4

Development of a Simulation Environment Applied to the Study of Fault-Tolerant Control Systems in Robotic Manipulators. Theoretical and Practical Comparisons

Claudio Urrea and John Kern
Departamento de Ingeniería Eléctrica, DIE
Universidad de Santiago de Chile, USACH, Santiago
Chile

1. Introduction

Industrial processes governed by automatic controllers can be subject to failures. Their incorrect operation can cause economic losses, workers' hazards, inconvenience for users, etc. Moreover, automation of processes by means of automatic control loops, although permitted freeing human operators from manual control and operation, has not immunized them from failures. A way to enhance their reliability is to provide them with tolerance mechanisms to face these failures (Blanke et al., 2001).

It is understood as 'failure' every change in the behavior of any component of the system (non-permitted deviation of any characteristic property or parameter), so the system cannot satisfy the function it has been designed for (Blanke et al., 2006). Control loops can hide failures preventing their detection to the point of causing irreparable damages forcing to stop the system or process.

Because of this, there is a growing need and concern for developing control systems capable of acceptable operation even in the case of failures, also being able to stop the process before irreparable damages arise. This kind of control systems are called 'failure tolerant': control systems that can, in the one hand, detect incipient failures in sensors and/ or actuators; and, in the other, adapt control laws quickly to preserve the specified operation in terms of production quality, safety, etc. (Bonivento et al., 2004). Figure 1 shows the block diagram corresponding to a fault-tolerant control system.

(Blanke et al., 2006).

In this chapter the development of a simulation environment applied to the study of fault tolerant control systems for robotic manipulators by employing MatLab-Simulink programming tools is presented. By using this simulator is possible to represent the behavior of a manipulator: in the presence of a failure into one of its actuators and the status of correction of such failure, through an active fault tolerant control system. The end-effector Cartesian trajectories and the evolution of the joints of the manipulator are provided by means of animations and graphics in a comfortable and intuitive programming environment.

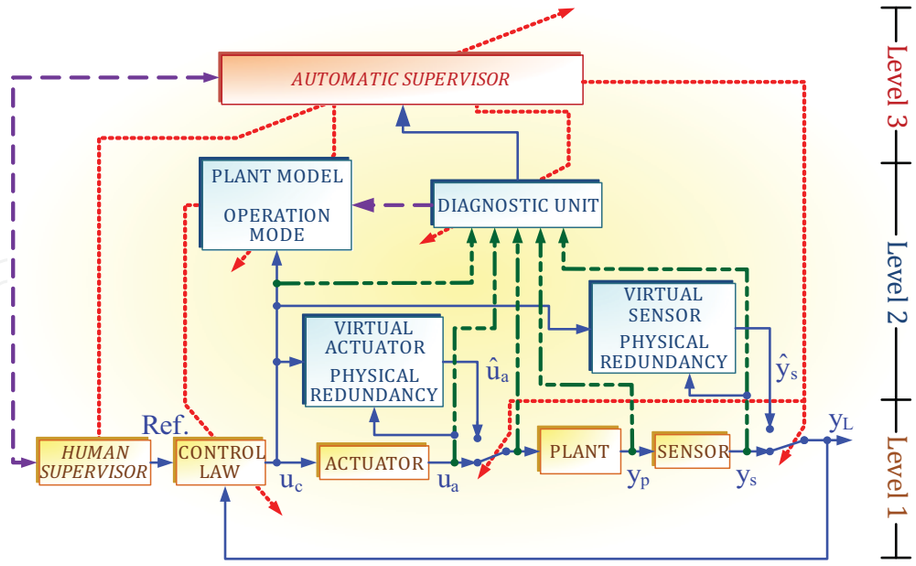


Fig. 1. Block diagram of a fault tolerant control system

2. Fault tolerant control

In simple terms, we can say that classic control is based in a control law ϑ that considers a set of objectives O and a set of constraints $\Re$, associated to the system's dynamic model. Such dynamic model frequently presents some inaccuracies, if we consider, for example (Arteaga et al., 2004).

- External perturbations.
- Time-varying parameters.
- Parameters with uncertainties.
- Noise in measurements.
- Unmodeled dynamics.

what brings as a consequence some limitations in the use of such classic controllers that need exact system parameters for a correct performance. A way to confront those uncertainties is to consider a fixed structure model that includes a set of unknown parameters to apply robust control and adaptive control techniques. Robust control and adaptive control for robots consider satisfactory performance in spite of uncertainties in inertial parameters of the joint model, external interference and unmodeled dynamics behaviour (Spong et al., 2005), (Kelly et al., 2005), (Craig, 1988). Specifically, robust control includes a control law that takes into account all the set of objectives and all the set of Θ parameters (known and unknown), and adaptive control is based on the estimation of on-line parameters, *i.e.* the value of unknown $\hat{\theta}$ parameters in the set of all possible system parameters.

When a failure in the system occurs, it is necessary to modify the set of objectives so to obtain an adequate control law to face the event, since both system constraints ($\hat{\Re}_f$) and unknown parameters ($\hat{\theta}_f$) can experience modifications. In order to overcome those requirements, fault tolerant control considers the modification of the set of objectives (O_f), creating the conditions of optimum operation and degraded operation for the system with no failures and in presence of failures, respectively, as shown in figure 2 (Blanke et al., 2001), (Puig et al., 2004), (Arteaga et al., 2004).

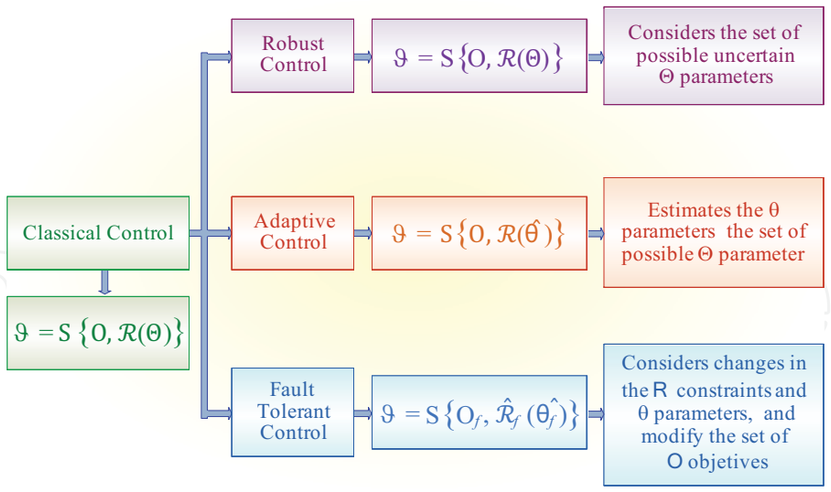


Fig. 2. Schematic diagram showing the types of control used in systems with parameter uncertainties and failures

The occurrence of failures in the system and the activation of tolerance mechanisms can be modeled as time discrete events, therefore fault tolerant control systems have and hybrid nature (Tsuda et al., 2001). When a sensor failure arises, this can be detected because the sensor reading will exceed some pre-determined threshold, in comparison with an expected value; under these conditions, fault tolerant control will not consider the information coming from the defective sensor, locking the associated joint. If there's a failure in a motor without a safety backup, the joint locks again, and the robot loses mobility of this joint. In any case, the robot loses a part of its movement range, and the ability to carry out its assigned tasks is limited (El-Salam et al., 2005).

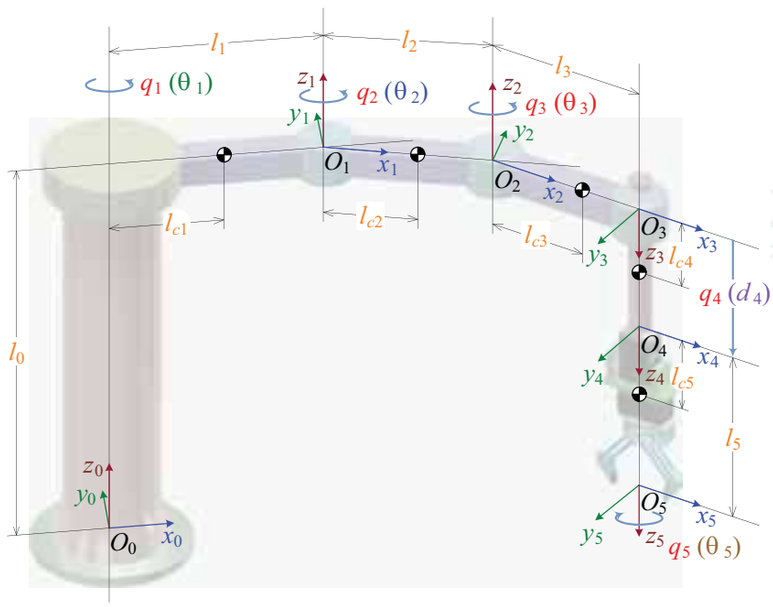


Fig. 3. Scheme of a redundant robotic manipulator of SCARA type

Having in mind the above mentioned failure behaviors, we will consider now the three first DOF of the robotic manipulator shown in figure 3, and the appearance of a failure in an actuator causing the blocking of the joint it commands, specifically joint number two, shown (in red) in figure 4. In such a situation, we consider a desired destiny point, given by coordinates (*x*,*y*), that under the action of a classic controller and in presence of the above mentioned failure, the manipulator is far from reaching.

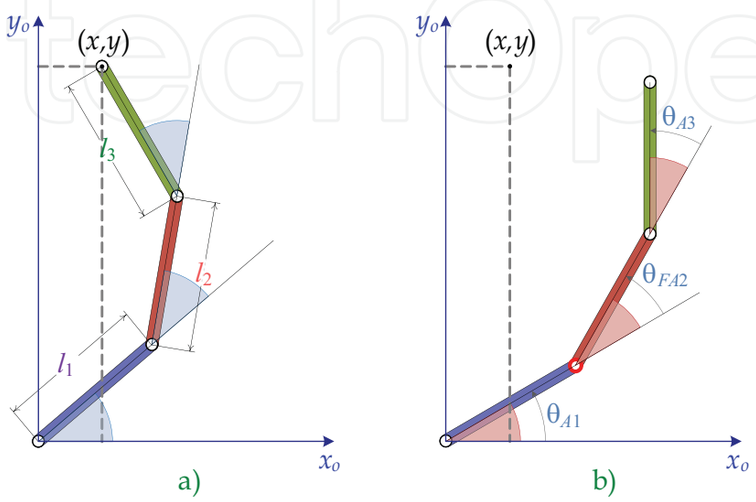


Fig. 4. Disposition of the three first DOF in the *x*-*y* plane of the redundant manipulator: a) Arriving to the destiny position (*x*,*y*); b) Not arriving to the destiny position (*x*,*y*): failure in actuator 2

For the design of fault tolerant control, it has been considered the generation of a virtual link, from the failing actuator that is blocked and its adjacent links, as seen in figure 5 (gray color):

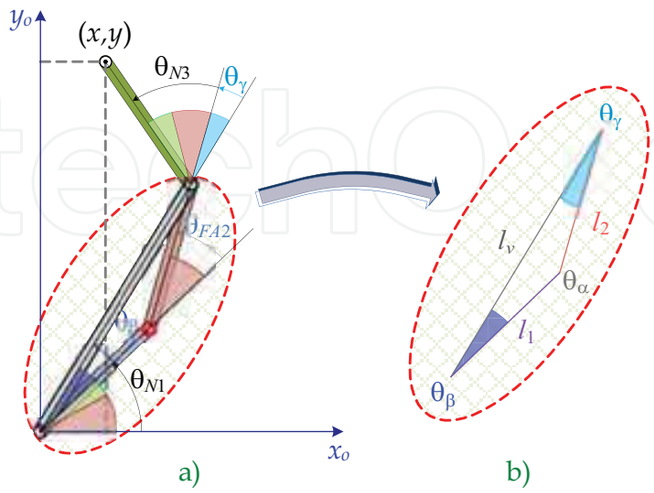


Fig. 5. Disposition of the three first DOF in the *x*-*y* plane of the redundant manipulator: a) Additional displacement of actuators 1 and 3, b) Equivalent diagram of displacement angles

3. Fault tolerant controller: Computed torque

Considering the hybrid nature of fault tolerant control, it is proposed an active fault tolerant control, including a different control law in function of the robotic manipulator status, *i.e.*, normal or failing, sensing possible failures on-line and, in accordance with that, reconfiguring the controller by selecting the most adequate control law (changing inputs and outputs) (figure 6). As classic controller it has been selected the computed torque controller. If we consider that the dynamic model of a manipulator with n joints can be expressed through the equation (1):

$$\boldsymbol{\tau} = \mathbf{M}(q)\ddot{\mathbf{q}} + \mathbf{C}(q,\dot{q}) + \mathbf{G}(q) + \mathbf{F}(\dot{q}) \tag{1}$$

where:

$\boldsymbol{\tau}$: Vector of Generalized forces ($n\times1$ dimension).
$\mathbf{M}$: Inertia matrix ($n\times n$ dimension).
$\mathbf{C}$: Vector of centrifugal and Coriolis forces ($n\times1$ dimension).
q : Components of joint position vector.
$\dot{q}$: Components of joint speed vector.
$\mathbf{G}$: Vector of gravity force ($n\times1$ dimension).
$\ddot{q}$: Joint acceleration vector ($n\times1$ dimension).
$\mathbf{F}$: Friction forces vector ($n\times1$ dimension).

The control law by computed torque consists in the application of a computed torque in order to compensate centrifugal, Coriolis, gravitatory and friction effects, as can be seen in equation (2).

$$\boldsymbol{\tau} = \hat{\mathbf{M}}(q)\left(\ddot{\mathbf{q}}_\mathbf{d} + \mathbf{K}_\mathbf{v}\dot{\mathbf{q}}_\mathbf{e} + \mathbf{K}_\mathbf{p}\mathbf{q}_\mathbf{e}\right) + \hat{\mathbf{C}}(\dot{q},\dot{q}) + \hat{\mathbf{G}}(q) + \hat{\mathbf{F}}(\dot{q}) \tag{2}$$

where:

$\hat{\mathbf{M}}$: Estimation of inertia matrix ($n\times n$ dimension).
$\hat{\mathbf{C}}$: Estimation of centrifugal and Coriolis forces vector ($n\times1$ dimension).
$\hat{\mathbf{G}}$: Estimation of the gravity force vector ($n\times1$ dimension).
$\hat{\mathbf{F}}$: Estimation of the friction forces vector ($n\times1$ dimension).

$$\mathbf{K}_\mathbf{v} = \begin{bmatrix} \mathbf{K}_{v1} & & & \\ & \mathbf{K}_{v2} & & \\ & & \ddots & \\ & & & \mathbf{K}_{vn} \end{bmatrix} \tag{3}$$

$\mathbf{Kv}$: Definite positive diagonal matrix ($n\times n$ dimension).

$$\mathbf{K}_\mathbf{p} = \begin{bmatrix} \mathbf{K}_{p1} & & & \\ & \mathbf{K}_{p2} & & \\ & & \ddots & \\ & & & \mathbf{K}_{pn} \end{bmatrix} \tag{4}$$

$\mathbf{Kp}$: Definite positive diagonal matrix ($n \times n$ dimension).
$\ddot{\mathbf{q}}_d$: Joint desired acceleration vector ($n \times 1$ dimension).

$$\mathbf{q}_e = \mathbf{q}_d - \mathbf{q} \tag{5}$$

$\mathbf{q}_e$: Joint position error vector ($n \times 1$ dimension).

$$\dot{\mathbf{q}}_e = \mathbf{q}_d - \dot{\mathbf{q}} \tag{6}$$

$\dot{\mathbf{q}}_e$: Joint speed error vector ($n \times 1$ dimension).
Matching equations (1) and (2), we have:

$$\hat{\mathbf{M}}(q)\left(\ddot{\mathbf{q}}_d + \mathbf{K}_v\dot{\mathbf{q}}_e + \mathbf{K}_p\mathbf{q}_e\right) + \hat{\mathbf{C}}(q,\dot{q}) + \hat{\mathbf{G}}(q) + \hat{\mathbf{F}}(\dot{q}) = \mathbf{M}(q)\ddot{\mathbf{q}} + \mathbf{C}(q,\dot{q}) + \mathbf{G}(q) + \mathbf{F}(\dot{q}) \tag{7}$$

$$\ddot{\mathbf{q}}_e + \mathbf{K}_v\dot{\mathbf{q}}_e + \mathbf{K}_p\mathbf{q}_e = \hat{\mathbf{M}}(q)^{-1}\left(\left(\mathbf{M}(q) - \hat{\mathbf{M}}(q)\right)\ddot{\mathbf{q}} + \left(\mathbf{C}(q,\dot{q}) - \hat{\mathbf{C}}(q,\dot{q})\right) + \left(\mathbf{G}(q) - \hat{\mathbf{G}}(q)\right) + \left(\mathbf{F}(\dot{q}) - \hat{\mathbf{F}}(\dot{q})\right)\right) \tag{8}$$

If estimation errors are low, joint errors approach to a linear equation:

$$\ddot{\mathbf{q}}_e + \mathbf{K}_v\dot{\mathbf{q}}_e + \mathbf{K}_p\mathbf{q}_e \approx 0 \tag{9}$$

The fault tolerant control law used for development of the simulator comprises two classic controllers per each computed torque that are "switched" to reconfigure the fault tolerant controller.

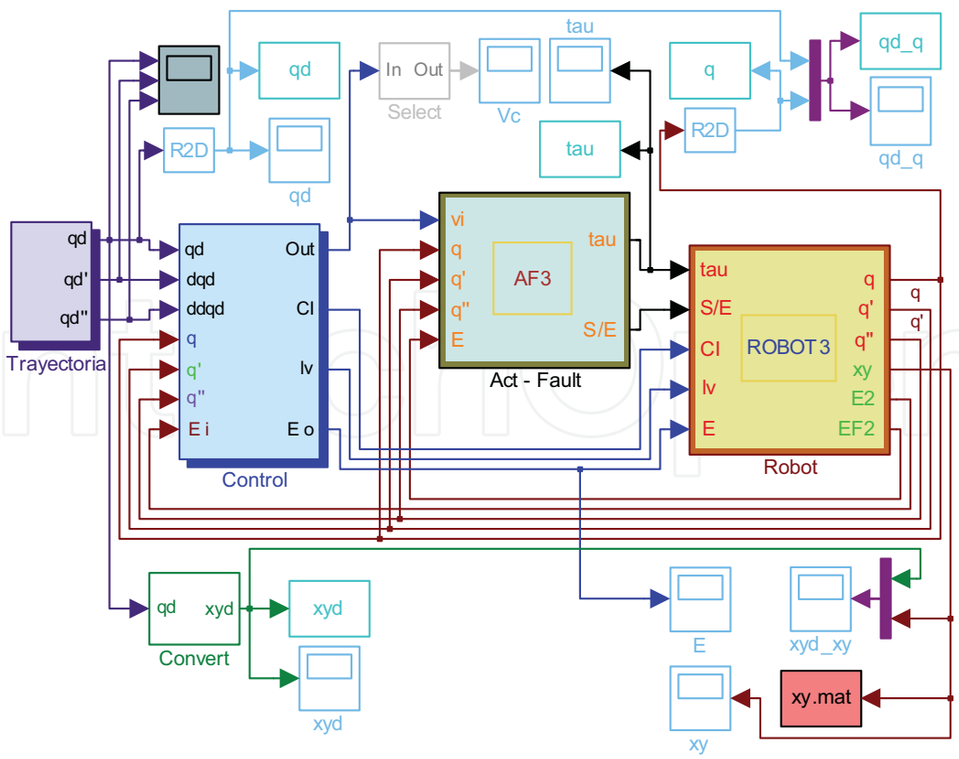


Fig. 6. Simulation diagram of the fault tolerant control system

4. Simulator development

In the development of the simulator, besides the control laws pointed in previous section and the respective manipulator dynamics, actuators dynamics is included corresponding to analogue servomotors with their characteristic position internal feedback. Each model has been carried out using MatLab embedded functions, grouped in distinctive blocks, as follows:

- Trajectory block.
- Control block.
- Actuator-failure block.
- Robot block.

as shown in figure 6.

Control block contains the control laws necessary to provide the manipulator with the ability to follow a desired trajectory, in normal operation and in presence of a failure, reconfiguring the controller in this last case to provide adequate signals to the actuators. In figure 7 we can see an inner view of the control block; this view has been divided in four stages for a better understanding.

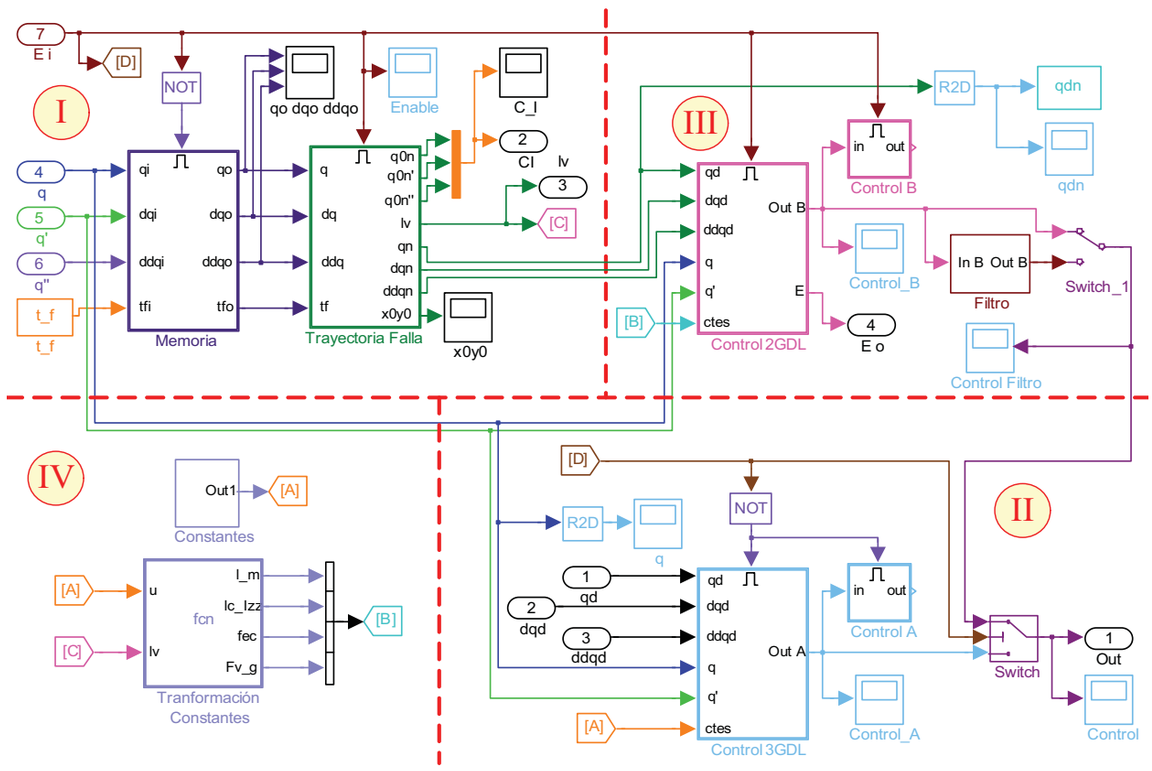


Fig. 7. Simulation diagram including the stages of the control block

Figure 8 shows stage I of the control block, dedicated to memorize the status of: position, speed and acceleration of the joints at the moment when a failure occurs, as well as the time when it happens. With this information, new joint trajectories can be generated.

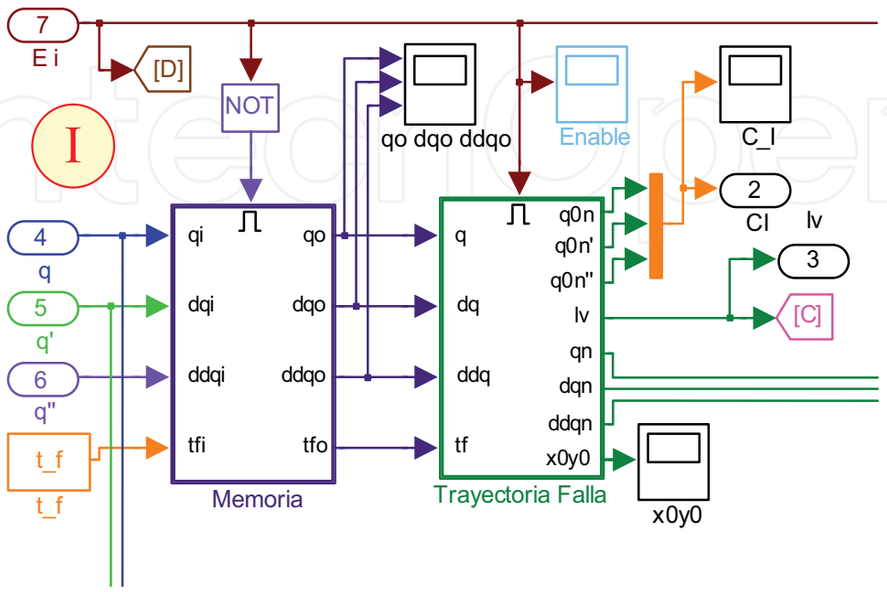


Fig. 8. Simulation diagram: stage I of the control block

Figure 9 shows stage II of the control block, used for processing the signals corresponding to the original desired trajectory.

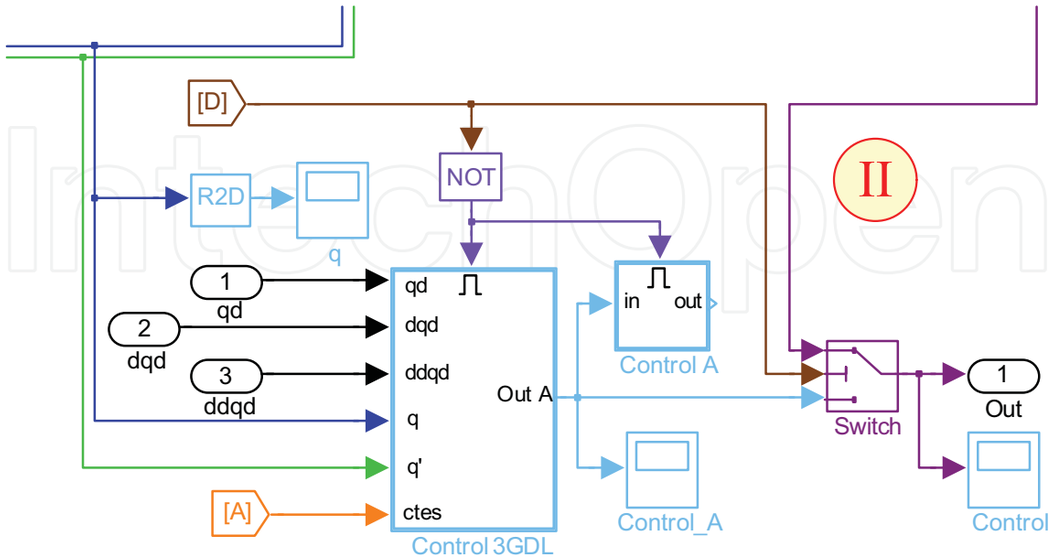


Fig. 9. Simulation diagram: stage II of the control block

The stage III of control block is shown in figure 10. This stage is dedicated to receive the signals from the new joint trajectories generated in stage I after occurring a failure, and to provide the new control signals into the actuators.

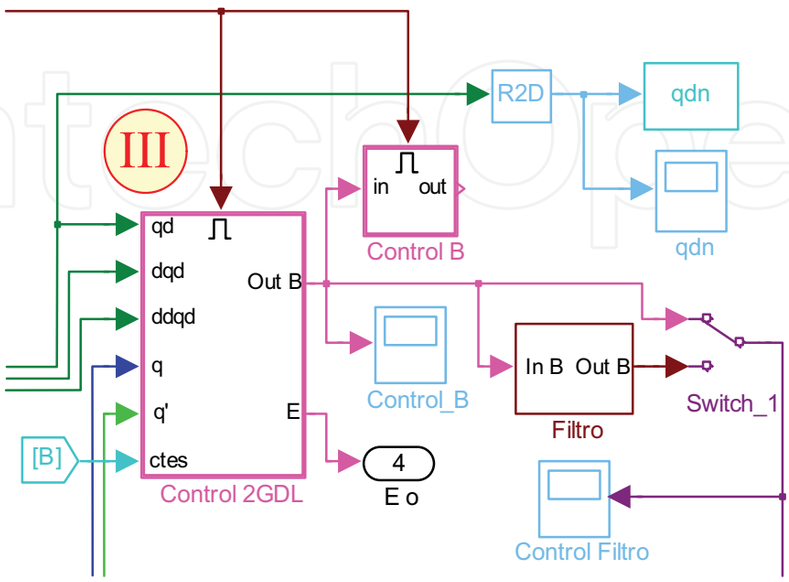


Fig. 10. Simulation diagram: stage III of the control block

When a failure arises in a real manipulator, the properties presented to actuators change, producing new inertias, longitudes, mass centers, etc. In the simulator, this function is carried out in stage IV of the control block, as pointed in figure 11.

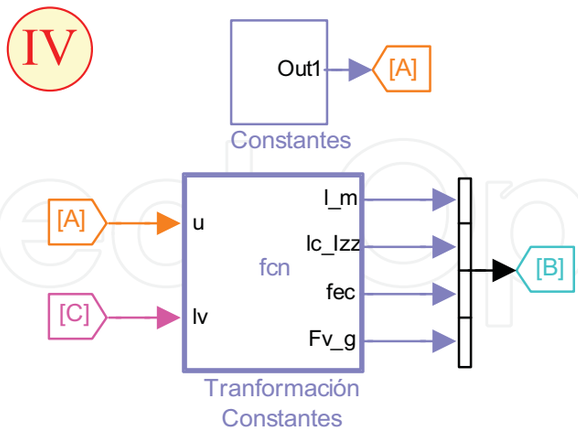


Fig. 11. Simulation diagram: stage IV of the control block

The necessary parameters used in functions developed at stages I to IV, corresponding to the control block, are entered via an interface generated through masking of MatLab embedded functions, this can be seen in figure 12. By means of this interface it is possible to enter the simulator with the controller, robot and failure time parameters.

Fig. 12. Window for entering controller parameters

In figure 13 we have an inside view of the robot block, there we can see the blocks containing the dynamics representing manipulator's non-failure and failure behavior. It is shown; also, a block labeled *Switch-Out*, oriented to the switching function at the moment a failure arises.

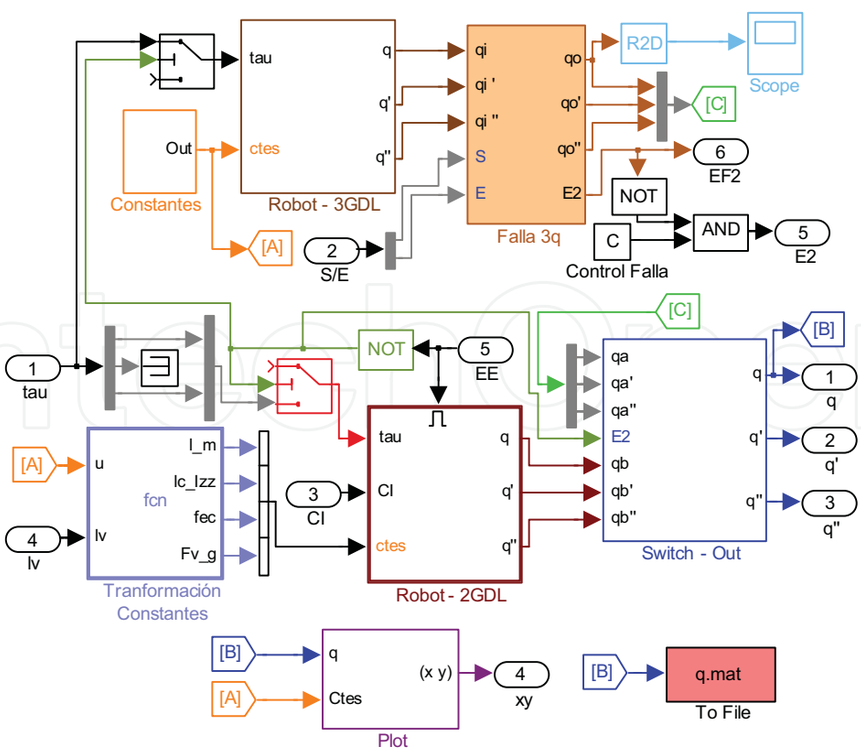


Fig. 13. Simulation diagram corresponding to failing and non-failing manipulator

The interface used for entering parameters of the robot manipulator is shown in figure 14.

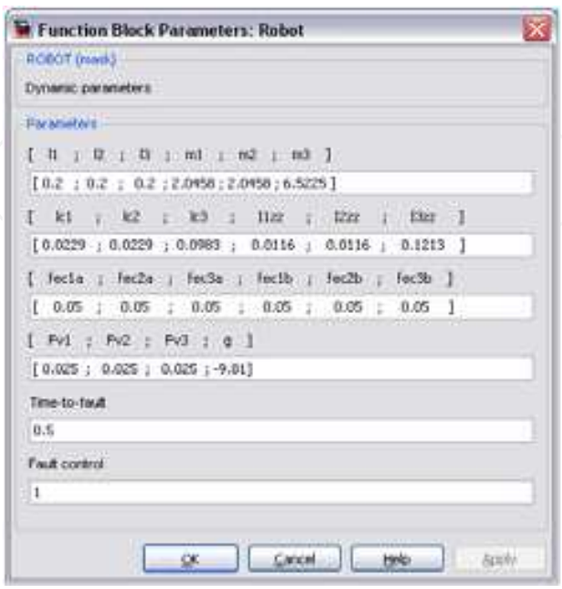


Fig. 14. Window for entering manipulator parameters

The data required for the generation of desired trajectory are charged through "mat" files in the trajectory block, as shown in figure 15.

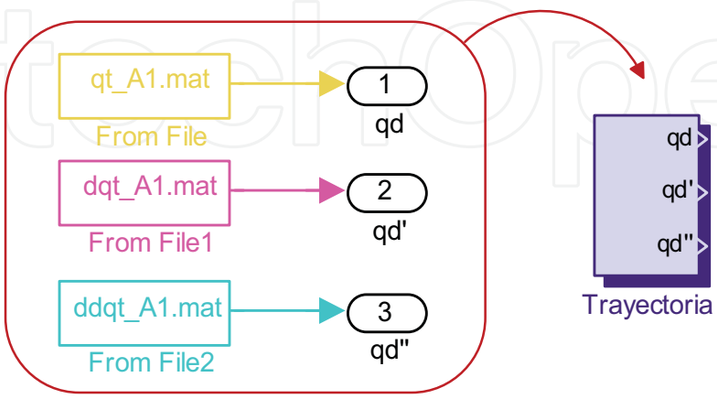


Fig. 15. Simulation diagram corresponding to trajectory generation

Figure 16 shows a view of the interface for analogue servomotor parameter entering.

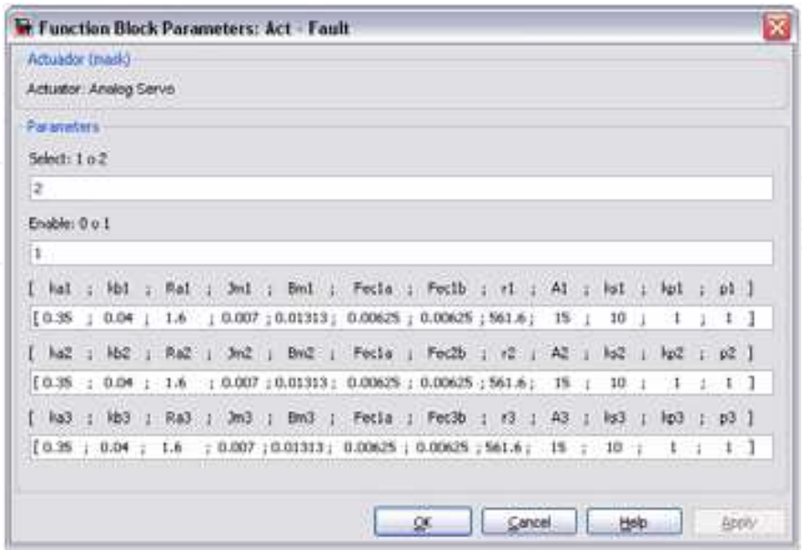


Fig. 16. Window for entering servomotor parameters

Finally, in figure 17 we can see the interface corresponding to the simulation environment for the fault tolerant control system applied on a manipulator. Through this interface, we can summarize the parameter entering windows showed in figures 14 and 16, and we can obtain the respective curves and animations of trajectories for the robot.

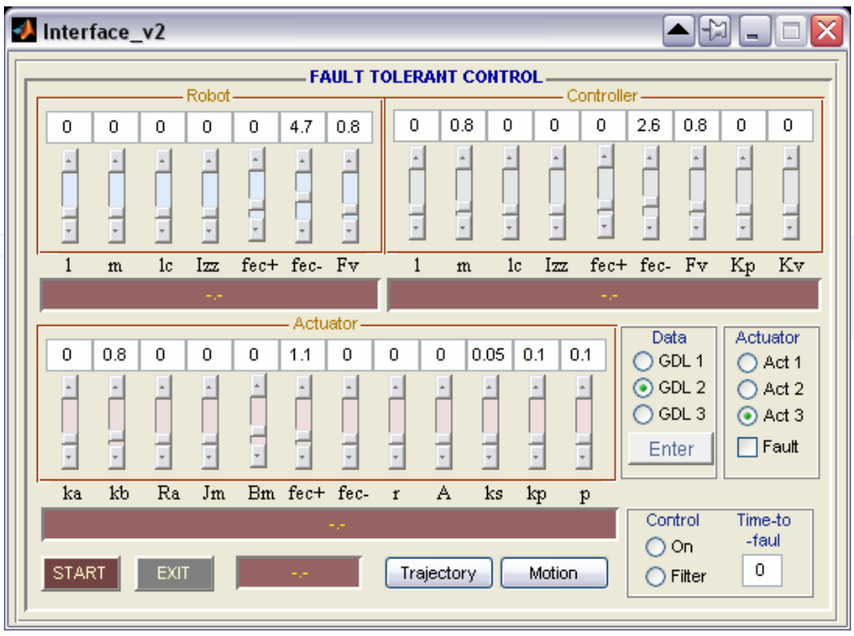


Fig. 17. Simulator user interface for parameter entering and generation of curves and animations

5. Results and conclusions

Here we show results obtained when subjecting the system to a trajectory composed by rectilinear sections. In figure 18 we can see the graphics corresponding to the simulation of trajectory tracking without failures, and in figure 19 a capture of the animation.

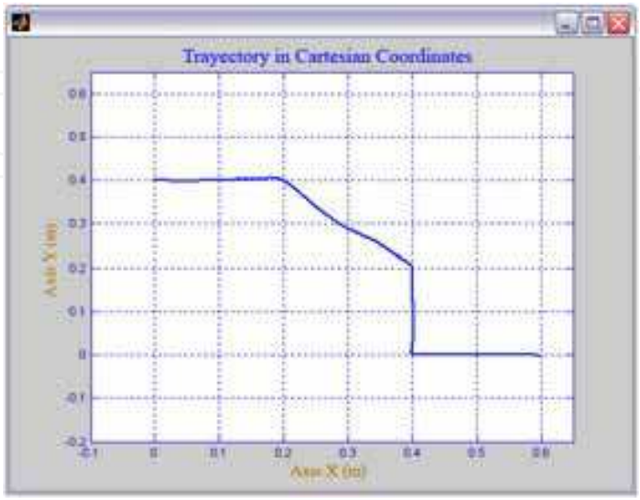


Fig. 18. Trajectory in Cartesian coordinates

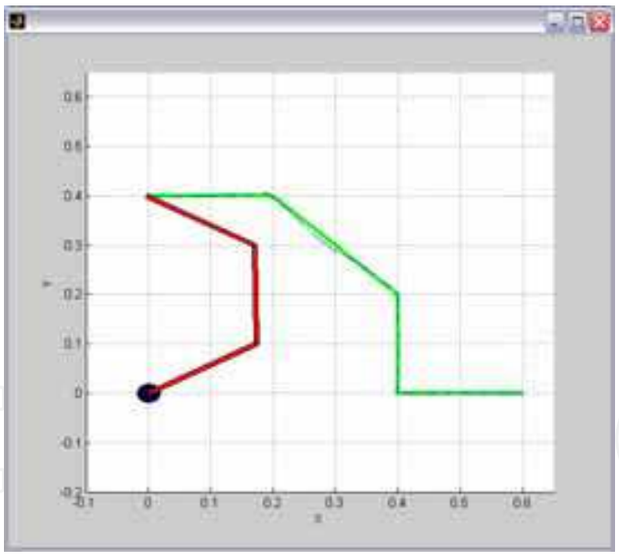


Fig. 19. Animation graphics of the manipulator following a desired trajectory

In figure 20 it is shown the curve corresponding to a simulation of trajectory tracking with a failure occurring in actuator nº 2 at 0.5 sec from initiating trajectory, and in figure 21 a capture of the respective animation, all this under the rule of classic control.

When we subject the manipulator having a failure under the action of the fault tolerant control, the remarkable deviation in Cartesian trajectory produced by classic control (figures 20 and 21) is significantly reduced. This can be seen in the simulation graphics in figure 22, and also in figure 23, showing a capture of the animation.

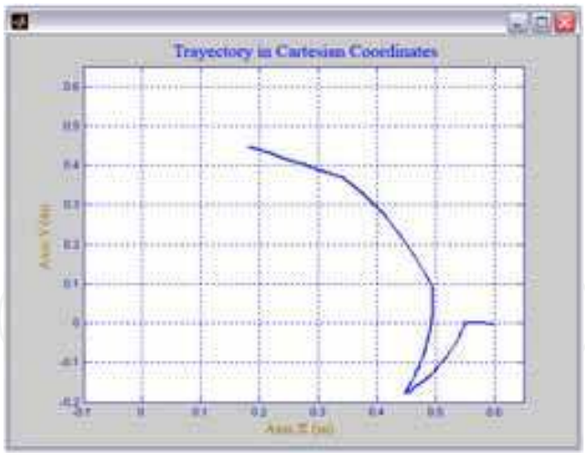


Fig. 20. Trajectory in Cartesian coordinates in presence of failure in an actuator

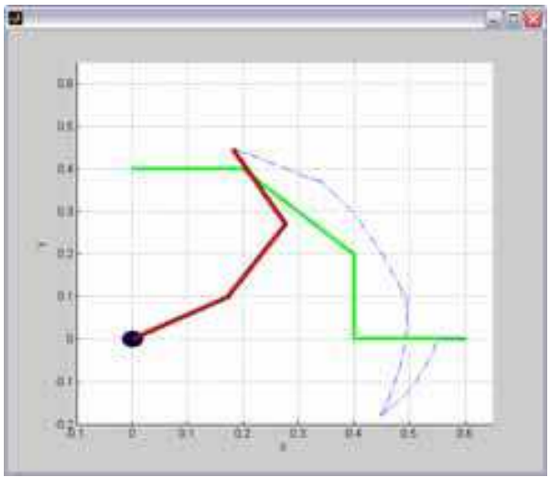


Fig. 21. Animation graphics of the manipulator following a trajectory in presence of failure in an actuator

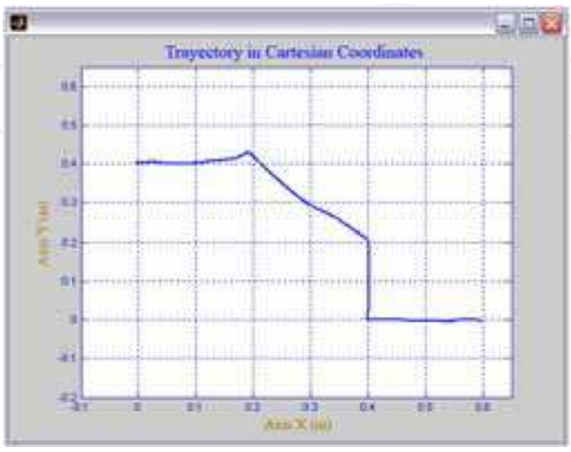


Fig. 22. Trajectory in Cartesian coordinates in presence of failure in an actuator, with failure correction

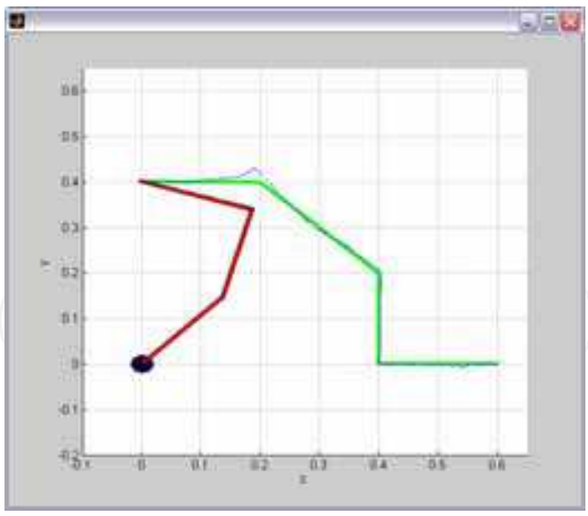


Fig. 23. Animation graphics following a trajectory in presence of failure in an actuator, with failure correction

6. Acknowledgements

This work had the support of the Department for Scientific and Technologic Research of the Universidad de Santiago de Chile, by means of Project 060713UO, Santiago, Chile.

7. Further developments

Thanks to the development of this work, from the implemented simulation tools and the obtained results, fault tolerant control systems essays are being currently carried out, in order to apply them to actual robotic systems, with and without link redundancy, like the SCARA-type robots shown in figure 24 and figure 25, respectively.

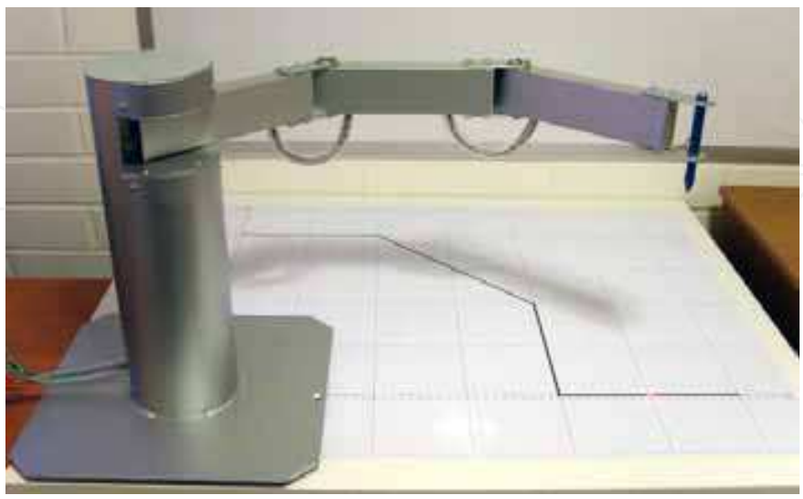


Fig. 24. SCARA-type redundant robot, DIE-USACH

Fig. 25. SCARA-type robot, DIE-USACH

8. References

Blanke, M., Staroswiecki, M. & Wu, E., 2001. *Concepts and Methods in Fault-Tolerant Control*. In Arlington, VA, USA: American Control Conference. Proceedings of the 2001.

Arteaga, F. et al., 2004. Control Tolerante a Fallas Mediante Técnicas de Estabilización Simultanea con Múltiples Dominios de Estabilidad. *Revista Ingeniería UC*, 11(Nº 3), pp.62-69.

Blanke, M., Kinnaert, M., Lunze, J. & Staroswiecki, M., 2006. *Diagnosis and Fault-Tolerant Control 2nd Edition*. New York: Springer-Verlag Berlin Heidelberg.

Blanke, M., Staroswiecki, M. & Wu, E., 2001. Concepts and Methods in Fault-Tolerant Control. In *Arlington, VA, USA: American Control Conference. Proceedings of the 2001.*, 2001. Pearson Education.

Bonivento, C., Isidori, A., Marconi, L. & Paoli, A., 2004. Implicit Fault-Tolerant Control: Application to Induction Motors. *Automatica, vol. 40, Nº 3*, p.355–371.

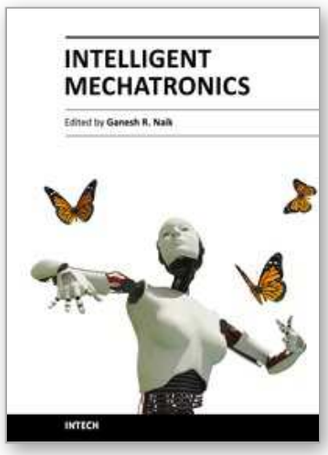
Craig, J., 1988. *Adaptative Control of Mechanical Manipulators*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Kelly, R., Santibáñez, V. & and Loría, A., 2005. *Control of Robot Manipulators in Join Space*. London: Springer-Verlag London Ltd.

Lewis, F., Dawson, D. & Abdallah, C., 2004. *Robot Manipulator Control Theory and Practice*. New York: Marcel Dekker, Inc.

Puig, V. et al., 2004. Control Tolerante a Fallos (Parte I): Fundamentos y Diagnóstico de Fallos. *Revista Iberoamericana de Automática e Informática Industrial*, 1(1), pp.15-31.

Spong, M., Hutchinson, S. & Vidyasagar, M., 2005. *Robot Modeling and Control, First Edition*. New York: John Wiley & Sons, Inc.



Intelligent Mechatronics

Edited by Prof. Ganesh Naik

ISBN 978-953-307-300-2
Hard cover, 248 pages
Publisher InTech
Published online 28, February, 2011
Published in print edition February, 2011

This book is intended for both mechanical and electronics engineers (researchers and graduate students) who wish to get some training in smart electronics devices embedded in mechanical systems. The book is partly a textbook and partly a monograph. It is a textbook as it provides a focused interdisciplinary experience for undergraduates that encompass important elements from traditional courses as well as contemporary developments in Mechtronics. It is simultaneously a monograph because it presents several new results and ideas and further developments and explanation of existing algorithms which are brought together and published in the book for the first time.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen