Settling Time (seconds), Actuation (volts).
Figura 6: Fuzzy Sentences for the flexible beam CSDA: "*Satisfactory Settling Time*", "*Large Actuation*" and "*Satisfactory Actuation*", "*Bad Rating*", "*Regular Rating*" and "*Good Rating*".

- `analysis.m`, where the control system is assembled and indexes extracted;
- `evaluation.m`, where the ratings are computed from the indexes, according with a FS file (in the case of the flexible beam problem, named as `flx_lnk.fis`).

For a new CSDA run (*e.g.* the flexible beam problem, shown in **bold letters**), one may roughly follow these steps:

1. define the input data, regarding (*i*) the parameters of the computational intelligence (*e.g.*, **the controller parameter labels,** `Q11, Q22, Q33, Q44`) and (*ii*) the control system definition and simulation environment (*e.g.*, **the definition of the design technique** $\mathcal{T}$, `lqr(P_,Q_,1)`);
2. define the models (*e.g.*, **state-space system attached with input and output labels**);
3. define the control system (*e.g.*, **using** `sumblk`, `connect` **and similar functions**);
4. define the FS to be used for ratings production from indexes; actually, simply evoke the command `fuzzy` in MATLAB® workspace (assuming that the toolbox is installed) and build by your own;
5. adapt the `evaluation.m` script in order to reflect the FS file name and the ratings / indexes used;
6. finally, run the script `ci.m`.

A book is being currently written about EDB and the whole detailed procedure.

6. CONCLUSIONS AND FINAL REMARKS

By briefly presenting key concepts and providing a survey, overview, example and tutorial, this work aimed to offer an interesting view of the benefits of Control System Design Automation (CSDA). In order to introduce the subject, one defined the term *Internal Flexibility* for a given Design Set $\mathcal{D}$. It was seen how *Consistency* may obfuscate *Internal*

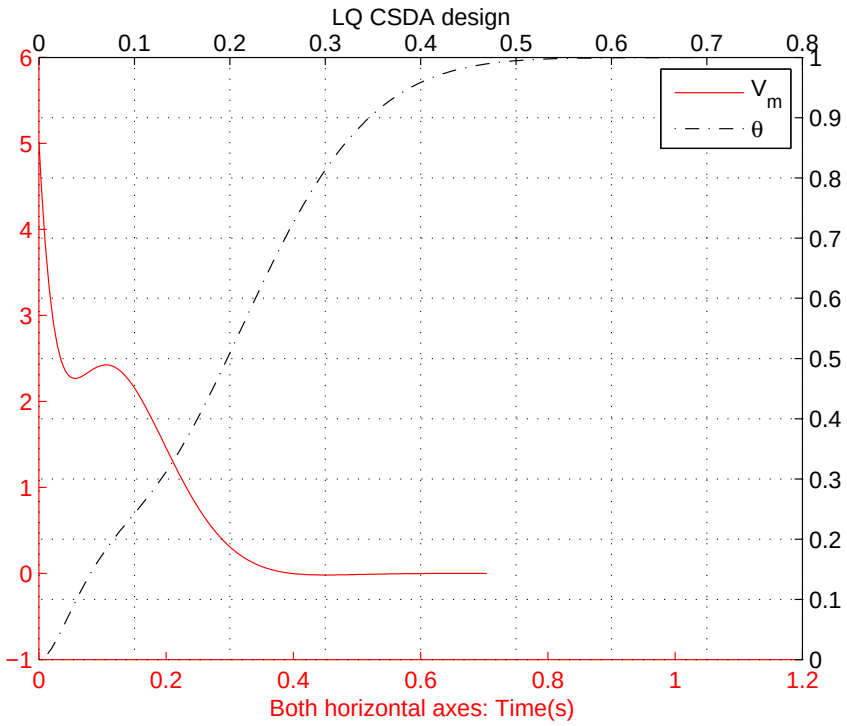


Figura 7: Angle θ (radians) and voltage V_m (volts) found by the CSDA LQ design.

Flexibility, and how CSDA can help to sustain a balance on both. By using CI elements, one can explore a design space, recovering somewhat the original control problem which is highly prone to be abbreviated due to resource constraints.

Computational Intelligence (CI) and correlate concepts were defined. Two relevant CI elements were introduced (Genetic Algorithms and Fuzzy Systems), so that a CI mechanism could be devised. Then, this mechanism was applied to a flexible beam problem. Compared with a conventional design, a CSDA improved overshootless one was obtained: a smaller settling time and a vigorous reduction of the actuation effort were achieved.

Furthermore, those better results found in a time span of mere 6 minutes were produced autonomously by a CI mechanism, leaving the human designer to advance other duties. In such perspective - speed - it is quite clear the superiority of the automated approach compared with the conventional one.

Finally, a tutorial was provided, based on the same CSDA code developed by the author for the MATLAB® environment and used to solve the flexible beam problem. It is expected that control engineers be awake to the benefits and advantages of CSDA; collaborative work between them (**you!**) and this author are always welcome.

7. BIBLIOGRAPHY

Alazard, D. and Apkarian, P., 1999. "Exact observer-based structures for arbitrary compensators". *International Journal of Robust and Non-linear control*, , No. 9, pp. 101–118.

Coello, C.A.C., Lamont, G.B. and van Veldhuizen, D.A., 2007. *Evolutionary algorithms for solving multi-objective problems*. Genetic and evolutionary computation. Springer Science+Business Media, LLC, New York, 2nd edition.

Craenen, B.C. and Eiben, A.E., 2002. "Computational intelligence". `http://www.eolss.net/`.

Fogel, D.B. and Anderson, R.W., 2000. "Revisiting bremermann's genetic algorithm. i. simultaneous mutation of all parameters". In *Proceedings...* IEEE Congress on Evolutionary Computation, San Diego, pp. 1204–1209.

Guedes, F.M., Santos, S.P. and Souza, L.C.G., 2010. "Attitude and vibration control of a rigid flexible link using LQR and H_∞ methods". In *Proc. of the VI National Congress of Mechanical Engineering*.

Konar, A., 2005. *Computational Intelligence: principles, techniques and applications*. Springer, New York.

Li, Y., Ang, K.H., Chong, G.C., Feng, W. and Tan, K.C., 2004. "CAutoCSD - evolutionary search and optimization enabled computer automated control system design". *International Journal of Automation and Computing*, Vol. 1, No. 1, pp. 76–88.

Maekawa, K. and Pang, G.K.H., 1998. "Control system design automation for mechanical systems". *Journal of Intelligent & Robotic Systems*, Vol. 21, pp. 239–256.

Mathworks, 2010. *Fuzzy Logic Toolbox 2 user's guide*. MathWorks, Natick. URL `http://www.mathworks.com`.

Mendel, J., 1995. "Fuzzy logic systems for engineering: a tutorial". *Proceedings of the IEEE*, Vol. 83, No. 3, pp. 345 –377.

Quanser, 2014. *The Rotary Control Lab - A Modular Single Source Solution You Can Control*.

Ramos, F.O., 2011. *Automation of $\mathcal{H}_\infty$ controller design and its observer-based realization*. Ph.D. thesis, Double

diploma: Instituto Nacional de Pesquisas Espaciais and Institute Supérieur de l'Aéronautique et de l'Espace, São José dos Campos (Brazil) and Toulouse (France).

Ramos, F.O., 2015. "Computational intelligence for control system design automation". In *Proc. of the European Aerospace GNC Conference*.

Ramos, F.O. and Alazard, D., 2009. "Observer-based form of a computational-intelligence designed satellite attitude controller". In *Proc. of the* 3^{rd} *European Conference for Aerospace Sciences*.

Ramos, F.O. and Alazard, D., 2010. "Semi-automated observer-based controller design". In *Proc. of the* 11^{th} *Pan-American Congress of Applied Mechanics*.

Ramos, F.O. and Leite Filho, W.C., 2007. "Design upgrade and validation of a launcher vehicle attitude controller combining genetic and linear quadratic optimization". In *Proc.* 19^{th} *International Congress of Mechanical Engineering*.

Ramos, F.O. and Souza, L.C.G., 2008. "CI-based design of a satellite attitude controller". In *Proc. of the* 9^{th} *International Conference on Motion and Vibration Control*.

Rudas, I.J. and Fodor, J., 2008. "Intelligent systems". *International Journal of Computers, Communications & Control*, Vol. III, No. S., pp. 132–138.

Souza, L.C.G., 2006. "Satellite attitude control system parameters optimization". In *Proceedings of the III European Conference on Computational Mechanics*.

Thompson, P.M., 1995. "Classical/$\mathcal{H}_2$ solution for a robust control design benchmark problem". *Journal of Guidance, Control and Dynamics*, Vol. 18, No. 1, pp. 160–169.

Wie, B. and Bernstein, D.S., 1992. "Benchmark problems for robust control design". *Journal of Guidance, Control and Dynamics*, Vol. 15, No. 5, pp. 1057–1059.

Zadeh, L.A., 1965. "Fuzzy Sets". *Information and Control*, Vol. 8, No. 3, pp. 338–353.

8. RESPONSIBILITY NOTICE

The author is the only responsible for the printed material included in this paper.