Research and Development Platform for Multimedia Streaming of MP3 Audio Content

Andrei Novak*, Mircea Stratulat, Daniela Stanescu**, Dan Chiciudean**, Bogdan Ciubotaru**, Razvan Cioarga****

* S. C. Nadcomp S. R. L, Oradea, Romania, Phone: +40 722 17 8222, Fax: +40 359 407445, E-mail: novak_andrei@rdslink.ro

** Software and Computer Engineering Department, Politehnica University of Timisoara, 2, V. Parvan Blvd, 1900 Timisoara, Romania, Phone/Fax: +40 256 40 3260, E-mail: {daniela.stanescu, dan.chiciudean, bogdan.ciubotaru, razvan.cioarga}@ac.upt.ro

Abstract: In the recent years, the MPEG Layer III (MP3) music compression format has become an extremely popular choice for digital audio compression. Its high compression ratio, and near CD quality sound make it a natural choice for storing and distributing music - especially over the internet, where space and bandwidth are important considerations. For example, using MPEG Layer-3 compression, 40 MBytes audio files have been compressed to approximately 3.5 MBytes. As a result of the MP3 popularity, a variety of portable MP3 players entered the market. We decided to design and implement a Hard Disk based MP3 player similar to products currently available (e.g. Creative Labs Nomad, Archos Jukebox 6000, Apple Ipod, etc.). Our goal was to design the player with minimal cost and to implement a FM Stereo Radio Transmitter module for ease of connectivity. This module resolves the compatibility problems with the current available car audio systems. In the same time system flexibility and scalability as well as system evolution to more advanced architectures were the main principles that drove the development of this platform. The primary enhancement of the platform will be to switch the communication module from a analogical FM radio transmitter to digital wired or wireless communication solutions.

Keywords: mp3 player, radio transmitter, hard disk, HD-based players, level shifting, i2c communication

1 Introduction

A digital audio player (DAP) is a device that stores, organizes and plays digital music files. It is more commonly referred to as an MP3 player (because of that format's ubiquity), but DAP's often play many additional file formats. Some

formats are proprietary, such as Windows Media Audio (WMA), and Advanced Audio Codec (AAC). Some of these formats also may incorporate restrictive digital rights management (DRM) technology, such as WMA DRM, which are often part of certain paid download sites. Other formats are completely patent-free or otherwise open, such as Ogg Vorbis, FLAC, Speex (all part of the Ogg open multimedia project), and Module file formats.

There are three main types of digital audio players: MP3 CD Players - These kinds of devices play CD's. Often, they play both audio CD's and homemade data CDs containing MP3 or other digital audio files; Flash-based Players - These are solid state devices that hold digital audio files on internal or external media, such as memory cards. These generally have a low storage device, typically ranging from 128 MB – 4 GB, which can often be extended with additional memory; Hard Drive-based Players or Digital Jukeboxes - Devices that read digital audio files from a hard disk. These players have higher capacities, ranging from 1.5 GB to 100 GB, depending on the hard drive technology. At typical encoding rates, this means that thousands of songs, perhaps an entire music collection can be stored in one MP3 player.

In this paper we will present the implementation of a Hard Drive-based player capable of streaming multimedia audio content via wireless communication interface. This system may be used as a stand alone device for playing mp3 files or as a broadcast audio station.

2 Related Work

The precursors to DAP's were portable CD players and Mini disc players. Non-mechanical DAPs were introduced following the popularity of the precursors.

The first Mp3 player in the world was created by SaeHan Information Systems in 1997. The MPMan F10 was later OEMed to the American market through Eiger Labs. The first non-mechanical digital audio player on the American market was the Eiger Labs MPMan F10, a 32 MB portable that appeared in the summer of 1998. It was a very basic unit and wasn't use expandable.

The first Mp3 player to really make a significant impact on the buying population in America was the Diamond multimedia Rio PMP300. Diamond multimedia wisely released the PMP in September of 1988 just prior to the Christmas season. Sales of the Rio far exceeded the company’s expectations, which led to a number of large companies’ decisions to enter the Mp3 race mp3play.

The first commercially available HD-based Mp3 Player was created by Compaq. It was created as a prototype personal audio appliance by Compaq's Systems Research Center (SRC) and Palo Alto Advanced Development group (PAAD).

The PJB project started in May 1998, and the PJB-100 product shipped in November 1999. This player was succeeded by a lot of players Creative Nomad Jukebox, Archos Jukebox 6000, Apple Ipod, etc mp3hard.

3 System Overview

Inspired by these HD-based players we decided to design a similar one, with minimal cost. We observed that all these players have compatibility problems with the current available car audio systems. The only way to connect them is via the RCA analog input of the audio systems. This is difficult sometimes, because a dashboard disassemble is needed.

We find a solution for this problem, the wireless technology. We implemented a FM Stereo Radio Transmitter to our MP3 player. With this, the player transmits the analog audio signal on FM frequency between 88 - 108 MHz. The only thing needed to connect the player to the audio system is to tune in the correct frequency on the FM radio receiver of the audio system. In this way there is no more need of a wire connection, only needed to keep the player in a range of three meters from the audio system and the signal is transmitted.

The mp3 player is capable to access and read thousands of mpeg digital audio files from a hard disk using a microcontroller. The microcontroller sends the data extracted from a file to mp3 decoder chip. This chip processing the date and send them to the digital analog audio converter (DAC). The DAC converts the digital date from the input and send analog audio signal to the output. That signal is sent to the loudspeakers and to the FM transmitter. We implemented for the player some push buttons and infra red port. With these we will be able to control the mp3 player.

The push buttons are as following: Play (play melody); Stop (stop playing); Next (next melody); Prev (previous melody); Volume + (volume up); Volume - (volume down); Dir (change directory); Shuffle Mode, Normal Mode.

4 Hardware Design

The basic parts used in this project were: Microcontroller - AT90S8535 (Atmel); Mp3 Decoder - STA013 (ST MicroElectronics); 18 bit serial DAC - CS4334 (Crystal/Cirrus Logic); FM Stereo Radio Transmitter - BA1404 (frequency 88-108 MHz). We used the ATMEL AT90S8535 microcontroller to control the components and data flow. The AT90S8515 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful

instructions in a single clock cycle, the AT90S8515 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The AT90S8515 provides the following features: 8 K bytes of In-System Programmable Flash, 512 bytes EEPROM, 512 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, flexible timer/counters with compare modes, internal and external interrupts, a programmable serial UART, programmable Watch-dog Timer with internal oscillator, an SPI serial port and two software selectable power saving modes. The Idle Mode stops the CPU while allowing the SRAM, timer/counters, SPI port and interrupt system to continue functioning. The power down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next external interrupt or hardware reset.

4.1 Microcontroller Connection

Figure 1 presents, the microcontroller connection circuit. The JP2 connector, labeled V+OUT, connecting the microcontroller Vcc pin to the voltage regulator circuit. This circuit is contains two kind of voltage regulators LM7805 and LM317T.

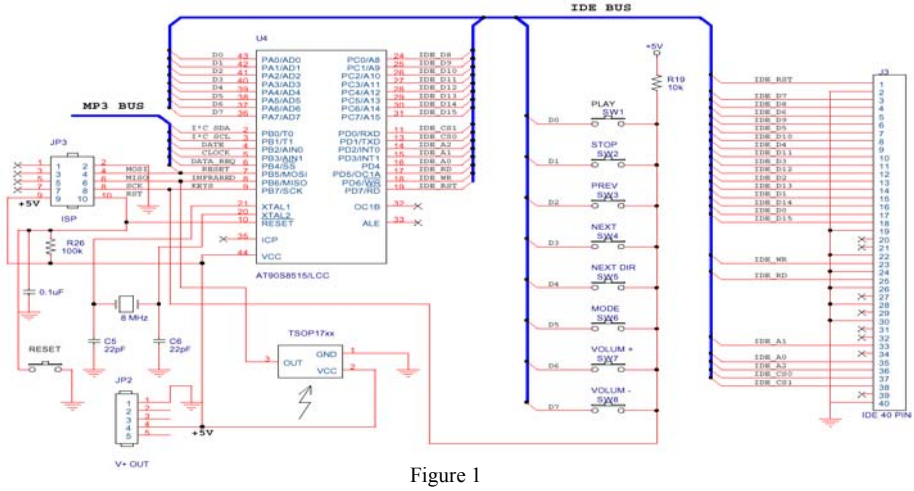


Figure 1
Microcontroller connections

With the LM7805 we obtained the 5V+ Vcc, necessary the microcontroller and the hard disk. With the LM717T we have the 3V+ for MP3 Decoder, DAC and the FM radio transmitter. We choice for microcontroller a crystal at 8 MHz, this will let as to use the microcontroller at his maximum capacity, 8 MIPS (eight thousands of instruction per second). This crystal is used to give a clock for microcontroller, by giving an oscillation between XTAL1 and XTAL2. To obtain that oscillation we also need a pair of capacitors rated between 10 pF and 100 pF, the value depending only from crystal sensitivity.

To program the microcontroller we need a programmer. We use a serial programmer similar to the STK200 from Kanda Systems that must be connected to the PC parallel port and to the microcontroller. To connect the programmer to microcontroller we used the JP3 connector, labeled ISP.

Five pins of the microcontroller are used for serial programming, MOSI (Master data output, slave data input), MISO (Master data input, slave data output pin), SCK (CLOCK), RESET and of course the GROUND.

Here we had to solve a problem. The normal state of the RESET line is high, then microcontroller executes the instructions from the flash and the programming mode is represented by the Low state of the RESET. Therefore we need to keep the RESET line High until a programming is started. When a programming is started the programmer must have to pull the RESET to LOW. We used a 100KΩ resistor connected with one end to the 5 V+ and the other end to the microcontroller RESET pin to make this pull up and down. When no signal came from the programmer on the RESET line, the RESET is HIGH with the Vcc and when a low is added from the programmer the RESET pin commute to LOW.

The same principle is used for the push buttons, when a button is pushed the pin that correspond to the push button there is HIGH and the 9 pin labeled KEYS is LOW, otherwise KEYS is HIGH and no signal is add on D0-7 pins.

The first eight data lines of the hard disk are connected to Port A (PA0-7) and the second eight lines (D8-15) are connected to Port C (PC0-7).The other lines needed from hard disk is connected to Port D.

For reception of the infrared signal we use a TSOP1738. This chip needs only a 5 V Vcc to pin 2 and it captures all infrared signals and send it to the microcontroller via the pin 2.

4.2 Mp3 Decoder Connection

The STA013 is a fully integrated high flexibility MPEG Layer III Audio Decoder, capable of decoding Layer III compressed elementary streams, as specified in MPEG 1 and MPEG 2 ISO standards. The device decodes also elementary streams compressed by using low sampling rates, as specified by MPEG2.5.

STA013 receives the input data through a Serial Input Interface. The decoded signal is a stereo, mono, or dual channel digital output that can be sent directly to a D/A converter, by the PCM Out-put Interface. This interface is software programmable to adapt the STA013 digital output to the most common DAC's architectures used on the market.

Figure 2 represent the connection circuit of the STA013 to the microcontroller and to the DAC. In this project we have two chips that work with different voltages. The STA013 is a 3 volt chip. The data sheet claims it can run between 2.7 to 3.6 volts. It must not be used at 5 volts. We need a level-shifting to connect the STA013 to the microcontroller.

The communication between the STA013 and the microcontroller is made via the MP3 BUS. This bus incorporates a I²C bus and a serial data bus. The I²C bus has two lines, the DATA line (SDA) and the CLOCK line (SCL), both need level-shifting because the SDA there is a bidirectional line and the SCK there is an input line. The serial bus has four lines DATA, CLOCK, DATE_REQ and RESET. The DATE_REQ is an output line and the other three are input lines, and they need level-shifting.

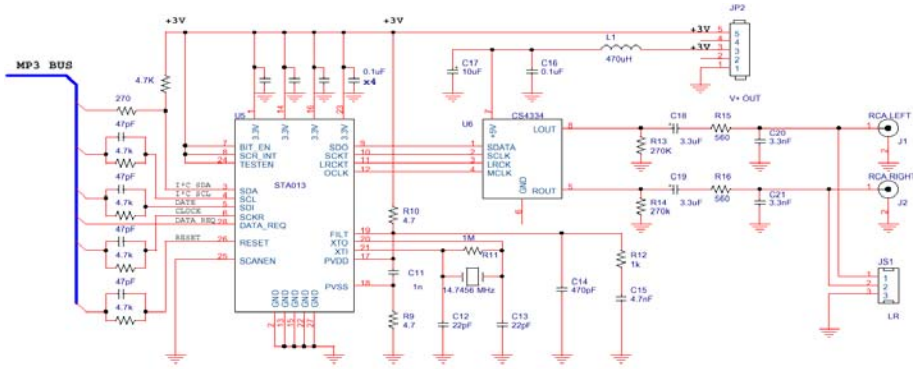


Figure 2
STA013 connections

4.3 Level-shifting

The outputs from a 5 volt powered chip must not be directly connected to the STA013 input pins. The STA013 inputs are not 5 volt tolerant. Each pin has a pair of input protection diodes. These diodes may conduct a small current. The simplest and easiest way to interface a 5 volt output to the STA013 input pins is with a series resistor, which will limit the current when the 5 volt output is high. There is some input capacitance (3.5 pF) on the input pins, so adding a small

capacitor in parallel with the resistor will allow the rapid edge to properly drive the input pin. The value of the resistor and capacitor are not critical, Figure 3 shows 4.7 K and 47 pF, which limits the steady-state current to less than 1/2 mA, and has been tested. The capacitor is not really necessary if the pin doesn't need to be driven rapidly, such as the RESET pin.

Connecting the SDA signal from a 5 volt microcontroller to the STA013 is a bit more complex, though a simple circuit can often be used. The SDA line is bidirectional, which either device can pull down, or a resistor provides the pull-up. Most microcontrollers have TTL thresholds, so a pull-up to the 3 volt supply will easily satisfy the 2.0 volt input high requirement. If the microcontroller will never drive the line high (only pull low or tri-state), then no other parts are probably required. This may be the case if the microcontroller has a dedicated I²C pin. In most cases, the microcontroller can drive the line high, and an additional resistor should be added to prevent damage to the STA013.

This current limiting resistor should be small, as it will form a resistor divider with the pull-up when the microcontroller drives low. If the pin can be put in a tri-state mode, the firmware should be written to use the tri-state for a logical high, or at least tri-state the SDA signal when not transmitting data, to save power.

The connecting of the DAC CS4334 is very simple. We only have to connect the SDATE, SCLK, LRCK, and MCLK to the same pins on the STA013. This DAC does not need a sophisticated output filter, because a filter is integrated in the chip.

4.4 FM Stereo Radio Transmitter

For the FM Stereo transmitter we use a BA1404. This device contains a stereo modulator, an FM modulator, and an RF amplifier. The stereo modulator creates a stereo composite signal, witch consists of a main (L+R), sub (L-R) and pilot signals, from a 38 KHz quartz controlled frequency. The FM modulator oscillates a carrier in the FM broadcast band (76 to 108 MHz) and modulates it with the composite signal. The RF amplifier creates energy to emit the modulated FM signal. It also functions as a buffer for the FM modulator. The schematic of this transmitter is very simple and it is taken from the original date sheet of the BA1404, see Figure 3.

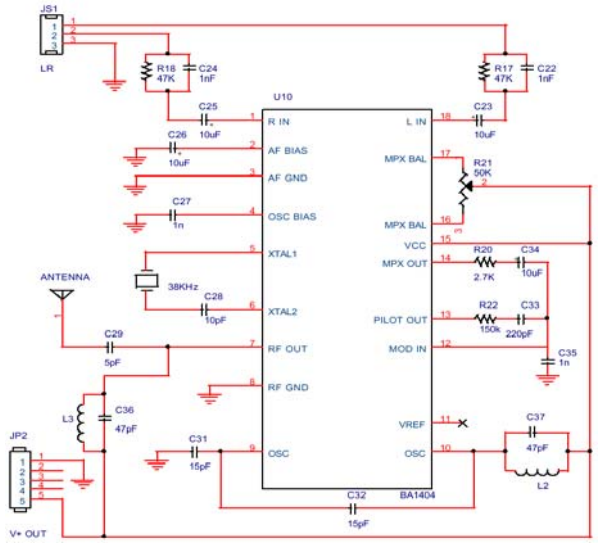


Figure 3
FM Stereo Radio Transmitter

5 Communication and Configuration via I²C

I²C is a 2-wire protocol, created by Philips. One line acts as a clock (SCL) and the other data (SDA). The protocol defines the ability to have multiple masters initiate communication, but here we'll only worry about the simple and common case where the microcontroller is the only device that controls the bus, and all the other chips (like the STA013) respond to queries initiated by the microcontroller.

The STA013 requires initialization by I²C communication. This step can not be avoided. One part of the required initialization is to sent a group of 2007 writes provided by ST in the file p02_0609.bin.

For simple applications like using the STA013, there are four fundamental operations. Only these four operations are needed to build routines that access the STA013 chip:

Start Condition: This is a high-to-low transition of the SDA line, while SCL is high. Normally SDA only changes when SCL is low. When SDA changes while SCL is high, it means either the start or stop of a communication, instead of data transfer.

Send A Byte, Get ACK Bit: Eight bits are sent by the microcontroller, each write to SDA occurs while SCL is low. A ninth clock is given and the microcontroller

receives an ACK bit from the STA013. If the STA013 received the byte, it will send a zero in this bit.

Receive A Byte, Send ACK Bit: The microcontroller gives eight clocks on SCL, and after each low-to-high clock transition, a bit is read from the STA013. During a ninth clock, the microcontroller pulls SDA low to acknowledge that it received the byte.

Stop Condition: This is a low-to-high transition of the SDA line, while SCL is high. After the stop condition, both lines are left high, which is the idle state of the I²C bus.

Using these four basic I²C operations, a function to read from the STA013 can be built as follows:

Start Condition; Send A Byte: The value is 0x86 (134). The seven most significant bits instruct the STA013 to listen (because there may be other chips connected to SDA and SCL). The LSB is clear, telling the STA013 that will be writing an address; **Send A Byte:** The value is the address where we need to read data. The main program will pass the address to our sta013_read function; **Stop Condition; Start Condition; Send A Byte:** The value is 0x87 (135). Again, the upper bits select the STA013, and the LSB sets up the next access to read;

Receive A Byte: Read the byte from the STA013. This will be returned to the main program; **Stop Condition**;

Writing to the STA013 is even easier. Here are the steps: **Start Condition; Send A Byte:** The value is 0x86 (134). The seven most significant bits instruct the STA013 to listen. The LSB is clear, telling the STA013 that will be writing; **Send A Byte:** The value is the address within the STA013 where we write the data. The main program will pass the address to our sta013_write function; **Send A Byte:** The value is the data passed from the main program to write into the STA013; **Stop Condition;**

The first step should be to check that the STA013 is actually present. Just read from address 0x01. If the read routine returns with an error, then no device sent an ACK bit and there is no chip installed. If there is an ACK, the data returned should always be 0xAC. Any other value means that the STA013 isn't working properly.

The next step is to transmit the 'p02_0609.bin' configuration file provided by ST. This file consists of 2007 address and data pairs. Sending the file is simply, we just write a loop that passes each pair to the 'sta013_write' function. Each ACK should be checked and the process aborted if any write doesn't receive any of its ACK's.

Once the configuration file is sent, the board specific settings must be sent. These settings are needed to setup the crystal and the CS4331 DAC. To configure the

STA013 for a crystal, in our case a 14, 74568 MHz crystal, we need a little program from ST, named ConfigurPLL version 1.0.

After that only three pairs of bytes are needed, two of them set the STA013 for the DAC. Address 0x84 represents PCMDIVIDER register and it is set to 1 for 256X over sample, 32 bit words (allows 24 bit output).The next address 0x85 represents PCMCONF register and it is set to 33 for I²C format. The last is address 0x24; witch is used to enable the DATE_REG pin by setting it to 4. Now the STA013 is set up for playing MP3, is only needed a Run Condition (address 0x72, data 0x01) to be ready and a Play Condition (address 0x13, data 0x01) to start playing the mp3. A full register description can be found in the original data sheet for the STA013.

6 System Extension

The system presented above provide a certain degree of flexibility and compatibility based on its FM radio transmitter that can broadcast audio content to FM enabled devices. But this solution has certain limitations and drawbacks in the context of the evolution of digital communications and multimedia broadcast solutions over digital communication link. According to this trend in the field of communication we present a different approach for our multimedia broadcast system. The system will be equipped with digital wired and wireless communication interfaces that will enable this platform to serve as a research and development support system for multimedia applications. This system will be suited for applications requiring mobility not only for the client devices but also for the server module. In the following pictures we will present the architecture of the both implementations, the FM radio transmitter based implementation and the digital communication enabled platform. In Figure 4 the block diagram of the system implemented based on the FM radio transmitter is presented. This solution in simple and can be produced at low costs because of the simplicity and low performance requirements for the components as well as because of the simple design and simple software components. But the drawback of this platform is its communication solution.

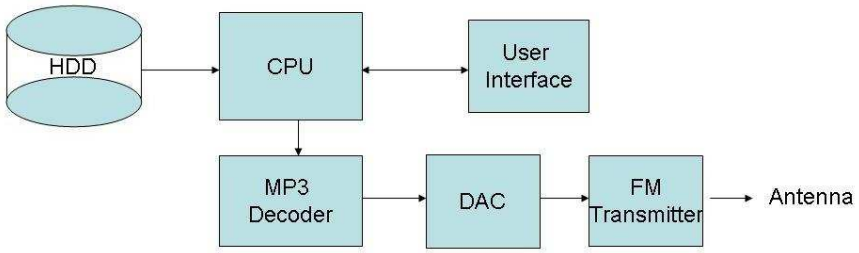


Figure 4
FM radio transmitter based solution

In Figure 5 we present the advanced architecture which involves digital communication interfaces that bring more complexity to the system but in the same time make it more flexible and with an enhanced research and application development possibilities. This solution gives more performance and is oriented on the technology trends in the field of multimedia broadcasting. In the same time the solution is more expensive due to the complexity of the communication interface, the performance requirements for the CPU which has to be faster that the one used in the other implementation. The software component is more complex as well because of the control components (drivers) that have to be implemented and the protocol stack.

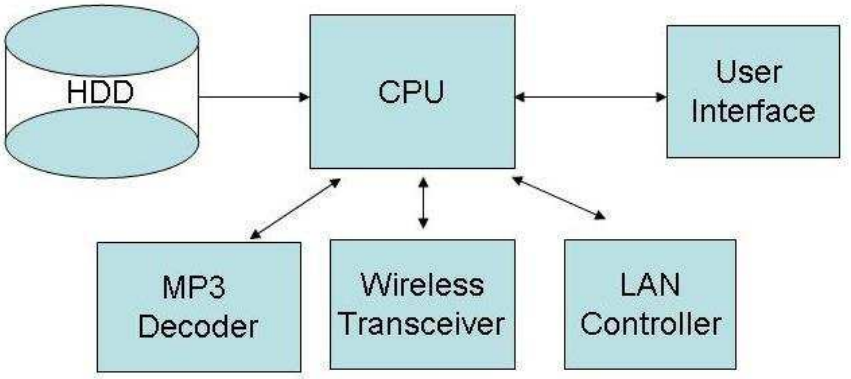


Figure 5
Digital communication based solution

Conclusions

The system presented in this paper represents the result of a research and development activity. Several similar solutions had to be studied as well as all the devices and circuits used for the development of this system. The challenges encountered by the development team were interfacing with storage devices and wireless communication module. This system be further developed as a commercial product or can be used as experimental test bed for didactic purpose. As future work, the communication module will be further developed in order to enable more complex wireless connectivity for the system and in the same time the storage module will be developed to permit connectivity with several other media storage systems. These two platforms can be used as experimental platforms for studies in the field of multimedia application involving content broadcast (or unicast) and development platforms for product implementations.

References

[1] Barry B. Brey: Architecture, Programming and Interface, Volume Fifth Edition, 2000

[2] Atmel Corporation: Atmel 8-bit avr risc microcontroller ats908515. Datasheet from: http://www.atmel.com, 1999

[3] Cirrus Logic Crystal Department: 8 pin stereo d/a converter for digital audio cs4334. Datasheet from: http://www.cirrus.com, 1997

[4] William Kleitz: Digital Electronics, A Practical Approach. 2002

[5] ROHM CO LTD: Fm stereo trasmitter ba1404. Datasheet from: www.DatasheetCatalog.com, 1997

[6] Muhammed Ali Maziali and Gillispie Maziali: Assembly Language, Design, and Interfacing, Vol. 1&2, 2000

[7] ST Microelectronics: Mpeg 2.5 layer iii audio decoder. Datasheet from: http://www.st.com/, 1998

[8] ST Microelectronics: Sta configuration file. p02 0609.bin from: http://www.st.com/, 1998

[9] ST Microelectronics: 1.2v to 37v voltage regulator lm317t. Datasheet from: http://www.st.com/, 2003

[10] PJRC: How to use the sta013 mp3 decoder chip. http://www.pjrc.com/tech/mp3/sta013.html, 2003

[11] The Rockbox: The history of hd-based mp3 players. http://www.rockbox.org/playerhistory/, 2004

[12] Vishay Telefunken Semiconductor: Photo modules for pcm remote control systems tsop1738. Datasheet from: www.DatasheetCatalog.com, 2001

[13] Kanda Systems: Stk-200 avr programmer. http://www.kanda.com/index.php3?cs=1&bc=direct&bw=%2Fstk200.html, 2998

[14] Unisonic Technologies: 3-terminal 1a positive voltage regulator lm7805. Datasheet from: www.datasheetcatalog.com, 2001

[15] John F. Wakerly: Digital Design Principles and Practices. 2000

[16] The Free Encyclopedia Wikipedia: Digital audio player. http://en.wikipedia.org/wiki/Digital audio player, Jan 2006

A Study of Sequence Clustering on Protein's Primary Structure using a Statistical Method

Alina Bogan-Marta

Faculty of Electrotechnics and Information Technology
University of Oradea
Universităţii 1, 410087 Oradea, Romania
e-mail: alinab@uoradea.ro

Nicolae Robu

Faculty of Automation and Computer Science and Engineering
"Politehnica" University of Timişoara
Vasile Pârvan No. 2, 300223 Timişoara, Romania
e-mail: nicolae.robu@rectorat.utt.ro

Abstract: The clustering of biological sequences into biologically meaningful classes denotes two computationally complex challenges: the choice of a biologically pertinent and computable criterion to evaluate the clusters homogenity, and the optimal exploration of the solution space. Here we are analysing the clustering potential of a new method of sequence similarity based on statistical sequence content evaluation. Applying on the same data the popular CLUSTAL W method for sequence similarity we contrasted the results. The analysis, computational efficiency and high accuracy of the results from the new method is encouraging for further development that could make it an appealing alternative to the existent methods.

Keywords: biological sequence, n-grams, entropy, dissimilarity matrix, exploratory data analysis

1 Introduction

In bioinformatics, **sequence clustering** algorithms attempt to group sequences that are somehow related. Generally, the clustering algorithms are single linkage clustering, constructing a transitive closure of sequences with a similarity over a particular threshold. The similarity score is often based on sequence alignment. Most of the time, sequence clustering is used to make a non-redundant set of

representative sequences [1] and sequence clusters are often synonymous with (but not identical to) protein families. Determining a representative structure for each *sequence cluster'* is the aim of many structural genomics initiatives [1]. The general purpose of grouping proteins into families leads to more sensitive detection of new members and improved discrimination against spurious hits based on the essential conserved features in a family [2].

The most obvious measure of the similarity (or dissimilarity) between two samples is the distance between them. One way to begin a clustering investigation is to define a suitable metric and compute the matrix of distances between all pairs of samples. If distance is a good measure of dissimilarity, then one would expect the distance between samples in the same cluster to be significantly less then the distance between the samples in different clusters [3].

There are numerous algorithms and associated programs to perform cluster analysis, for example, hierarchical methods [4], self-organizing maps [5], k-means [6], and model-based approaches [7], [8], [9]. Existing clustering approaches that have been applied to biological sequences, mostly proteins, are reviewed in [2]. Many of them are based on manual or semi-manual procedures; others are fully automatic but less reliable. To our knowledge, there is no generally accepted method that is able to produce automatically an accurate clustering of a large biological sequence database. Conventional clustering algorithms employ distance (or similarity) measure to form the clusters [9] when graph partitioning algorithms exploit the structure of a graph to find highly connected objects. Hence, the biologist wishing to perform cluster analysis is faced with a dyzzing array of algorithmic choices and little basis on which to make a choice.

Having proposed a new similarity measure for protein sequences in a previous work [11] we come here to analyse it in clustering process. The new method is based on Markov chains representation known as *n*-gram in statistical language modeling. A similarity measure estimation derived from cross entropy was adopted from information theory field in order to compute the similarity between the resulting *n*-grams. The new strategy was applied for the task of clustering protein sequences using a geometrical representation, based on the dissimilarity matrices derived from sequence comparisons within two different databases of proteins. On the largest experimetal database we apply the similarity method used by CLUSTAL W. It is one of the most popular tools (freely available at http://www.ebi.ac.uk/clustalw/), based on a multiple sequence alignment strategy. We are using this tool in order to compare the correlation values between the clusters obtained using the similarity method of CLUSTAL W with those using the new statistical method.

2 Method

2.1 The New Statistical Similarity Method

Protein sequences from all different organisms can be treated as texts written in a universal language in which the alphabet consists of 20 distinct symbols, the amino-acids. The mapping of a protein sequence to its structure, functional dynamics and biological role then becomes analog to the mapping of words to their semantic meaning in natural languages. This analogy can be exploited by applying *statistical language modeling* and *text classification techniques* for the advancement of biological sequences understanding. Scientists within this hybrid research area believe that the identification of Grammar/Syntax rules could reveal entities/relations of high importance for biological and medical sciences.

In the presented method, we adopted a Markov-chain grammar to build for our protein dataset *2*-gram, *3*-gram and *4*-gram models. To clarify things we chose a hypothetical protein sequence WASQVSENR. In the *2*-gram modeling the available tokens/words were {WA AS SQ QV VS SE EN NR}, while in the *3*-gram representation they were {WAS ASQ SQV QVS VSE SEN ENR}. Based on the frequencies of these tokens/words (estimated by counting) and by forming the appropriate ratios of frequencies, the entropy of an *n*-gram model can be readily estimated using (1) as comes from Van Uytsel and Compernolle's work [12].

$$\hat{H}_L(X) = -\frac{1}{N}\sum_{W^*} Count\left(w_1^n\right)\log_2 p_L\left(w_n \middle| w_1^{n-1}\right) \quad , \tag{1}$$

where the variable X has the form of an *n*-gram $X = w_1^n \Leftrightarrow \{w_1, w_2, \ldots, w_n\}$ and $Count(w_1^n)$ is the number of occurrences of w_1^n. The summation runs over all the possible *n*-length combinations of consecutive *w* (i.e. $W^*=\{\{w_1,w_2,\ldots,w_n\},\{w_2,w_3,\ldots,w_{n+1}\},\ldots.\}$) and *N* is the total number of *n*-grams in the investigated sequence. The second term, $p\left(w_n \middle| w_1^{n-1}\right)$, in (1) is the conditional probability that relates the *n*-th element of an *n*-gram with the preceding *n*-1 elements. Following the principles of maximum likelihood estimation (MLE) [13], it can be estimated by using the corresponding relative frequencies:

$$\hat{p}\left(w_n \middle| w_1^{n-1}\right) = \frac{Count\left(w_n\right)}{Count\left(w_1^{n-1}\right)}. \tag{2}$$

This measure is indicative about how well a specific protein sequence is modeled by the corresponding *n*-gram model. While this measure could be applied to two distinct proteins (and help us to decide about which protein is better represented by the given model), the outcomes cannot be used for a direct comparison of them. Thus, the common information content between two proteins *X* and *Y* is expressed via the formula:

$$E(X,Y) = -\sum_{all\ w_1^n} P_X(w_1^n) \log P_Y\left(w_n \mid w_1^{n-1}\right) \tag{3}$$

The first term $P_X\left(w_1^n\right)$ in (3) corresponds to the reference protein sequence *X* (i.e. it results from counting the words of that specific protein). The second term corresponds to the sequence *Y* based on which the model has to be estimated (i.e. it results from counting the tokens of that protein). Variable w_1^n ranges over all the words (that are represented by *n*-grams) of the reference protein sequence.

2.2 Sequence Comparison Strategies with the New Similarity Method

Having introduced the new similarity measure, we proceed here with the description of its use in order to perform comparisons within protein databases. The essential point of our approach is that the compared proteins in a given database (containing annotated proteins with known functionality, structure etc.) are represented via *n*-gram encoding and the above introduced similarity is utilized to compare their representations.

We considered two different ways in which the *n*-gram based similarity is engaged in efficient database searches. The most direct implementation is called hereafter as *direct method.* A second algorithm, the *alternating method*, was devised in order to cope with the fact that the proteins to be compared could be of very different length. It is easy to observe the need of having two methods if sequences of very different length are compared. The procedure of experimenting with both methods and contrasting their performances gave the opportunity to check the sensitivity of the proposed measure regarding the length of the sequences.

Direct Method

Let S_q be the sequence of a query-protein and $\{S\}=\{S_1, S_2, \ldots S_N\}$ the given protein database. The first step is the computation of 'perfect' score (PS) or 'reference' score for the query-protein. This is done by computing $E(S_q,S_q)$ using the query-protein both as reference and model sequence (we call here 'model' the sequence compared with the query) in equation (3). In the second step, each protein S_i, i=1…N, from the database serves as the model sequence in the computation of a similarity score $E(S_q,S_i)$, with the query-protein serving as reference sequence. In this way, N similarities are computed $E(S_q,S_i)$, i=1,..,N. Finally, these similarities are compared against the perfect score PS by computing the absolute differences $D(S_q,S_i)=|E(S_q,S_i)–PS|$. The 'discrepancies' in terms of information content between the query-protein and the database-proteins are expressed. By ranking these N measurements, we can easily identify the most similar proteins to the query-protein as those which have been assigned the lowest distance $D(S_q,S_i)$.

Alternating Method
The only difference with respect to the direct method is that when comparing the query-protein with those from the database, the role of reference and model protein can be interchanged based on the shortest (the shortest sequence plays the role of reference sequence in (3)). The other steps, perfect-score estimation, ranking and selection, follow as previously.

3 Experiments

3.1 Sequence Databases

The proposed strategy based on measuring protein similarity was demonstrated and validated using two experimental databases. A small one, contains an overall sample of 100 protein sequences where two distinct groups of protein data had been selected as follows. The first 50 entries of the database correspond to proteins selected at random from the NCBI public database [14]. The last 50 entries corresponds to proteins resulted from different mutations of the p53 gene. The mutations were selected randomly from the database we created using the descriptions provided by the International Agency for Research on Cancer (IARC) Lyon, France [15]. This set of 50 proteins, denoted hereafter as p53-group, is expected to form a tight-cluster of textual-patterns in the space of biological semantics. On the contrary, the rest 50 proteins should appear as textual-patterns in the same space that differ not only from other, but also (and mainly) from the p53-group. It could be formulated as the problem of two class recognition.

The second database is a set of 1460 proteins extracted from Astral SCOP 1.67 sequence resources [16]. From the available/original corpus of data, which is a structured one, only those families containing at least 10 protein sequences were included in our new database. In this way, 31 different families unequally populated were finally included. We mention that the annotation of our database follows the original annotation relaying on the biological meaning of similarity concept (and therefore can be considered as providing the 'ground-truth' for the protein classification). As in the small database set, we expected that all the proteins belonging to the same family would appear as a tight cluster of textual patterns and having a proper similarity measure so as we could differentiate the existent families.

This database (of 1460 proteins) was organized at random in 3 different sets with less than 500 sequences, in order to observe at a smaller scale the behavior of the applied similarity technique.

3.2 Results

The geometrical consideration, according to which the patterns are represented by points (i.e. the end tails of corresponding vectors), in a multidimensional space, is very useful in order to conceptualize morphological relationships between patterns, to search for natural groupings inside the sample patterns, etc. The key idea is that similar patterns are mapped onto nearby points [17].

In order to validate the two variants of the strategy we proposed, are followed some classical steps of *Exploratory Data Analysis*. Generating the procedure of similarity search between the sequences in each data set we have, we built the corresponding dissimilarity matrix (that comes from N×N comparisons) used by the representation technique to illustrate the geometrical distribution of our data. In Figures 1, 2, 3 and 4 the matrix containing all the possible dissimilarity measures $D(S_i,S_j)$, $i,j=1,2,\ldots N$ is depicted as a grey scale image, for both algorithmic variants of our method and three different *n*-gram models.

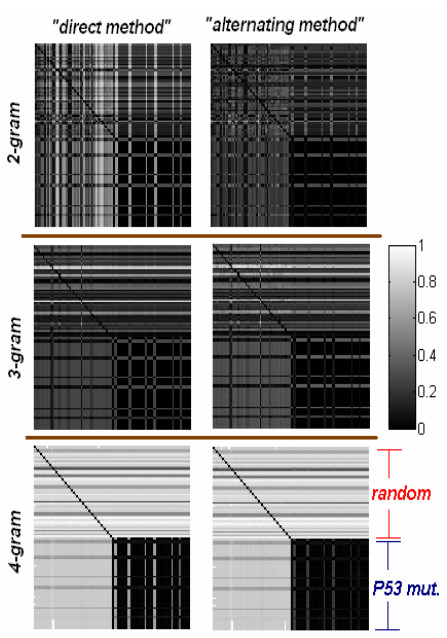


Figure 1
Visualization of the matrices containing all the possible pairwise dissimilarities of the 100 proteins for 2,3,4-gram models

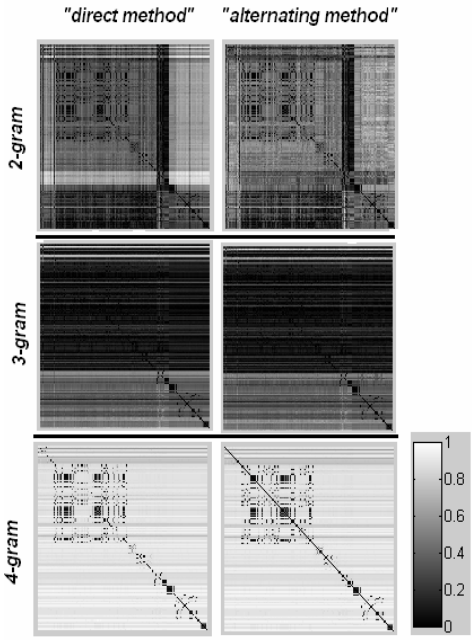


Figure 2
Visualization of the matrices containing all the possible pairwise dissimilarities for the 497 proteins of Set1, for 2,3,4-gram models

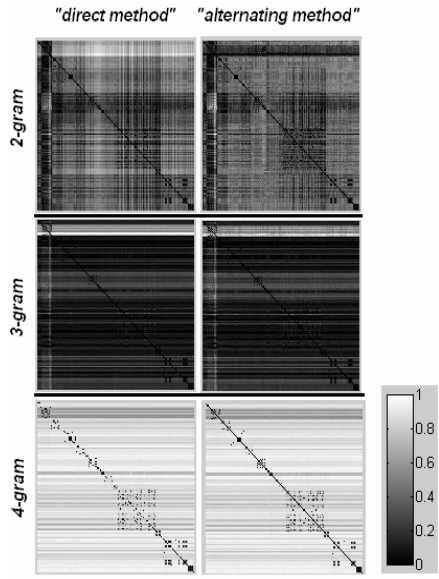


Figure 3
Visualization of the matrices containing all the possible pairwise dissimilarities for the 497 proteins of Set2, for 2,3,4-gram models

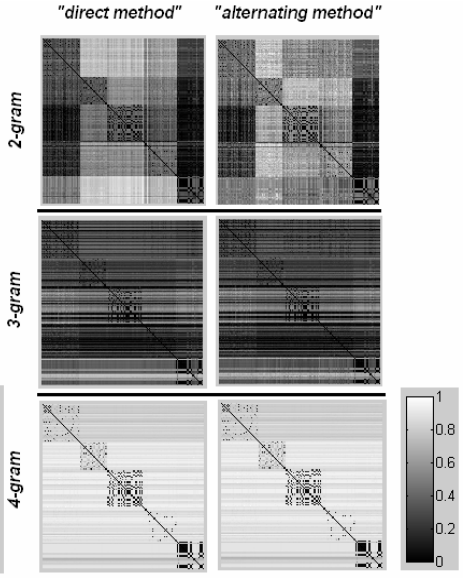


Figure 4
Visualization of the matrices containing all the possible pairwise dissimilarities for the 466 proteins of Set3, for 2,3,4-gram models

In the adopted visualization scheme all the shown matrices (after proper normalization) share a common scale in which the 1 (white) corresponds to the maximum distance in each matrix. It is worth mentioning here that the 'ideal' spatial outlay is a white matrix with only a black segment at the lower right corner. Therefore, it is evident from all these figures that *4*-gram modeling has a very good representation for searching sequence similarity within the given database.

Due to the obvious separation of sequences in the small database we tried to identify the affiliation of sequences grouped as tight cluster in the dark corner of Fig. 1. The results are shown in Figure 5 which is a low-dimensional representation of protein sequences using dissimilarity measure for the small set of experimental data. Here, it is obvioius the fact that we obtained a very god solution to the two class identification problem.

Regarding the cluster identification in the second experiments, we used a strategy based on Huberts's statistics [18] in determining the correlation factors between the clusters we obtained and the 'ground-truth' offered by the original protein sequence families/superfamilies structure.

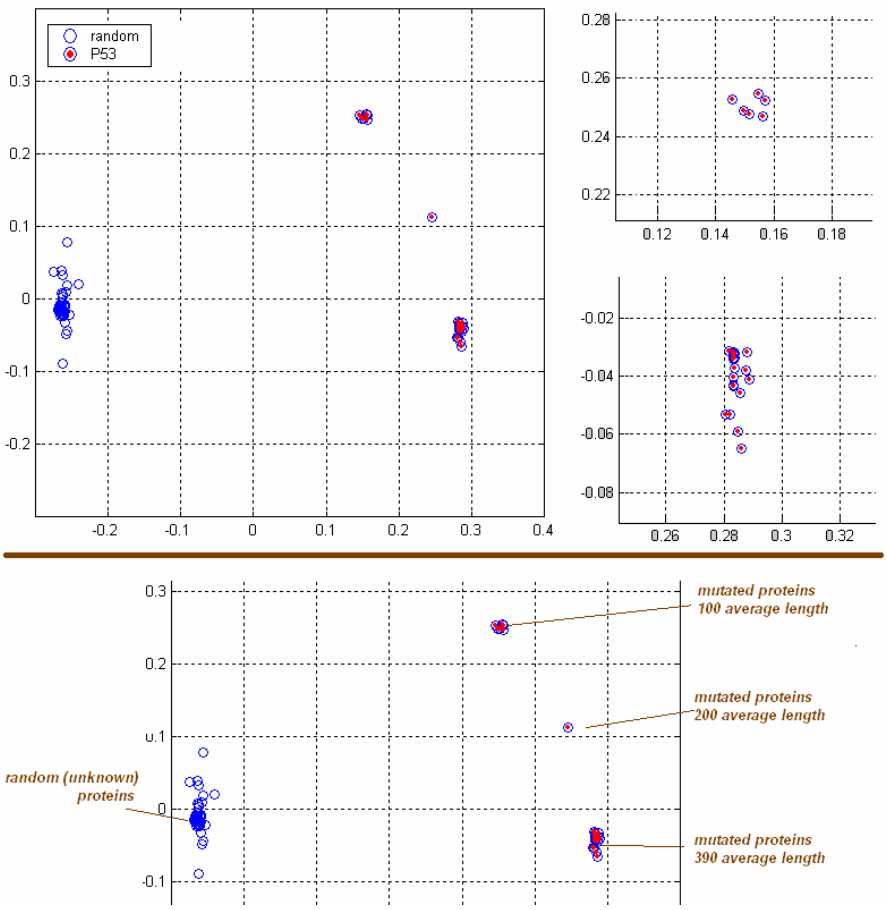


Figure 5

Low-dimensional representation of protein sequences using dissimilarity measure for the small set of experimental data

In order to compare the performance of the new statistical approach with that of an already well recognised method, we apply CLUSTAL W similarity method on the structured database (as it is more complex). The tool performs multiple sequence alignment and generates pairwise similarity scores based on the identification of conserved sequence regions. These scores are used to cluster the protein sequences based on the direct principle of relatedness (the higher the score values the closer the sequences are). In Table 1 we show the correlation factor values.

Table 1
Correlation values based on Huberts's statistics for set 1, 2, 3 of the second protein database using CLUSTAL W and the two approaces of the new method

Set	*CLUSTAL W method*	*Direct method*	*Variant method*
1	0.202	0.1230	0.1110
2	0.127	0.0347	0.0354
3	0.297	0.339	0.3622

In the interpretation of the correlation values there are considered as good scores the high values and it is not the case achieved with the assumption we made (the possible identification of biological structural classification). Even though, it is obvious that CLUSTAL W results are not too far from ours and for the third set we get even better results. Under this circumstances the biologist reasoning helps in elucidating the clusters representation meaning. The explanation comes from the fact that many times sequences of different length and with partial identity in content may belong to the same biological family. So, the clusters we get are representing the similar sequences in textual representation. This conclusion is already justified by the very good performance of mutated proteins identification in the small database.

Conclusions

The method experimented in this paper constitutes a step forward in investigating the engagement of language modelling for characterizing, handling and understanding biological data in the format of sequences. Specifically, we studied the efficiency of this new method in revealing the context relatedness between sequences. The experimental results indicate the reliability of our algorithmic strategy for expressing similarity between proteins according to the sequence text content. Given the conceptual simplicity of the introduced approach, it appears as an encouraging alternative to previous well-established techniques.

Here won't be discussed the two methods comparative performance as the results of the similarity search are geometrically represented but regarding the order of the employed *n*-gram model, after testing with order of 2, 3, 4, 5 we noticed, as can be seen in Figures 1-4 that the performance of the method increases with the order of the model up to 4. After the order of 5 due to the lack of data, the corresponding maximum likelihood estimates become unreasonable uniform and very low.

Analysing the meaning of clusters identified in visual representation of the dissimilarity matrices we may consider that this content evaluation similarity measure performs well for sequences having related textual representation. This aspect may lead to a general clustering strategy. Despite the corelation values that didn't confirm our biological expectation we are motivated to adjust the similarity method by working with functional groups of amino acids. It has to be made the

observation that till now we worked only with concepts from information theory field applied on protein sequence. In addition, CLUSTAL W similarity method that works with biological informations didn't give much higher scores for correlation test. In absence of other correlation reference values we are motivated to consider that it may be used a subjective classification of sequences in SCOP database or in CLUSTAL W similarity principle.

Considering the algorithmic simplicity and computational efficiency of our approach, in this form, we are justified to suggest it as a first choice when searches in large databases are required. In terms of time complexity, without a detailed analysis we are motivated to consider this method efficient especially when search procedure is running over large databases containing long sequences. This motivate us to pursue further on how to achieve even higher performance.

Acknowledgement

Experiments of this work were supported by the EU project Biopattern: Computational Intelligence for biopattern analysis in Support of eHealthcare, Network of Excellence Project No. 508803. Authors are very grateful to dr. Nikos Laskaris for invaluable help in understanding and applying exploratory data algorithms used to represent and conclude over the results.

References

[1] Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Sequence_clustering

[2] Heger A, Holm L: Towards a Covering Set of Protein Family Profiles, Progress in Biophysics and Molecular Biology, Vol. 73, 2000, pp. 321-337

[3] Duda O Richard, Hart E Peter, Stork G David: Unsupervised Learning and Clustering, in Pattern Classification, 2nd edition, John Wiley & Sons, Inc, USA, 2001, pp. 538

[4] Hartigan JA: Clustering Algorithms, John Wiley & Sons, New York, 1975

[5] Kohonen T: Self-organizing Maps. Springer-Verlag, Berlin/Heidelberg, 1997

[6] MacQueen J: Some Methods for Classification and Analysis of Multivariate Observations, in Proceedings of the 5th Berkeley Symp. Math. Stat. Probability. Ed, Cam LML and Neyman J. University of California Press, 1965, 281-297

[7] Fraley C, Raftery AE: Model-based Clustering, Discriminant Analysis, and Density Estimation. J Am Stat Assoc 2002, Vol. 97, pp. 611-631

[8] McLachlan GJ, Basford KE: Mixture Models: Inference and Applications to Clustering, Ed. Marcel Dekker, 1988

[9] McLachlan GJ, Peel D: Finite Mixture Models, Ed. Willey, New York, 2000

[10] Kirsten M., Wrabel, S., & Horvath, T: Distance-based Approaches to Relational Learning and Clustering, in Relational Data Mining, Springer-Verlag New York, Inc. 2000, pp. 213-230

[11] Bogan-Marta A, Gavrielides A Marios, Pitas Ioannis, Lyroudia Kleoniki: A New Statistical Measure of Protein Similarity based on Language Modeling, GENSIPS 2005, May 22-24, in Newport, RI, USA

[12] Van Uytsel DH, and Van Compernolle D: Entropy-based Context Selection in Variable-length n-gram Language Models, *IEEE Benelux Signal Proc. Symp.*, 1998, pp. 227-230

[13] Manning CD, and Schütze H: Foundations of Statistical Natural Language Processing, Massachusetts Institute of Technology Press, Cambridge, Massachusetts London, England, 2000, pp. 554-556; 557-588

[14] National Center for Biotechnology Information: (http://www.ncbi.nlm.nih.gov/)

[15] International Agency for Researc on Cancer: http://www.iarc.fr/p53/Somatic.html

[16] Structural Classification of Proteins official site: http://scop.mrc-lmb.cam.ac.uk/scop/

[17] Laskaris AN: Algorithms for Vectorial Pattern-Analysis, in Basics of Geometrical Data-Analysis, under publishing

[18] Laskaris NA, Ioannides AA: Clinical Neurophysiology, Nr. 113, 2002, 1209-1226

Use of Multi-parametric Quadratic Programming in Fuzzy Control Systems

Zsuzsa Preitl, Radu-Emil Precup

Dept. of Automation and Applied Inf., "Politehnica" University of Timisoara
Bd. V. Parvan 2, RO-300223 Timisoara, Romania
E-mail: zsuzsa.preitl@aut.upt.ro, radu.precup@aut.upt.ro

József K. Tar, Márta Takács

Institute of Intelligent Engineering Systems, Budapest Tech
Bécsi út 96/B, H-1034 Budapest, Hungary
E-mail: tar.jozsef@nik.bmf.hu, takacs.marta@nik.bmf.hu

Abstract: The paper presents some main aspects regarding multi-parametric quadratic programming (mp-QP) problems. Model Predictive Control (MPC) is considered as a particular mp-QP problem, and this powerful tool is applied for control and simulation through a case study. Since the solutions to mp-QP problems can be expressed as piecewise affine linear functions of the state, a new implementation in terms of adaptive network-based fuzzy inference systems is proposed. The presentation is focused on the double integrator plant as a frequently appearing case study (electrohydraulic servo-system).

Keywords: Multi-parametric quadratic programming, model predictive control, adaptive network-based fuzzy inference systems, Multi-Parametric Toolbox

1 Introduction

By multi-parametric programming, a linear or quadratic optimization problem is solved off-line. The multi-parametric approaches are based on off-line computation of the feedback law, having their advantages and disadvantages [15]. The resulting explicit controller generates regions for the control law, their number increases with the complexity of the problem, being able to become easily prohibitive. This is mainly due to the exponential number of transitions between regions, which can occur when a controller is developed in a dynamic programming fashion.

The main method to solve multi-parametric linear programming problems was proposed in [1] and described in [2]. The method is based on constructing the critical regions iteratively, by examining the graph of bases associated to the linear programming tableau of the original problem. Other methods are presented in [3, 4, 5].

The most cited method to solve multi-parametric quadratic programming (mp-QP) problems was formulated in [6]. The method constructs a critical region in a vicinity of a given parameter using Karush-Kuhn-Tucker conditions for optimality, and then it explores recursively the parameter space outside such regions. A very efficient implementation until now is given in [2], and other methods to solve mp-QP problems are presented in [7, 8, 9].

Model predictive control (MPC) represents the accepted standard for complex constrained control problems in industrial applications [10]. During each sampling interval, starting at the current state, an open-loop optimal control problem is solved over a finite horizon, leading to a moving horizon strategy. The drawback of MPC is the relatively high on-line computational effort, which limits its applicability to control relatively slow plants.

This paper addresses the process of moving the necessary calculations for the implementation of MPC off-line in the conditions of considering it a special case of mp-QP problem [6]. In case of MPC algorithms solved in terms of mp-QP problems with piecewise affine linear solutions as functions of the state, the implementation problem is relatively complex [10].

One of the aims of the paper is to propose a new implementation of mp-QP solutions in terms of adaptive network-based fuzzy inference systems [13, 14].

This paper is organized as follows. The next Section presents the main aspects concerning the problem setting in mp-QP and an algorithm to solve mp-QP problems accompanied by comments. In Section 3 the MPC as particular case of mp-QP is analyzed. Then Section 4 is dedicated to the implementation of the piecewise affine linear solutions as functions of the state in case of mp-QP in terms of a neuro-fuzzy approach using adaptive network-based fuzzy inference systems. Section 5 deals with the applications of the mp-QP problems in case studies concentrated on the well accepted double integrator plant in several settings, and the last Section highlights the conclusions.

2 Problem Setting in Multi-parametric Quadratic Programming

The definition of an mp-QP problem is given in terms of [2, 6]:

$$\hat{J}(x) = \min_{z} J(z, x) = \frac{1}{2} z' H z \quad \textit{subject to} \quad G z \le W + S x, \tag{1}$$

where $z \in R^s$ are the optimization (manipulated) variables, $x \in R^n$ is the parameter vector, $H \in R^{s \times s}$, $H > 0$, $W \in R^m$, $S \in R^{m \times n}$. Other non-homogenous problems with the general objective function (2):

$$J(z, x) = z' H z + x' F z, \tag{2}$$

can always be transformed in the problem (1) using the variable substitution (3):

$$\tilde{z} = z + H^{-1} F' x. \tag{3}$$

To solve the mp-QP problem (1) it is necessary to calculate the polyhedral partition of the parameter space. With this respect, the following three definitions are useful [15].

Definition 1: A convex set $\mathrm{Q} \subseteq R^n$ given as an intersection of a finite number of closed half-spaces:

$$\mathrm{Q} = \left\{ x \in R^n \mid Q^x x \le Q^c \right\}, \tag{4}$$

is called polyhedron.

Definition 2: A bounded polyhedron $\mathrm{P} \subseteq R^n$:

$$\mathrm{P} = \left\{ x \in R^n \mid P^x x \le P^c \right\}, \tag{5}$$

is called polytope.

It is obvious from these two definitions that every polytope represents a convex, compact (i.e., bounded and closed) set.

Definition 3: The linear inequality $a' x \le b$ is called valid for a polyhedron P if $a' x \le b$ holds for all $x \in \mathrm{P}$. A subset of a polyhedron is called a face of P if it is represented as:

$$\mathrm{F} = \mathrm{P} \cap \left\{ x \in R^n \mid a' x = b \right\}, \tag{6}$$

for some valid inequality $a' x \le b$. The faces of polyhedron P of dimension 0, 1, $(n-2)$ and $(n-1)$ are called vertices, edges, ridges and facets, respectively.

Connecting to the mp-QP problem (1), given a close polyhedral set $\mathrm{K} \subset R^n$ of parameters:

$$\mathrm{K} = \left\{ x \in R^n \mid Tx \le z \right\}, \tag{7}$$

it is denoted by $\hat{\mathrm{K}} \subset \mathrm{K}$ the region of parameters $x \in \mathrm{K}$ such that the mp-QP problem (1) is feasible and the optimum $\hat{J}(x)$ is finite.

The fundamental aspect of multi-parametric approaches to optimization is that for any given $\bar{x} \in \mathrm{K}$, $\hat{J}(x)$ denotes the minimum value of the objective function in (1) for $x = \bar{x}$, the function $\hat{J} : \hat{\mathrm{K}} \to R$ called value function, expresses the dependence on *x* of the minimum value of the objective function over $\hat{\mathrm{K}}$, and the single-valued function $\hat{z} : \hat{\mathrm{K}} \to R$ describes for any fixed $x \in \hat{\mathrm{K}}$ the optimizer $\hat{z}(x)$ corresponding to $\hat{J}(x)$.

To solve the mp-QP problem (1), the algorithm consisting of the following steps can be used [2]:

- Define the matrices *H*, *G*, *W* and *S* of the problem and set K in (7) according to the desired CS performance objectives.
- Calculate the partition of K according to the steps 1 … 9:

1: Let $x_0 \in \mathrm{K}$ the centre of the largest ball contained in K for which a feasible *z* exists, and ε the solution to the linear programming problem (8) related to this centre:

$$\varepsilon = \max_{z,x} f \text{ subject to } T_i x + f \parallel T_i \parallel \le Z_i,\ i = \overline{1, n_T},\ Gz - Sx \le W, \tag{8}$$

where n_T stands for the number of rows T_i of the matrix *T*.

2: If $\varepsilon \le 0$, then the partition is calculated (no full dimensional critical region is in K). Else, continue with step 3.

3: Solve the mp-QP (1) for $x = x_0$ to obtain $(\hat{z}_0, \hat{\lambda}_0)$.

4: Determine the set of active constraints A_0 when $z = \hat{z}_0$, $x = x_0$, and build G_{A0}, W_{A0} and S_{A0}.

5: If r = rank $G_{A0} < l$ (the number of rows of G_{A0}), then take a subset of *r* linearly independent rows, and redefine G_{A0}, W_{A0} and S_{A0} accordingly.

6: Determine $\hat{\lambda}_{A0}(x)$ and $\hat{z}_{A0}(x)$ from (9):

$$\hat{\lambda}_{A0}(x) = -(G_{A0} H^{-1} G_{A0}')^{-1}(W_{A0} + S_{A0} x),\ \hat{z}_{A0}(x) = -H^{-1} G_{A0}' \hat{\lambda}_{A0}(x). \tag{9}$$

7: Characterize the critical region from (10) where the first inequality corresponds to the constraints in (1) and the second one to the Lagrange multipliers in (9) that must remain nonnegative as *x* is variable:

$$\begin{aligned} &GH^{-1}G_{A0}'(G_{A0}H^{-1}G_{A0}')^{-1}(W_{A0}+S_{A0}x) \le W+Sx, \\ &-(G_{A0}H^{-1}G_{A0}')^{-1}(W_{A0}+S_{A0}x) \ge 0. \end{aligned} \tag{10}$$

8: Define and partition the rest of the region according to [2].

9: Partition each new sub-region.

This algorithm explores the set K of parameters recursively. The partition of the rest of the region into polyhedral sets can be represented as a search tree, with a maximum depth equal to the number of combinations of active constraints.

Concerning the controller implementation, at each sampling interval a polyhedral partition is calculated, with as many different partitions as the length of the prediction horizon. If the Receding Horizon Technique (RHT) is used, then only one polyhedral partition is used out of those which were calculated. That is the reason why the generated controller has only one form. Concerning the algorithm presented before, the following three comments must be highlighted.

Firstly, the very first partition is based on choosing x_0 adequately. Since a choice is involved, there is no single solution to the algorithm.

Secondly, the determination of the set of active constraints is critical, since the remaining regions are partitioned based on some combination of active constraints. Namely, each region represents a region in the parameter space where a number of combinations is active. There is an upper bound for the maximum number, 2^m, where m is the cardinal of the set of all constraints.

Thirdly, the selection of active constraints is not a simple task. All algorithms are based on an iterative procedure that builds up the parametric solution by generating new polyhedral regions of the parameter space at each step. The methods differ in the way they explore the parameter space, that is, in the way they identify active constraints corresponding to the critical regions neighbouring to a given critical region [2]. In [6] the unconstrained critical region is constructed and then the neighbouring critical regions are generated by enumerating all possible combinations of active constraints.

Note that the expression of (9) and the fact that the algorithm is applied for several regions, the control signal will be a piecewise affine linear function of the state.

3 Model Predictive Control as Multi-parametric Quadratic Programming Problem

MPC is employed to solve the constrained regulation problem [6] consisting of regulating towards the origin the plant represented by the discrete-time linear time invariant model (11):

$$\begin{aligned} &x(t+1) = A\,x(t) + B\,u(t), \\ &y(t) = C\,x(t), \end{aligned} \tag{11}$$

fulfilling the constraints (12):

$$y_{\min} \le y(t) \le y_{\max},\ u_{\min} \le u(t) \le u_{\max}, \tag{12}$$

at all time instants $t \ge 0$, where $x(t) \in R^n$, $u(t) \in R^m$ and $y(t) \in R^p$ are the state, input and output vectors, respectively, the variables in (12) are vectors with appropriate dimensions and the pair (A, B) is stabilizable.

Assuming that the state $x(t)$ is fully available at the current time instant t, the MPC is defined in terms of (13):

$$\begin{aligned} &\hat{U} = [\hat{u}'_1,...,\hat{u}'_{t+Nu-1}]' = \underset{U=[u'_1,...,u'_{t+Nu-1}]'}{\arg\min}\ J(U, x(t)) = x'_{t+Ny|t}\,P x_{t+Ny|t} + \\ &\qquad + \sum_{k=0}^{Ny-1} [x'_{t+k|t}\,Q x_{t+k|t} + u'_{t+k}\,R\,u_{t+k}] \text{ subject to} \\ &y_{\min} \le y_{t+k|t} \le y_{\max},\ k = \overline{1, N_c}, \\ &u_{\min} \le u_{t+k} \le u_{\max},\ k = \overline{0, N_c}, \\ &x_{t|k} = x(t), \\ &x_{t+k+1|t} = A\,x_{t+k|t} + B\,u_{t+k},\ k \ge 0, \\ &y_{t+k|t} = C\,x_{t+k|t},\ k \ge 0, \\ &u_{t+k} = K\,x_{t+k|t},\ N_u \le k < N_y. \end{aligned} \tag{13}$$

The MPC problem (13) is solved at each time instant t, where $x_{t+k|t}$ is referred to as the predicted state vector at time $t+k$, obtained by applying the input sequence $u_t, \ldots, u_{t+k-1}$ to the plant (11) starting from the state $x(t)$ in the conditions of K being the state-feedback gain. It is assumed in (13) that $Q = Q' \ge 0$, $R = R' > 0$, $P \ge 0$ and $(Q = C'C, A)$ is detectable. The other parameters in (13) specific to MPC are N_y, N_u and N_c representing the output, input and constraint horizon, respectively, with $N_u \le N_y$ and $N_c \le N_y - 1$. Substituting the state vector $x_{t+k|t}$:

$$x_{t+k|t} = A^k \mathrm{x(t)} + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j}, \tag{14}$$

(13) can be re-expressed as the mp-QP problem (15):

$$V(x(t)) = \frac{1}{2}x'(t)Yx(t) + \min_{U} \frac{1}{2}U'H\,U + x'(t)F\,U \text{ subject to}$$
$$GU \le W + Ex(t), \tag{15}$$

where the column vector $U \in R^s$, $s = mN_u$, is the optimization vector, $H = H' > 0$, and the matrices H, F, Y, G, W and E are obtained from Q, R and (13) [6].

Connecting the approaches in Section 2 and Section 3, the control signal elaborated by the model predictive controller is also a piecewise affine linear function of the state. The following Section will be dedicated to a neuro-fuzzy implementation of the controllers as solutions to mp-QP problems.

4 Neuro-fuzzy Implementation of Piecewise Affine Linear Functions of the State

Examining the algorithm presented in Section 2, the control signal u as piecewise affine linear function is considered in the particular case of a second-order system, $n = 2$, where the general function can be expressed as:

$$u = f(x_1, x_2)\,. \tag{16}$$

Considering three linguistic terms for each of the two input variables, defined in their initial forms in Fig. 1, the complete rule base of a Takagi-Sugeno fuzzy system [16] consisting of 9 rules, $R_1 \ldots R_9$, can be expressed in terms of (17):

$$\begin{aligned}
&R_1: \text{IF } [x_1 \text{ IS } X_{1N}]\, AND\, [x_2 \text{ IS } X_{2N}] \text{ THEN } [u = A_1x_1 + B_1x_2 + C_1]\\
&R_2: \text{IF } [x_1 \text{ IS } X_{1Z}]\, AND\, [x_2 \text{ IS } X_{2N}] \text{ THEN } [u = A_2x_1 + B_2x_2 + C_2]\\
&R_3: \text{IF } [x_1 \text{ IS } X_{1P}]\, AND\, [x_2 \text{ IS } X_{2N}] \text{ THEN } [u = A_3x_1 + B_3x_2 + C_3]\\
&R_4: \text{IF } [x_1 \text{ IS } X_{1N}]\, AND\, [x_2 \text{ IS } X_{2Z}] \text{ THEN } [u = A_4x_1 + B_4x_2 + C_4]\\
&R_5: \text{IF } [x_1 \text{ IS } X_{1Z}]\, AND\, [x_2 \text{ IS } X_{2Z}] \text{ THEN } [u = A_5x_1 + B_5x_2 + C_5]\,.\\
&R_6: \text{IF } [x_1 \text{ IS } X_{1P}]\, AND\, [x_2 \text{ IS } X_{2Z}] \text{ THEN } [u = A_6x_1 + B_6x_2 + C_6]\\
&R_7: \text{IF } [x_1 \text{ IS } X_{1N}]\, AND\, [x_2 \text{ IS } X_{2P}] \text{ THEN } [u = A_7x_1 + B_7x_2 + C_7]\\
&R_8: \text{IF } [x_1 \text{ IS } X_{1Z}]\, AND\, [x_2 \text{ IS } X_{2P}] \text{ THEN } [u = A_8x_1 + B_8x_2 + C_8]\\
&R_9: \text{IF } [x_1 \text{ IS } X_{1P}]\, AND\, [x_2 \text{ IS } X_{2P}] \text{ THEN } [u = A_9x_1 + B_9x_2 + C_9]
\end{aligned} \tag{17}$$

The membership function shapes in Fig. 1 have been obtained supposing that the fuzzy system includes the scaling factors (which can be non-linear in case of implementing MPC), and they correspond to so-called dsigmoidal-type functions according to [14] with the expressions in (18) as difference of two sigmoidal ones:

$$\mu_i(x) = 1/\{1+\exp\{-a_i(x-c_i)]\} - 1/\{1+\exp\{-a_{i+1}(x-c_{i+1})]\},$$
$$x \in \{x_1, x_2\},\ i \in \{1,3,5,7,9,11\}. \tag{18}$$

In the conditions of using use the max and min operators in the inference engine and the weighted average method for defuzzification due to the special form of the consequence in each rule it is easy to observe that the Takagi-Sugeno fuzzy system described here is well suited to model piecewise affine linear functions of the state appearing in case of mp-QP problems.

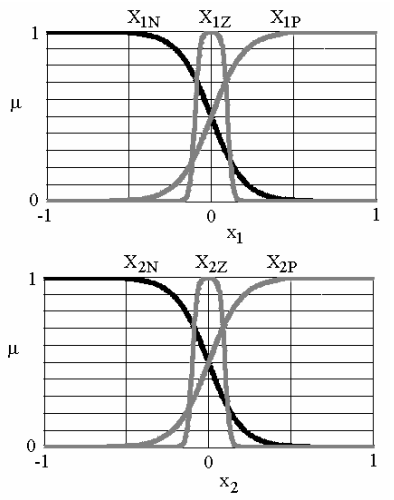


Figure 1
Initial membership functions of input linguistic terms

The adaptive network-based fuzzy inference system, with the schematic diagram illustrated in Fig. 2, has 5 layers:

- Layer I: The inputs x_1 and x_2 are passed through the input linguistic terms with the membership functions having the parameters presented in (18),
- Layer II: The fuzzy rules in (17) are constructed according to the inference engine using the product (Π) of the two antecedents, the output of each node representing the firing strength of a rule,
- Layer III: The firing strengths are normalized dividing them to the sum of all rule firing strengths,
- Layer IV: The normalized firing strength is multiplied by the output functions of the fuzzy inference system,
- Layer V: The overall system output is calculated as the sum of all incoming signals.

Due to this structure similar to that of certain neural networks, the learning techniques specific to neural networks can be applied to minimize the quadratic objective function E:

$$E = \sum_{i=1}^{N} (u_i^d - u_i)^2 \ , \tag{19}$$

where: u_i^d – desired control signal, piecewise affine linear function of the state, to be implemented by the Takagi-Sugeno fuzzy system, u_i – control signal elaborated by the Takagi-Sugeno fuzzy system, N – number of input-output data pairs.

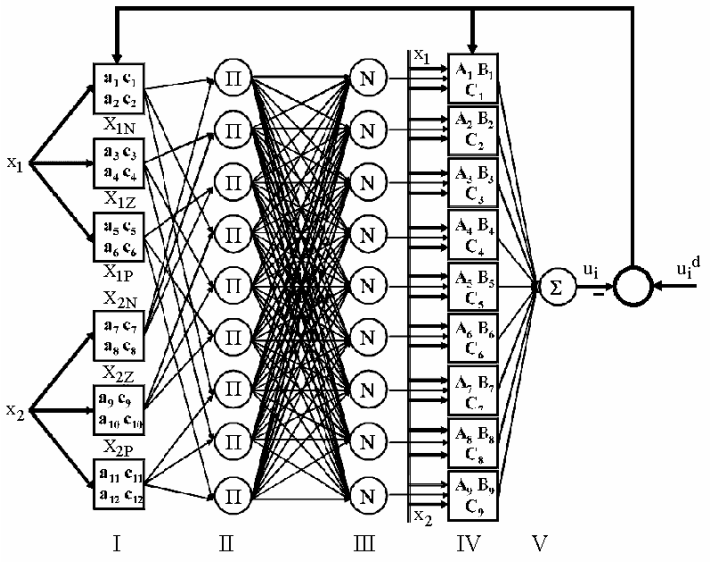


Figure 2
Schematic diagram of adaptive network-based fuzzy inference system

Minimizing the objective function in (19) using the adaptive structure presented in Fig. 2 and appropriate learning techniques leads to the correction of the parameters of the Takagi-Sugeno fuzzy systems in both the antecedents and the consequents.

5 Case Study

A powerful calculation and simulation tool for multiparametric programming consists in the Multi-Parametric Toolbox (MPT). In the paper the MPT is used for modelling, control and simulation of a double integrator plant [17] in its continuous-time representation (for zero initial conditions) (20):

$$y(s) = \frac{1}{s^2}u(s)\,, \tag{20}$$

its equivalent discrete-time state-space representation obtained discretizing in terms of the forward rectangular method for a sampling period of 1 sec., is:

$$\begin{aligned} x(t+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \end{aligned} \tag{21}$$

and the system constraints are:

$$-1 \le u \le 1\,,\quad -15 \le y \le 15\,,\quad -15 \le x_1, x_2 \le 15\,. \tag{22}$$

The objective is to regulate the double integrator (21) towards the origin while minimizing the following quadratic performance index and input constraint [6]:

$$J = \sum_{t=0}^{\infty} [\,y'(t)\,y(t) + 0.01u^2(t)]\,,\ -1 \le u \le 1\ . \tag{24}$$

This problem can be solved by using the MPC algorithm in (13) with $N_u = N_y = 2$, $R = 0.01$, $Q_{11} = 1$, $Q_{12} = Q_{21} = Q_{22} = 0$, and the matrices K and P obtained solving a Riccati equation [6]. The CS behaviour for initial conditions of (21) is illustrated in Fig. 3.

Applying the mp-QP algorithm presented in Section 2 leads in the first phase to 25 controller regions. Merging the controller regions, in this case it was achieved from 25 regions to 21 regions, for the last sampling interval. These regions can be plotted as in Figure 4.

Applying the mp-QP algorithm the control signal will have the piecewise affine linear form (23) that can be well modelled using the neuro-fuzzy approach presented in the previous Section.

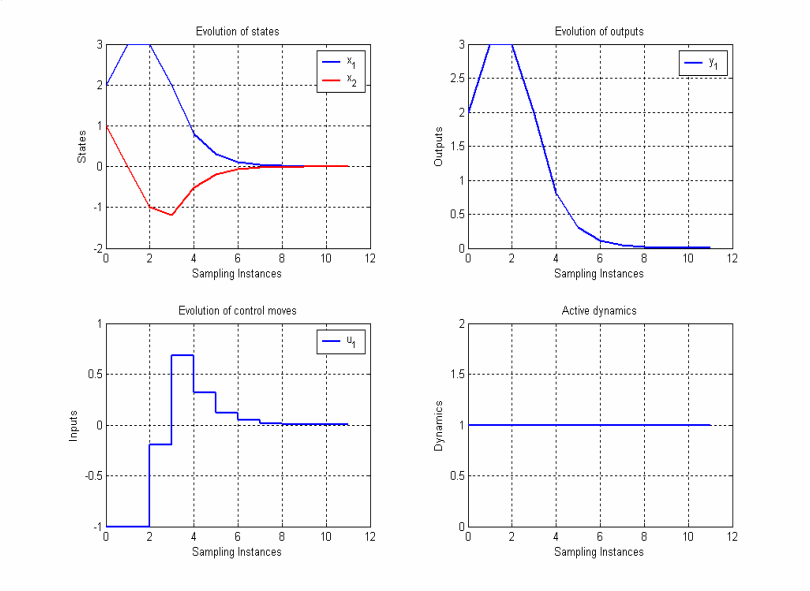


Figure 3
Control system behaviour

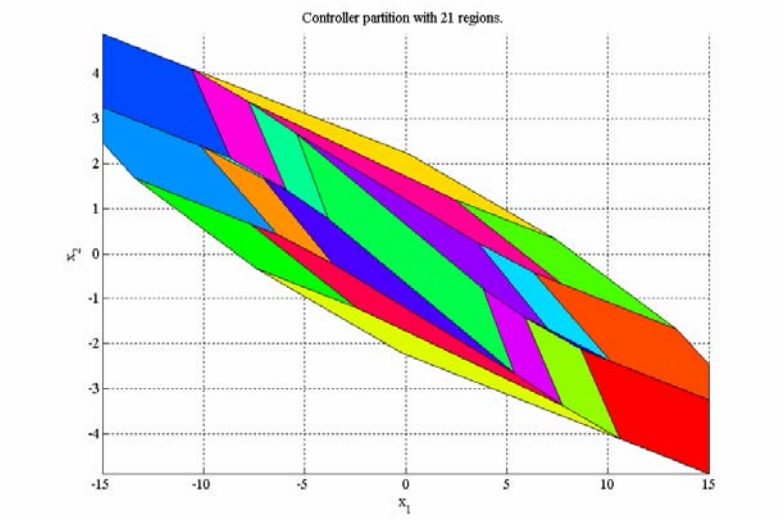


Figure 4
Controller regions

$$u = \begin{bmatrix} -0.57917087\ 112176 & -1.54562698\ 132617 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.43504549\ 869264 & -1.42514751\ 747608 \\ -0.43504549\ 869264 & -1.42514751\ 747608 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.34224024\ 245851 & -1.33799618\ 358994 \\ -0.34224024\ 245851 & -1.33799618\ 358994 \\ -0.21419288\ 842852 & -1.21419288\ 842852 \\ -0.21419288\ 842852 & -1.21419288\ 842852 \end{bmatrix} x + \begin{bmatrix} 0 \\ -1.00000000\ 000000 \\ -1.00000000\ 000000 \\ -1.00000000\ 000000 \\ -1.00000000\ 000000 \\ -1.00000000\ 000000 \\ -1.00000000\ 000000 \\ -1.00000000\ 000000 \\ 0.45607670\ 792517 \\ -0.45607670\ 792517 \\ 1.00000000\ 000000 \\ 1.00000000\ 000000 \\ 1.00000000\ 000000 \\ 1.00000000\ 000000 \\ 1.00000000\ 000000 \\ 1.00000000\ 000000 \\ 1.00000000\ 000000 \\ 0.87990997\ 751597 \\ -0.87990997\ 751597 \\ 1.72724809\ 760682 \\ -1.72724809\ 760682 \end{bmatrix} \quad (23)$$

The control law can also be plotted, as presented in Fig. 5.

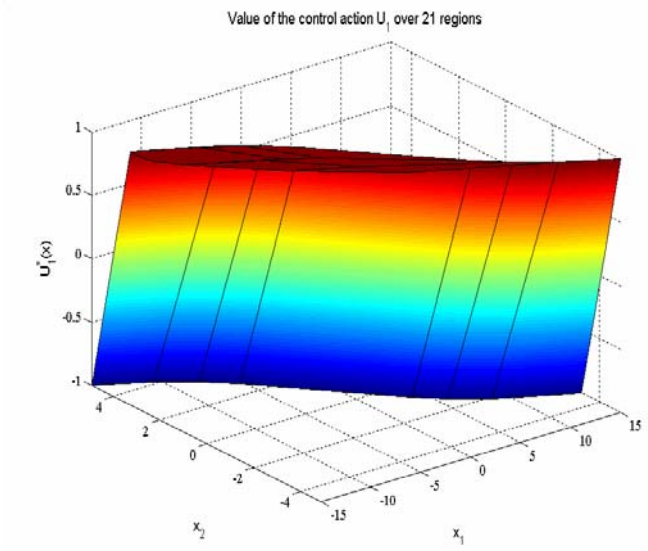


Figure 5
Control signal over the regions

Closed loop trajectories in the state-space can be plotted as well, for all initial conditions that are feasible. For this problem setting (regulation towards origin) for several initial conditions these trajectories are depicted in Fig. 6.

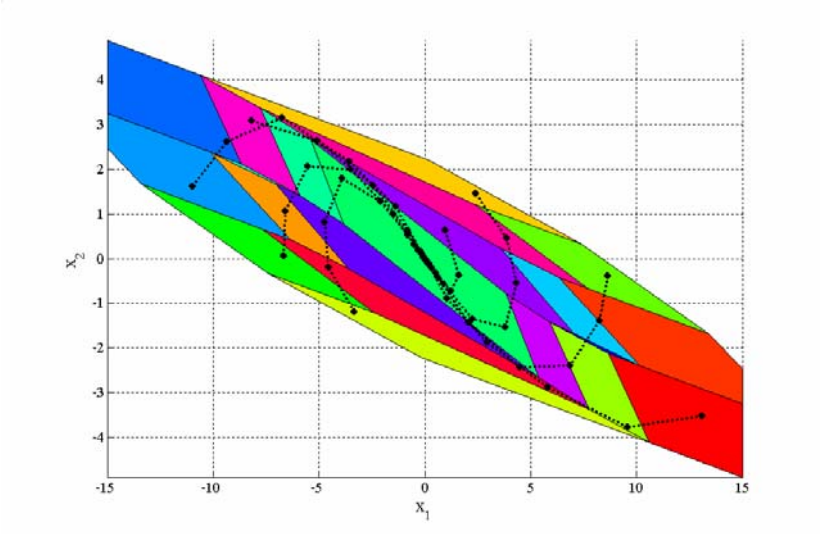


Figure 6
Closed loop trajectories for different initial conditions

Transforming the MPC problem (13) into a mp-QP one (15), the solution to the MPC algorithm can be calculated as piecewise affine linear functions of the state.

Conclusions

The paper presents aspects concerning the mp-QP problems with solutions implemented in state-feedback form as piecewise affine linear functions of the state. These solutions are subject to very convenient implementations as Takagi-Sugeno fuzzy systems using neuro-fuzzy techniques. An example of a double integrator plant is presented, applying the so-called MPT toolbox.

Future research will deal with the computer-aided solution of mp-QP and MPC problems. On the other hand, applications to discrete-event systems in manufacturing and robot control will be tackled [18, 19, 20].

Acknowledgement

The support stemming from the cooperation between Budapest Tech Polytechnical Institution and "Politehnica" University of Timisoara in the framework of the Hungarian-Romanian Intergovernmental Science & Technology Cooperation Program no. 35 ID 17 and from two CNCSIS grants is kindly acknowledged.

References

[1] T. Gal, J. Nedoma: Multiparametric Linear Programming, in Management Science, Vol. 18, 1972, pp. 406-442

[2] F. Borelli: Constrained Optimal Control of Linear and Hybrid Systems, Springer-Verlag, Berlin, Heidelberg, 2003

[3] M. Schechter: Polyhedral Functions and Multiparametric Linear Programming, in Journal of Optimal Theory and Applications, Vol. 53, No. 2, 1987, pp. 269-280

[4] T. Gal: Postoptimal Analyses, Parametric Programming, and Related Topics, de Gruyer, Berlin, 2nd edition, 1995

[5] C. Filippi: On the Geometry of Optimal Partition Sets in Multiparametric Linear Programming, Technical Report 12, Department of Pure and Applied Mathematics, University of Padova, Italy, 1997

[6] A. Bemporad, M. Morari, V. Dua, S. Pistikopoulos: The Explicit Linear Quadratic Regulator for Constrained Systems, in Automatica, Vol. 38, No. 1, 2002, pp. 3-20

[7] M. M. Seron, J. A. De Doná, G. C. Goodwin: Global Analytical Model Predictive Control with Input Constraints, in Proceedings of 39th IEEE Conference on Decision and Control, Sydney, Australia, 2000, pp. 154-159

[8] P. Tøndel, T. A. Johansen, A. Bemporad: An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions, in Proceedings of 40th IEEE Conference on Decision and Control, Orlando, FL, Vol. 3, 2001, pp. 2849-2854

[9] M. Baotic: An Efficient Algorithm for Multi-Parametric Quadratic Programming, Technical Report AUT02-04, Automatic Control Laboratory, ETH Zürich, Switzerland, 2002

[10] E. F. Camacho, C. Bordons: Model Predictive Control, Springer-Verlag, Berlin, Heidelberg, 2nd edition, 2004

[11] H. W. Chiou, E. Zafiriou: Frequency Domain Design of Robustly Stable Constrained Model Predictive Controllers, in Proceedings of the American Control Conference, Baltimore, USA, Vol. 3, 1994, pp. 2852-2856

[12] T. A. Johansen, I. Petersen, O. Slupphaug: On Explicit Suboptimal LQR with State and Input Constraints, in Proceedings of 39th IEEE Conference on Decision and Control, Sydney, Australia, 2000, pp. 662-667

[13] J. Jang: ANFIS: Adaptive Network-based Fuzzy Inference System, in IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, 1993, pp. 665-685

[14] Mathworks: Fuzzy Logic Toolbox User's Guide V. 3, Natick, MA, 2000

[15] M. Kvasnica, P. Grieder, M. Baotić, F. J. Christophersen: Multi-Parametric Toolbox (MPT), Tutorial

[16] T. Takagi, M. Sugeno: Fuzzy Identification of Systems and its Applications to Modelling and Control, in IEEE Transactions on Systems, Man, and Cybernetics, Vol. 15, No. 1, 1985, pp. 116-132

[17] S. Preitl, R-E. Precup, Zsuzsa Preitl: Sensitivity Analysis of Low Cost Fuzzy Controlled Servo Systems, 16th World Congress of International Federation of Automatic Control 2005, Prague, Czech Republic, Preprints, Editors: P. Horacek, M. Simandl, P. Zitek, DVD, paper index 1794 (6 pages)

[18] L. Horváth, I. J. Rudas: Modeling and Problem Solving Methods for Engineers, Elsevier, Academic Press, 2004

[19] L. Horváth, I. J. Rudas, C. Couto: Integration of Human Intent Model Descriptions in Product Models, in Digital Enterprise - New Challenges Life-Cycle Approach in Management and Production, Kluwer Academic Publishers, 2001, pp. 1-12

[20] L. Horváth, I. J. Rudas, J. F. Bitó, A. Szakál: Adaptive Model Objects for Robot Related Applications of Product Models, in Proc. 2004 IEEE International Conference on Robotics & Automation ICRA 2004, New Orleans, LA, USA, 2004, pp. 3137-3142

Identification and Model-based Compensation of Striebeck Friction[1]

Lőrinc Márton

Dept. of Electrical Engineering
Sapientia Hungarian University of Transylvania
Trandafirilor 61, 540053 Piata, Romania
martonl@ms.sapientia.ro

Béla Lantos

Dept. of Control Engineering and Information Technology
Budapest University of Technology and Economics
Magyar tudósok krt. 2, H-1117 Budapest, Hungary
lantos@iit.bme.hu

Abstract: The paper deals with the measurement, identification and compensation of low velocity friction in positioning systems. The introduced algorithms are based on a linearized friction model, which can easily be introduced in tracking control algorithms. The developed friction measurement and compensation methods can be implemented in simple industrial controller architectures, such as microcontrollers. Experimental measurements are provided to show the performances of the proposed control algorithm.

Keywords: tracking control, friction modelling and identification, friction compensation

1 Introduction

Many industrial applications require precise positioning of a mechanical system, namely moving an object in a given position in space with a given orientation. Some common applications are material manipulation with industrial robots or cranes, positioning in Hard Disk Drives or optical drives. Other applications

[1] The research results from this paper originally were presented in the [10] and [11] conference papers.

require a controlled motion in space along a predefined path, for example welding, spray painting with robots (trajectory tracking).

These tasks are commonly solved with feedback control. The aim of the control algorithm is to calculate the command signal for electric motors or other type of actuators which drive the mechanical system in order to obtain zero or acceptably small difference between the real and desired position.

Nowadays many types of high resolution position sensors and precise, fast actuators are available at relatively low cost. With these devices very high precision position control tasks became achievable for industrial applications. However in many practical applications it was observed that the high precision position tracking control performances are severely influenced by friction in a negative sense. This is why the search for new friction modelling and identification techniques became a popular research trend.

Friction is a nonlinear phenomenon which is universally present in the motion of bodies in contact. In servo controlled machines friction has an impact in all regimes of operation. In high precision positioning systems it is inevitable to know the value of the friction force to assure good control characteristics and to avoid some undesired effects such as limit cycle and steady state error, tracking lags.

The nonlinear and dynamic behavior of friction is accentuated near zero velocities. Many practical applications require precise motion control in low velocity regime. As examples can be mentioned the space telescopes that should track the motion of a star [1] and positioning applications, when the start point and the end point are near to each other.

Accordingly, in high precision position control systems the friction force should be taken into consideration at the formulation of the control law. To describe the friction phenomena, in [2] nonlinear models were proposed that can explain the nonlinear behavior of friction at low velocities. Form theoretical point of view an important result was formulated in [3], in which it was shown that the dynamics of a single input mechanical system with Coulomb friction has a well defined, absolutely continuous solution (Carathedory solution). To measure the frictional parameters, friction identification and measurement methods were also discussed in many studies. In [4] a time domain identification method is proposed for static friction models which are not necessarily linear in parameters. The method needs no information of acceleration and mass, the only assumption is that the initial and final velocity during the identification must be identical. Neural network based identification methods are also popular to capture the frictional behavior. In [5] Support Vector Machines were proposed for friction modelling and identification. The advantage of this approach is that it can identify nonlinearities from sparse training data. Genetic algorithm based identification for the Striebeck friction parameters was described in [8]. Frequency domain identification methods were also proposed to identify friction. The study [9] presents a frequency domain identification method for simultaneous identification of velocity and position

dependent friction. The compensation of the frictional effects in positioning systems is also discussed in many papers. In the paper [6] a nonlinear observer is developed to compensate the Coulomb friction. In the works of Laura et al. the extended Kalman filter technique is proposed for friction estimation [7].

In this paper a model based compensation method is presented. The rest of the paper is organised as follows: Section 2 is divided in three subsections. The first subsection presents the friction model, which is used in the control algorithm. Afterward, the second subsection presents a parameter identification for the introduced model. The third subsection in Section 2 presents the proposed tracking control algorithm. In Section 3 experimental measurements are given. The final Section sums up the conclusions of this study.

2 Friction Modelling, Identification and Compensation

2.1 Friction Modelling near Striebeck Velocities

Many models were developed to explain the friction phenomenon. These models are based on experimental results rather than analytical deductions and generally describe the friction force (F_f) in function of velocity (v).

The classical *static + kinetic + viscous friction model* is the most commonly used in engineering. This model has three components: the constant Coulomb friction term ($F_C sign(v)$), which depends only on the sign of velocity, the viscous component ($F_V v$), which is proportional with the velocity and the static term (F_S), which represents the force necessary to initiate motion from rest and in most of the cases its value is grater than the Coulomb friction: (see Figure 1.)

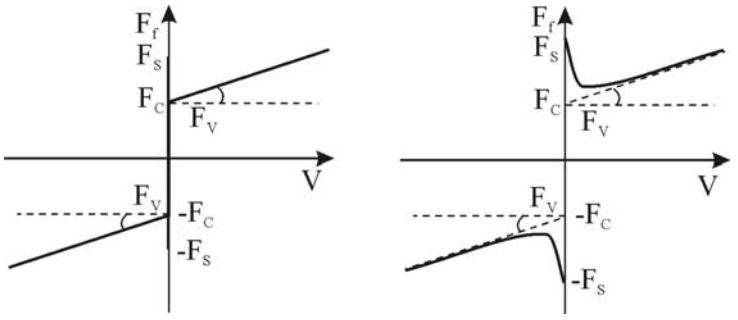


Figure 1
Static+kinetic+ viscous and Striebeck friction

$$F_f = \begin{cases} F_C sign(v) + F_V v, v \neq 0 \\ F_S, v = 0_+ \\ -F_S, v = 0_- \end{cases} \quad (1)$$

The servo-controlled machines are generally lubricated with oil or grace (hydrodynamic lubrication). Tribological experiments showed that in the case of lubricated contacts the simple static +kinetic + viscous model cannot explain some phenomena in low velocity regime, such as the *Striebeck effect*. This friction phenomenon arises from the use of fluid lubrication and gives rise to decreasing friction with increasing velocities.

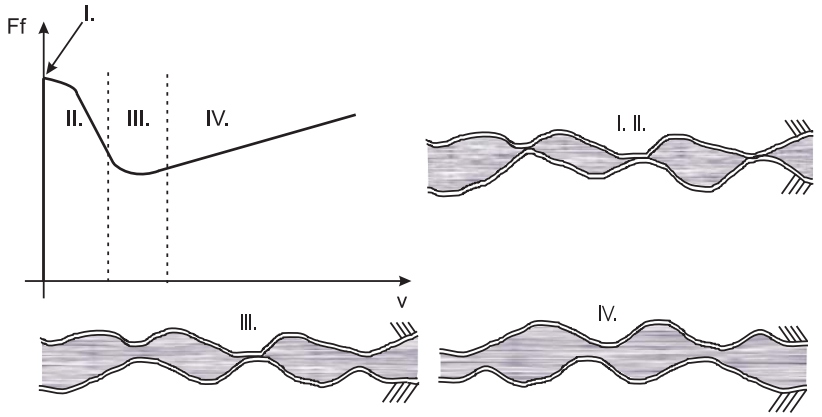


Figure 2
Striebeck Friction Regimes

To describe this low velocity friction phenomenon, four regimes of lubrications can be distinguished (see Figure 2). *Static Friction: (I.)* the junctions deform elastically and there is no excursion until the control force does not reach the level of static friction force. *Boundary Lubrication: (II.)* this is also solid to solid contact, the lubrication film is not yet built. The velocity is not adequate to build a solid film between the surfaces. A sliding of friction force occurs in this domain of low velocities. The friction force decreases with increasing velocity but generally is assumed that friction in boundary lubrication is higher than for fluid lubrication (regimes three and four). *Partial Fluid Lubrication: (III.)* the lubricant is drawn into the contact area through motion, either by sliding or rolling. The greater the viscosity or motion velocity, the thicker the fluid film will be. Until the fluid film is not thicker than the height of aspirates in the contact regime, some solid-to-solid contacts will also influence the motion. *Full Fluid Lubrication: (IV.)* When the lubricant film is sufficiently thick, separation is complete and the load is fully supported by fluids. The viscous term dominates the friction phenomenon, the solid-to-solid contact is eliminated and the friction is 'well behaved'. The value of the friction force can be considered as proportional with the velocity.

From these domains results a highly nonlinear behavior of the friction force. Near zero velocities the friction force decreases in function of velocity and at higher velocities the viscous term will be dominant and the friction force increases with velocity. Moreover it also depends on the sign of velocity with an abrupt change when the velocity pass through zero.

For the moment no predictive model of the Striebeck effect is available. Several empirical models were introduced to explain the Striebeck phenomena, such as the Tustin model [2]:

$$F_f = (F_C + (F_S - F_C)e^{-|v|/v_S})sign(v) + F_V v \tag{2}$$

The model introduced in this paper is based on Tutin friction model and on its development, the following aspects were taken into consideration:

- allows different parameter sets for positive and negative velocity regime
- easily identifiable parameters
- the model clearly separates the high and low velocity regimes
- can easily be implemented and introduced in real time control algorithms

For the simplicity, only the positive velocity domain is considered, but same study can be made for the negative velocities. Assume that the mechanical system moves in $0 \ldots v_{max}$ velocity domain.

Consider a linear approximation for the exponential curve represented by two lines: d_{1+} which cross through the $(0,F_f(0))$ point and it is tangent to curve and d_{2+} which passes through the $(v_{max}, F_f(v_{max})$ point and tangential to curve. (see Figure 3.) These two lines meet each other at the v_{sw} velocity. In the domain $0 \ldots v_{sw}$ the d_{1+} can be used for the linearization of the curve and d_{2+} is used in the domain $v_{sw} \ldots v_{max}$. The maximum approximation error occurs at the velocity v_{sw} for both linearizations.

If the positive part of the friction model (2) is considered ($v>0$), the obtained equations for the d_{1+} and d_{2+}, using Taylor expansion, are:

$$d_{1_+} : F_{L1f_+}(v) = F_S + \left.\frac{\partial F_f(v)}{\partial v}\right|_{v=0} v = F_S + (F_V - (F_S - F_C)/v_s)v \tag{3}$$

$$\begin{aligned} d_{2_+} : F_{L2f_+}(v) &= F_f(v_{\max}) + \left.\frac{\partial F_f(v)}{\partial v}\right|_{v=v_{\max}} (v - v_{\max}) \\ &= F_f(v_{\max}) + (F_V - (F_S - F_C)/v_s)e^{-v_{\max}/v_S}(v - v_{\max}) \end{aligned} \tag{4}$$

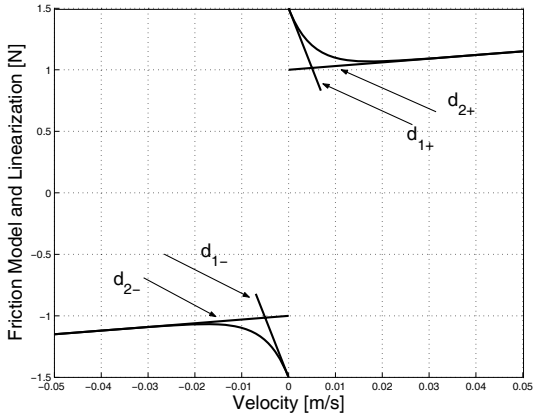


Figure 3
Linearization of Striebeck friction

Thus the linearization of the exponential friction model with bounded error can be described by two lines in the *0 ... v_{max}* velocity domain:

$$d_{1_+} : F_{L1f_+}(v) = a_{1+} + b_{1+}v, \; if \; 0 \le v \le v_{sw+} \tag{5}$$

$$d_{2_+} : F_{L2f_+}(v) = a_{2+} + b_{2+}v, \; if \; v_{sw+} \le v \le v_{\max} \tag{6}$$

with:

$$v_{sw+} = \frac{a_{1+} - a_{2+}}{b_{2+} - b_{1+}} = \frac{-F_S + F_f(v_{\max}) + (F_V - (F_S - F_C)/v_s)e^{-v_{\max}/v_S}}{(F_V - (F_S - F_C)/v_s)(-1 + e^{-v_{\max}/v_S})} \tag{7}$$

Same study can be made for negative velocities. Based on linearization, the friction can be modelled as follows:

$$F_f(v) = \begin{cases} a_{1+} + b_{1+}v, \; if \; 0 \le v \le v_{sw+} \\ a_{2+} + b_{2+}v, \; if \; v_{sw+} \le v \le v_{v\max} \\ a_{1-} + b_{1-}v, \; if \; v_{sw-} \le v \le 0 \\ a_{2-} + b_{2-}v, \; if \; v_{v\max} \le v \le v_{sw-} \end{cases} \tag{8}$$

It can be seen that the model is linearly parameterized and it can be implemented with low computational cost.

2.2 Friction Measurement and Parameter Identification

2.2.1 Friction Measurement

For the friction force measurements it is assumed that the load is driven by a servo motor and the torque developed by the motor is proportional with the command signal. The friction force can appear inside the motor, in the gearbox between the load and the motor and at the load side. The friction to identify is the sum of all these friction forces. As it was presented in the previous subsection, the relationship between the friction behavior F_f and the velocity v is a mapping $F_f = F_f(v)$. The identification task in this is to obtain the parameters of the model (8) from a finite number (N) available measurements $(v_i, F_{fmi}), i=1..N$, where $F_{fmi} = F_{fi} + d_i$. The term d_i is the measurement error on the i'th measurement data.

The method is presented for positive velocity regime.

The dynamics of the positioning system reads as:

$$\begin{aligned} \dot{x} &= v \\ m\dot{v} &= u - F_f(v) \end{aligned} \tag{9}$$

with m mass of the load and u is the control input force, x denotes the position.

It can be seen that if the velocity is kept constant, the friction force is proportional with control signal u, $F_f(v) = u$. Hence if the positioning system is stabilized to different angular velocities v_i, the value of the friction force will be proportional with the command signal.

The method needs high precision velocity control. It is known that the linear PI control algorithm assures only poor transient performances for velocity tracking but guarantees precise final tracking accuracy, if the reference velocity is kept constant. It suggests that for parameter identification it is enough to use standard PI algorithm for velocity control: $u = K_P((v_{ref} - v) + 1/T_i \int (v_{ref} - v)dt)$. The well tuned PI controller guarantees precise velocity control.

The measurement algorithm can be summarized as follows: (see Figure 4.)

- *Stabilize the velocity to* v_{refi}
- *Wait a time period T1 to get rid of transients.*
- *After the transients, calculate the average of the speed (v) and the control signal (u) over a time period T2 to get rid of measurement noise.*
- *Save the measurement data* (v_i, u_i).
- *Repeat the sequence for the next velocity* v_{refi+1}

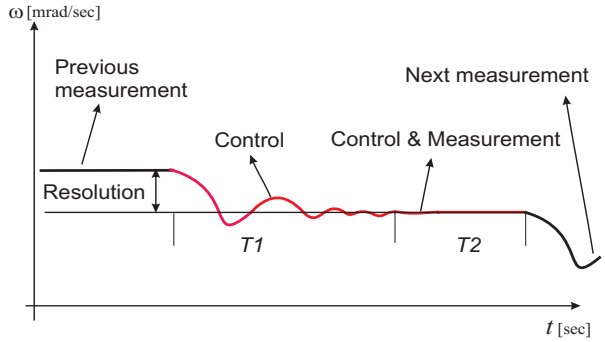


Figure 4
Velocity control for friction measurement

Note that during the data collection the closed loop velocity control algorithm remains active.

For precise velocity stabilization, precise velocity measurements are needed. Denote the encoder resolution with N_E. This value is generally given in Pulses Per Turn (*PPT*). The velocity can be measured with the encoder by counting the encoder pulse over a period of time T (*pulse counting method*). The velocity measurement accuracy is: $\Delta v_C = 2\pi / N_E T[rad/\sec]$. However at low velocities higher accuracy can be obtained using the *pulse timing method:* count the number of pulses of a high frequency external timer over a single encoder pulse. In this case the accuracy of this measurement method is $\Delta v_T = Nv^2 / 2\pi f[rad/\sec]$ where f is the frequency of the timer. For the better accuracy at low speed regime the pulse timing method and at the high speed regime the pulse counting method should be used. The optimal switching between the methods is at the speed $v_{TC} = 2\pi / N_E \sqrt{f/T}\,[rad/\sec]$, where $\Delta v_T = \Delta v_C$. By combining these two methods the velocity can be measured using standard industrial encoders ($N_E \leq$ *5000 PPT*) with high accuracy even at low speed regime.

2.2.2 The Parameters of the Lines

The first line (d_{1+}), given by (5), characterizes the friction phenomena at low velocities, where the friction force has a downward behavior in function of velocity. At high speeds the friction increases almost linearly with the velocity, the second line (d_{2+}), given by (6), should be fitted on this part of the Striebeck curve. Hence, let us consider two subgroup of measurement data: the first N_1 measurements at the decreasing part of the curve, and the final N_2 measurements where the friction force increases with velocity.

The parameters of d_{1+} and d_{2+} can be determined as a solution of the following optimization problems:

$$\min_{a_{1+},b_{1+}} \sum_{i=1}^{N_1} (F_{fi}(v_i) - (a_{1+}v_i + b_{1+}))^2 \qquad \min_{a_{2+},b_{2+}} \sum_{i=N_2}^{N} (F_{fi}(v_i) - (a_{2+}v_i + b_{2+}))^2 \quad (10)$$

Applying standard optimization techniques such as the the *Least Squares (LS)* method, the friction parameters can easily be calculated.

2.3 Position Tracking Combined with Friction Compensation

The tracking control problem for the system (9) can be formulated as follows: given a reference position trajectory $x_d=x_d(t)$, a twice differentiable function of time, determine the command signal u which guarantees, that $x(t)-x_d(t) \rightarrow 0$, as $t \rightarrow \infty$.

In order to solve the tracking problem, define the following tracking error metric, which combines the time dependent position and velocity errors: ($e_x = x - x_d, e_v = v - v_d$):

$$S = e_v + \lambda e_x \quad (11)$$

where $\lambda > 1$ is a constant parameter.

Based on the plant model (9), the tracking error dynamics reads as:

$$\dot{S} = (u - F_f(v))/m - \ddot{x}_d + \lambda e_v \quad (12)$$

Consider the control law as:

$$u = -m(-\ddot{x}_d + \lambda e_v + K_S S) + \hat{F}_f(v), \quad K_S > 0 \quad (13)$$

Note that if we have $\hat{F}_f(v) - F_f(v) \approx 0$ yields $\dot{S} + K_S S = 0$, the combined position and velocity error converges to zero.

Hence the tracking problem can be solved, if the control algorithm contains a precise friction compensator term, which can 'cancel' the effect of frictional force on the system dynamics. $\hat{F}_f(v)$ can be modelled according to the relation (8) where the model parameters can be obtained as it was described in subsection 2.2.

3 Experiemental Results

3.1 Experimental Setup

The experimental setup consists of a permanent magnet *24V* DC servo motor with 38.2 [mNm/A] torque constant. The motor drives a metal disc with known inertia (J = 0.015 kgm^2) through a *1:66* gear reduction (N=*66*). Friction is introduced via a metal surface, which is held against the disc (see Figure 5). The contact between the disc and the metal surface is lubricated with grease. The reaction torque generated by the friction component related to the motor side also can be written as a sum of three terms $F_f = F_{fR} + F_{fG} + F_{fL}/N$, where F_{fR} denotes the friction component inside the motor, F_{fG} denotes the friction component inside the gearhead, F_{fL} is the friction component at the load side.

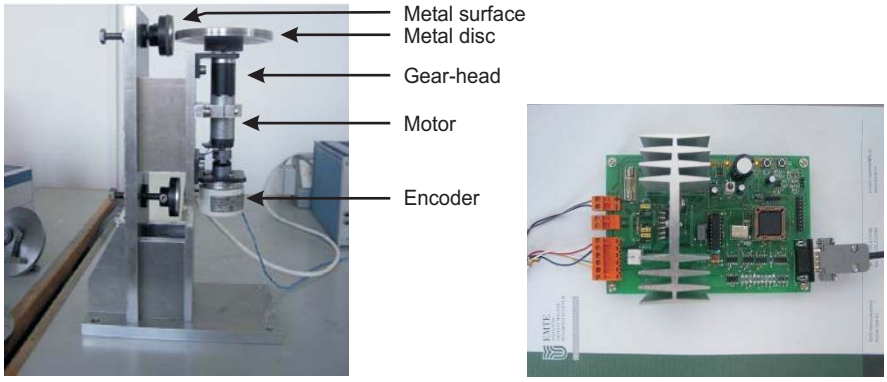


Figure 5
The experimental setup and the control circuit

The friction measurement and control algorithm are implemented on a PIC18 type microcontroller with *40 MHz* clock frequency. The used C compiler for the implementation of the control algorithms allows floating point representation. The microcontroller is connected to an IBM-PC computer through RS232 serial port. The PC is used only for data monitoring and off-line data processing.

The DC servo motor is driven by a H-bridge amplifier. The armature current is controlled by a high speed, analog current controller. The microcontroller is interfaced to the current servo amplifier through a 11 bit DAC. The command signal calculated by the control algorithm running on the controller represents the reference for the current controller. Hence the positioning system is controlled by a cascade control architecture.

The angular position and velocity of the mechanical system are measured using a *5000 PPT* two channel rotational encoder. The encoder is interfaced through a signal conditioner circuit to microcontroller which also determines the direction of rotation. The impulses of the encoder are counted using the embedded 16 bit timers of the controller. The pulse counting method uses the *Timer 0* block of the controller which has external clock input. The counting period is set to *5 msec*. The pulse timing method is implemented using the *Capture* block of the controller, which generates an interrupt when positive signal edge appears on its external input. The high frequency timer, necessary for the measurement in pulse timing mode is derived from the microcontroller clock frequency. The switching between the two methods are implemented in the velocity and position measurement software module.

3.2 Measurement and Identification Results

To obtain the low velocity friction characteristics, the friction force was measured in *0 ... 0.5 [rad/sec]* velocity domain (at the load side). The speed resolution was chosen *5 [mrad/sec]*. Accordingly, totally *N=100* measurements data were collected. A *PI* type control algorithm stabilizes the motor speed for each reference speed with $K_P=15$ proportional gain and $T_i=0.24$ *[sec]* integral time constant. The algorithm was implemented with *5 [msec]* sampling period. When the reference speed value is changed, for *T1=50* sampling period no data was collected in order to get rid of transients, and after that for *T2=16* sampling period the average of the velocity values and control signal values were calculated to obtain one measurement point.

The measurements clearly capture the increasing and decreasing part of the Striebeck curve. (See Figure 6.) On the first *15* measurements at low velocities a line was fitted using *LS* method, which optimized the cost function (10), to obtain the parameters a_{1+} and b_{1+}. On the last *50* measurements another line was fitted to obtain the a_{2+} and b_{2+} parameters, which characterize the high velocity regime. The v_{sw+} parameter was determined from the relation (7).

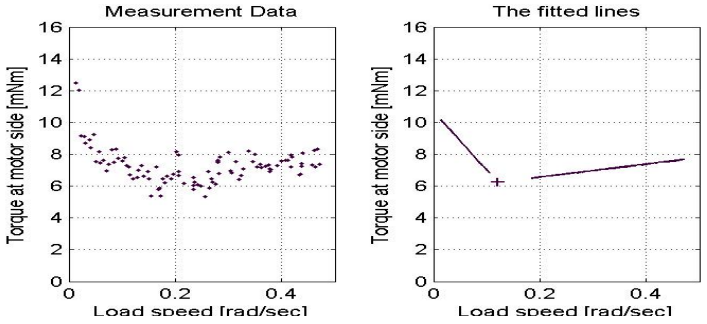


Figure 6

Friction measurement results and the fitted lines

Note that the friction was determined at the load side and the velocity is the velocity of the load. It can be seen (Figure 6) that the obtained model fits well the measurement data.

For the positive velocity regime, the following parameter values obtained during the identification are: $a_{1+} = 11.6$ *[mNm],* $b_{1+} = -61.2$ *[mNmsec/rad].* $a_{2+} = 5.7$ *[mNm],* $b_{2+} = 4$ *[mNmsec/rad],* $v_{sw+} = 0.085$ *[rad/sec]*

3.3 Compensation Results

In order to test the proposed friction compensation method, the control law (13) was implemented on the microcontroller, with the following parameters: $K_S=1$, $\lambda=10$. Because the motion is rotational, the mass m is replaced by the inertia of system (J). The desired track contains has acceleration, constant speed, and deceleration regimes in both positive and negative velocity regimes.

Two experiments were carried out. In the first experiment, the friction compensator term ($\hat{F}_f(v)$) was neglected from the control law. In the second experiment, the friction compensator term was introduced in the control law according to the model (8). The experimental results are presented in Figures 7 and 8. For the numerical evaluation, the following error sum is considered. $E_S = \frac{1}{N}\sum_{i=1}^{N}|S_i|$, where N represents the number of measurements ($N=1000$). Without friction compensation, we obtained $E_S=0.3558$. With friction compensation we obtained E_S=0.0656. Accordingly, the proposed friction compensation method clearly outperforms the classical linear control algorithms, in which the friction is not taken into consideration.

Conclusions

A friction identification and comepensation method was proposed for mechatronic systems, which operates at low velocity regimes near Striebeck velocities. The introduced control algorithm is developed for position tracking tasks and it is based on a linearized friction model, which can easily be implemented on simple industrial controller and its parameters can be identified with standard LS techniques. Experimental measurements shows the proposed tracking control algorithm, modified with a friction compensator term guarantees more precise trajectory tracking than classical control algorithms.

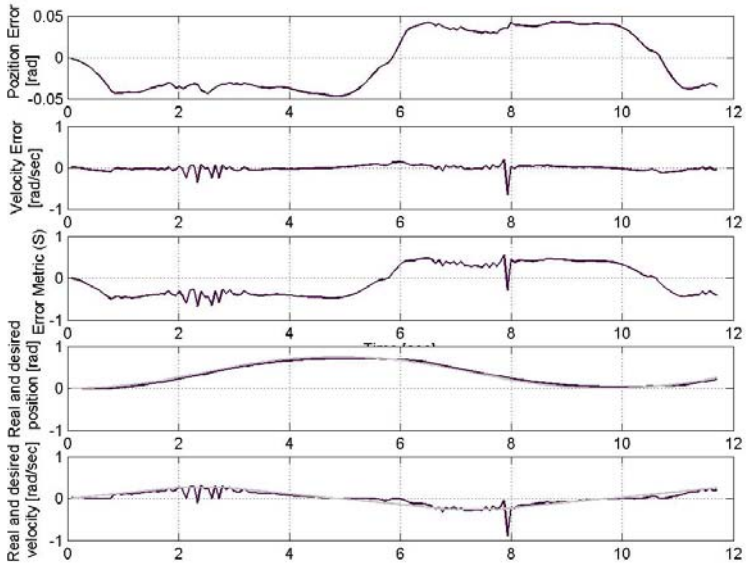


Figure 7
Tracking without friction compensation

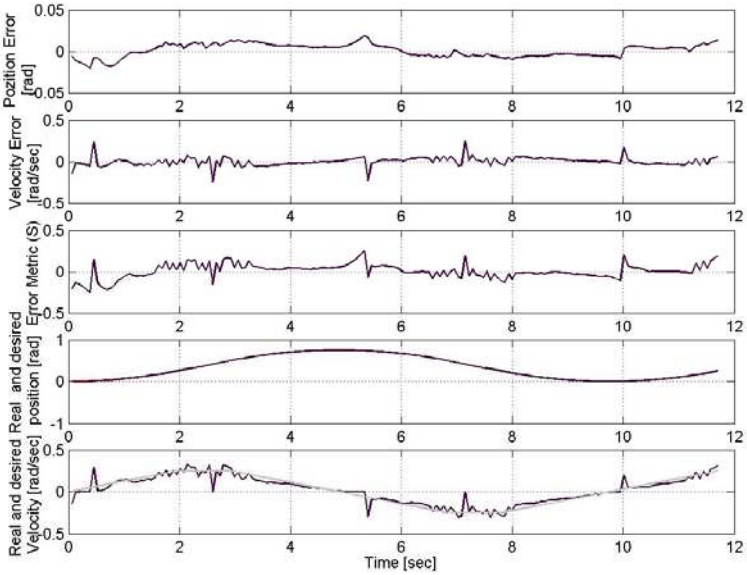


Figure 8
Tracking with friction compensation

Acknowledgement

The research was supported by the Hungarian National Research Program under grant No. OTKA T 042634. The first author research was also supported by Institute for Research Programmes from Sapientia Hungarian University of Transylvania.

References

[1] Claudio H. Rivetta, Charles Briegel, Paul Czarapata: Motion Control Design of the SDSS 2.5-m Telescope. in Proc. of SPIE, August 2000, pp. 212-221

[2] Brian Armstrong-Helouvry: Stick Slip and Control in Low-Speed Motion, IEEE Trans. on Automatic Control, October 1990, 38(10): 1483-1496

[3] Seung-Jean Kim, In-Joong Ha: On the Existence of Caratheodory Solutions in Mechanical Systems with Friction, IEEE Trans. on Automatic Control, November 1999, 44(11): 2086-2089

[4] Seung-Jean Kim, Sung-Yeol Kim, In-Joong Ha: An Efficient Identification Method for Friction in Single-DOF Motion Control Systems, IEEE Trans. on Control Systems Technology, July 2004, 12(4): 555-563

[5] D. Bi, F. Li, S. K. Tso, G. L. Wang: Friction Modeling and Compensation for Haptic Display-based on Support Vector Machine, IEEE Trans. on Industrial Electronics, April 2004, 51(2): 491-500

[6] Bernard Friedland: On Adaptive Friction Compensation. IEEE Trans. on Automatic Control, October 1992, 37(10):1609-1612

[7] Ashok Ramasubramanian, Laura Ray: Friction Cancellation in Flexible Systems using Extended Kalman-Bucy Filtering. In Proc. of the American Control Conference, Denver, Colorado, June 2003

[8] M. Verge: Friction Identification with Genetic Algorithms, In Proc. IFAC World Congress, Prague, July 2005

[9] Milos R. Popovic, Andrew A. Goldenberg: Modeling of Friction Using Spectral Analysis, IEEE Trans. on Robotics and Automation, 14 (1): 114-123, February, 1998

[10] Márton L, Béla L.: Tracking Control of Mechatronic Systems Based on Precise Friction Compensation, In Proc. 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, Timisoara, Romania, May 2006

[11] Márton L., Kutasi N.: Practical Identification Method for Striebeck Friction, In Proc 6th International Symposium of Hungarian Researchers in Computational Intelligence, Budapest, Hungary, November 2005

Human Behavior Model Based Control Program for ACC Mobile Robot

Claudiu Pozna, Fritz Troester

University Transilvania Brasov, Romania, cp@unitbv.ro
University of Applied Science Heilbronn, Germany, troester@hs-heilbronn.de

Abstract: Present work is a part of the ACC autonomous car project. This paper will focus on the control program architecture. To design this architecture we will start from the human driver behavior model. Using this model we have constructed a three level control program. Preliminary results are presented.

Keywords: Control Program, Behavior model, Autonomous car, Mobile robot

1 Introduction

One of the main projects of Applied Sciences University Heilbronn, Germany is named 'The Automotive Competence Centre (ACC)'. The aim of this project is to improve knowledge in the field of the car locomotion control engineering. More precisely, we intend to develop control algorithms which can be used to control an entire autonomous car. To achieve our purpose, we have transformed a car into a mobile robot [16], [17]. Knowing several results from this area ([9]...[14]), this development is made on the background of our institution's (University of Applied Sciences Heilbronn) main aim: the education of the future engineer.

Based on these remarks, we briefly sum up our conclusions concerning the developments of autonomous cars:

- Robust and intelligent algorithms have been developed to control an autonomous car. These algorithms involve a high level of mathematical resource, which must be implemented in real time applications;
- There are many projects made by powerful car companies. The results of these projects are: 'it can be done but it is too expensive', moreover, much knowledge of these projects can be used in the development of new systems;

After these conclusions, it is important to mention what kinds of benefit can be expected from our project. We can sum these up as follows:

- Information which will be used in the education of the future engineers by the University Heilbronn,
- Information about the design of a low price driving robot,
- Information about the implementations of high level control algorithms in real time applications,
- Development, design and testing of simple manoeuvres in real car driving like parking, etc.

In [16] we have presented the mechanical and electronically construction of the ACC mobile robot (see Figure 1).

Figure 1
The driving robot

The structure of the robot is presented in Figure 2 where the actuators (rectangles) and the sensors (circles) of the driving robot are plotted. More precisely: 1 is the actuator or the sensor for braking subsystem; 2 is the actuator or the sensor for steering subsystem; 3 is the actuator or the sensor for the gear level; 4 is the actuator respectively the sensor for ignition key; 5 is the actuator or the sensor for the gas; 6 is the ultrasonic sensor; 7 is the GPS. The robot ECU is the Electronic Control Unit which control the robot via CAN bus. In fact we replace the human driver by a mechatronic structure which is able to recognize the environment (with the extra sensors: ultrasonic sensor, GPS, etc.) and to drive the car (with the mentioned actuators, sensors and with the car ECU's).

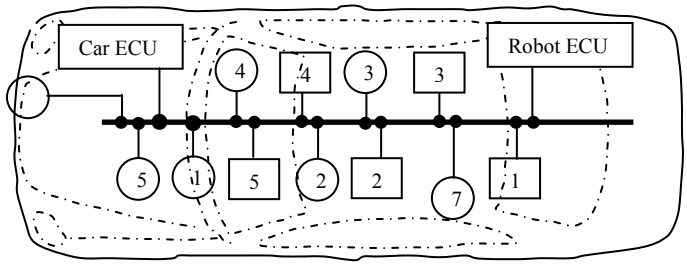


Figure 2
The robot structure

The next level in our project was to imagine and implement the soft which will control the robot under changeable driving circumstances. Present work focuses on the control program architecture design. The idea is to imagine architecture based on human driver decisions model.

2 The Driver's Behaviour Model

Some discussions about other ideas are necessary. From the references ([1]...[8]) we know that the 'Driver's Behaviour' model is used in the simulation field [3], [4] and also in autonomous or mixed manual and autonomous field [2]. The first researches on the subject started in 1950 [3] and began with the 'Skill-based driving model', continued with the 'Motivational model' which considers the drivers emotional state (from this class we can enumerate the 'Risk compensation'; 'Risk avoidance' and 'Risk threshold models') and in the last years is developed in a 'Hierarchical control structure' (by Milchon). The 'Hierarchical control structure' divides driving into three levels of control: a strategic level which establishes the goal of the driving; a tactical level which finds the solution to accomplish the goal; an operational level that implements this solution on the low level control of the vehicle. Behind this 'Hierarchical control structure' many scientific papers consider and develop problems like: 'Longitudinal behaviours models' [2]; 'Lateral behaviours model' [5], [1]; 'Brake behaviour' [2] etc. The solutions of these problems are varied: 'Linear optimal Control', 'Heuristic human driver models', 'Adaptive control strategy', 'Neuronal Network and fuzzy logic', 'Mental models', etc.

Because we intended to make a heuristic approach, we were interested to find control programs architectures which model the human behaviour. Such architecture is presented in [3] and [4]. Some conclusions about these briefly overviews are the following:

- In the scientific literature referring to 'Driver Behaviour Model' we have found several results which can be adapted and used in the ACC robot control;
- Recent works accept the Milchon three levels architecture;
- Many papers are focusing in developing the tactical level where the program must find the solutions in condition of changeable driving circumstance.

Model design is a possible solution of this problem. In this case the model is the 'human driver behaviour'. Our idea starts from this point: we consider that is more suitable to model, and implement the 'human driver decisions act' than the 'human driver actions'.

To obtain the model of human driver behaviour a preliminary analyze must answer to the following questions: '*how a common driver acts, or what is a driving behaviour?*'; '*can we obtain some fundamental true about this behaviour and use them in our construction?*' and more '*can we identify tools to transpose this behaviour in soft?'*

To answer to the first question we must give the definition of the 'behaviour' by underlining the semantic characteristics of 'Driving behaviour'. First it is important to establish the category tree of this word: from [15] we have {act → activity → (behaviour, practice,...)}. According to this the behaviour is: *'an action or a set of actions performed by a person under specified circumstances that reveal some skill, knowledge or attitude*'. From the scientific literature which concern the driving behaviour ([1]...[8]) and from our experience the driving behaviour has a special character. To describe this character we focus on the word 'custom' which is from the same category tree {act → activity → practice → custom,.} and which is defined like: '*accepted or habitual practice*'. In many situations these customs have a special nature: automatism: *any reaction that occurs automatically without conscious thought or reflection.* Now we can present what we understand by 'Driving Behaviour': *an action or a set of actions performed by a person under driving circumstances, action which tend to be transformed in customs and even in automatisms*; *in fact the 'Driving Behaviour' is composed from a collection of behaviours (the driver's behaviour when he makes the ignition, the driver's behaviour when he stops the car, ...).*

From the same theoretical and practical research, we establish the following '*fundamental truth*' for the 'Driving Behaviour':

1 *A priori* the driver establishes the current driving goal;

2 A behaviour is a set of actions;

3 These behaviours are linked together creating a system which allows to obtain solutions in the driving circumstance;

4 The translation from one behaviour to other is triggered by brow casting an event;

5 This system is developed by learning - experience;

6 Behaviours presume decisions with an incomplete set of information;

7 In time, these set of actions tend to be transformed in customs and automatisms.

These propositions are in accord with the well known three level architecture of Milchon: the strategically level where the driver establishes his goal, the tactical level where the driver finds the solution to accomplish the goal and the operational level where the driver implements these solutions. Using these propositions we can focus on the tactical level and model (approximate) the '*Human Driving Behaviour*' by a collection of high linked programs (behaviours) which are stored in a memory. The decision to run a certain program is made by a manager program. This decision is based on the goal of driving and acknowledging about the environment (driving circumstance). Each program (behaviour) is a succession of instructions (action) which impose parameters and trigger actuators. For a better understanding of this concept we will compare it with the well know Lego toys concept where several buildings (goals) can be made (solve the driving circumstance) using a finite type of bricks (program - behaviour). Using this analogy we will underline that it is very important to provide the interconnections of the bricks, and have an appropriate collection of them.

In what concerns the soft implementation, starting from these seven propositions, we can imagine the utility of state machine for handling the behaviours, fuzzy logic to enable the decisions based on incomplete information, or in describing the environment, and neuronal network to implement the learning processes.

After we have answer to the analyze questions a graphical representation of all these results will make our concept more understandable (see Figure 3)

Some explanations are necessary:

- The strategically level, where the robot must compute his goal is replaced with an interface where the human operator imposes the goal;
- The 'Program Manager' analyzes the goal in the driving circumstances which are obtained from the sensors; the result of this process is the status vector of the robot (the desired position, velocity, etc.) and the decision to run certain program from the 'Behaviours' subsystem;
- The 'Behaviours' are composed by three parts:
    - The 'Error Machine' which compares the status vector with state vector (the positions, velocity, etc. obtained from the sensors);
    - The 'Behaviours Programs': is a collection of programs (bricks); each program is able to solve a special environment situation (ignition, emergency stop, zero position, errors....);

- The 'Actuators Manager' which manage the actuators of the robot;

- The 'Output Interface' allows to the human operator to read the state vector and the errors of the robot and also memorizes the robot state history;
- The 'Actuators Communications' outputs data to the microcontrollers of each actuator.

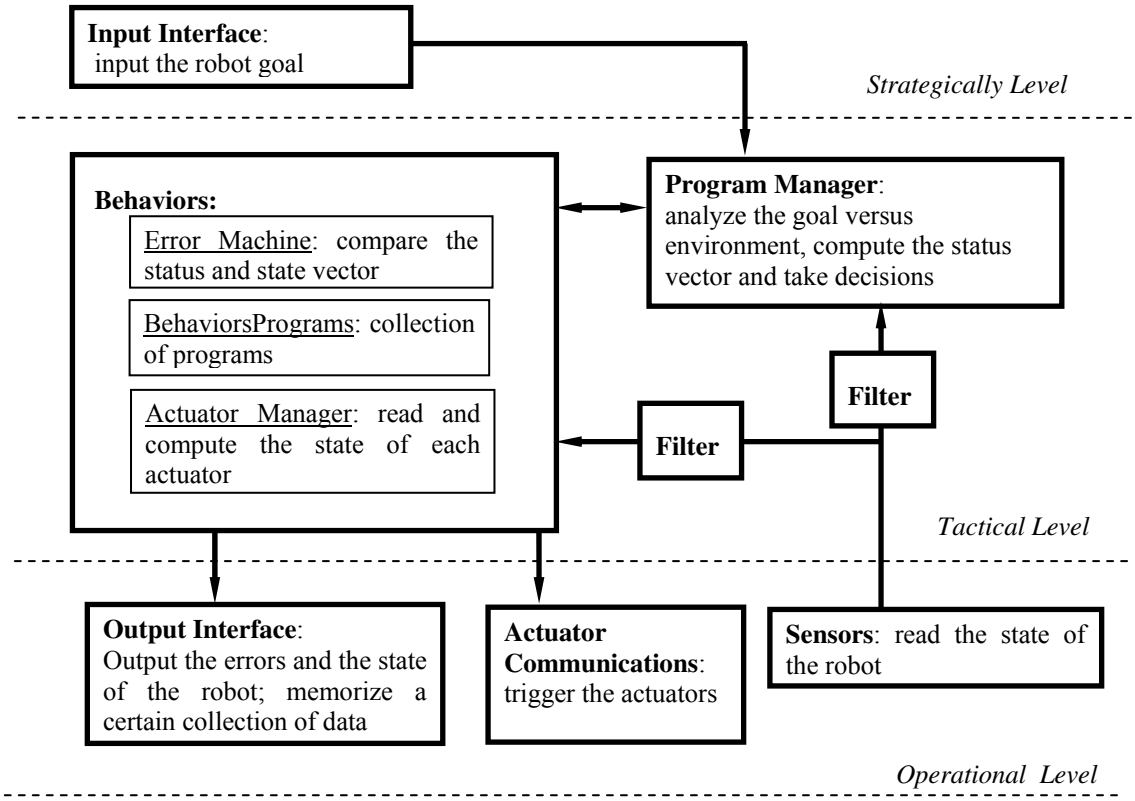


Figure 3
The program architecture

To build the 'Behaviours' subsystem it is important to imagine the structure of the programs (bricks). Understanding that this subsystem can, and will, be enriched in Figure 4 we propose three different structures, named: 'basic behaviours', 'error behaviour' and 'simple behaviour'. The main differences between these bricks are the connection type (P previous, N next, E error, QI quick in, QO quick out) and also the direction of information flow.

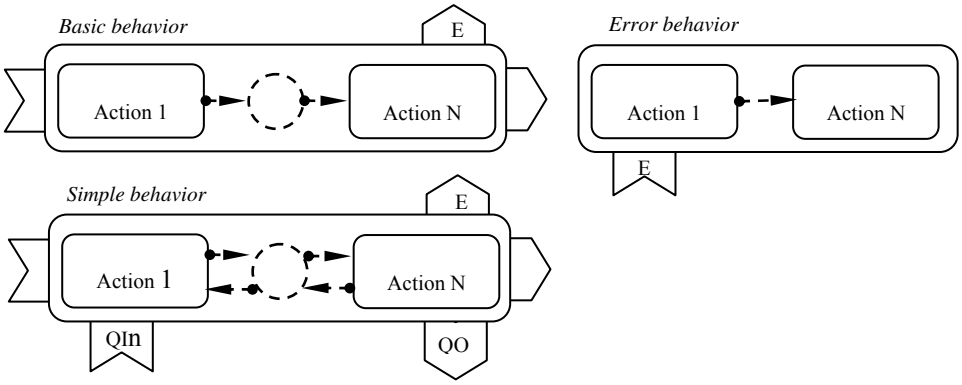


Figure 4
The behaviors structure (bricks)

In the 'Behaviours' subsystem an 'Error Machine' program is running. The aim of this program is to compare de 'status vector' (the desired variable of the robot: car speed, steering angle etc.) with the 'state vector' (the real variable read from the sensors: car speed, steering angle etc.). The decisions, about which brick must be connected, are made by the 'Program Manager'. This program compares the goal of the robot with the driving circumstance; establishes the status vector and enables the brick which must run. After these decisions the program continues to compare the robot goal with driving circumstance. If the result is acceptable, nothing is changed (the same brick is run), in contrary, a 'Crisis' or a 'Failure' event is broadcast. 'Crisis' means that a new behaviour is needed, so the status vector as well as the brick is changed. 'Failure' means that we don't have solutions (behaviours) to solve the problem and we must stop safely the robot. We present these processes using the diagram from Figure 5.

Conclusions about the program architecture are necessary:

- The structure of the program is a Milchon type structure;
- In this case the goal of the robot is imposed by the human operator via the 'Input interface';
- The tactical level finds solution, linking several programs (bricks). A brick is a succession of action; an action sets up parameters and trigger actuators;
- The human behaviour approximation consist in modelling the human decision process and not the human acting process;
- There are two control loops:
    - A high level control solved by the 'Manager Program': goal versus robot performance;

- Operational level control, low level control solved by each actuator microcontroller, desired value versus current value.

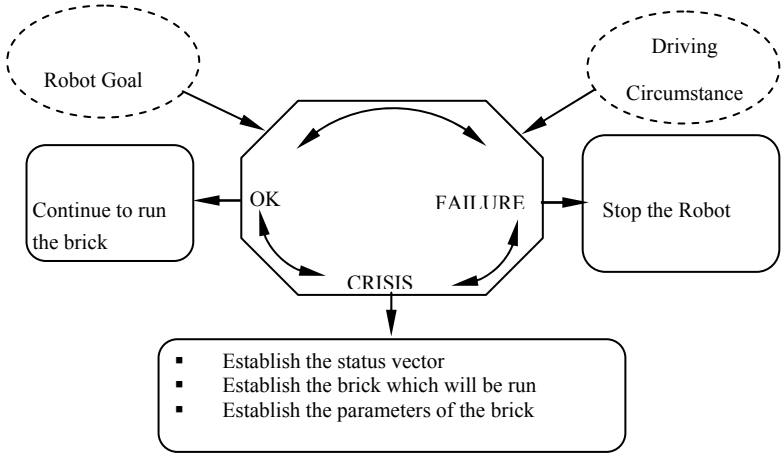


Figure 5
The Program Manager structure

3 Preliminary Results

This paper presents only partial results on our control program construction. The control program is made in Matlab and use the xPC toolbox. In order to interact via CAN with the already made mechanical and electronically parts we have started by creating the communication tools (see Figure 6). Using these tools we have created the operational level.

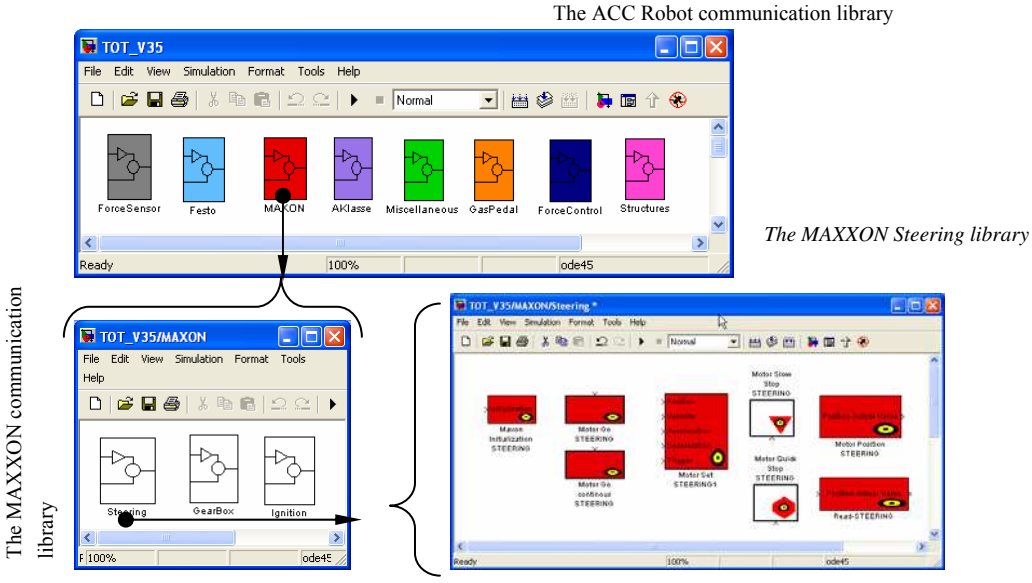


Figure 6
The ACC Robot communication library

In order to implement the 'Behaviours' structure in MATLAB we choose the 'Stateflow' toolbox. Some result is presented in Figure 7 where the three subsystems 'Error Machine', 'Behaviours' and 'Actuators Manager' are captured. The 'Error Machine' checks element by element the relation between the state and status vector. In the 'Behaviours Programs' are (in this stage of development) for bricks:

- 'Basic Behaviours': Initialization of actuators (the aim of this program is to enable the MAXXON actuators); Start (start the ignition of the car, and to change the gear level position from P to D); ZeroGo (put in zero position all the actuators: release the gas pedal; brake the car; turn the steer in position zero, change the gear level into position P; turn off the ignition key);
- 'Error Behaviours': ZeroE (put in the zero position all the actuators when an error event is brow cast).

These bricks build up a linked structure. In actual state of development the bricks are connected in a statically structure, the only possibility that the information flow change is an error event. Because the connections between the brick are already created, the aim of the future 'Program Manager' will be to enable the appropriate connection between the bricks.

The ‘Actuators Manager’ it is a parallel structure which permit the parallel use of all the actuators.

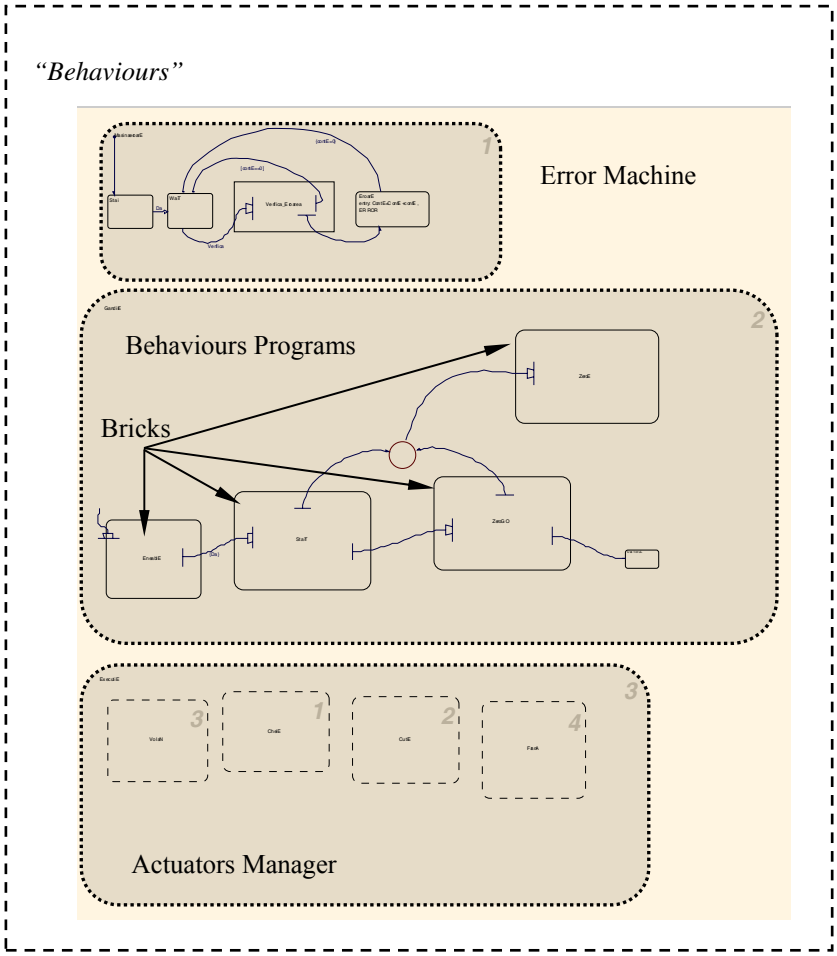


Figure 7
The actual state of construction of the ‘Behaviours’

The control program is plotted in Figure 8. In this picture it can be seen the connection between the state machine (‘Behaviours’) and the subsystems from the operational level named ‘Sensor Communication’ and ‘Actuators Communication’.

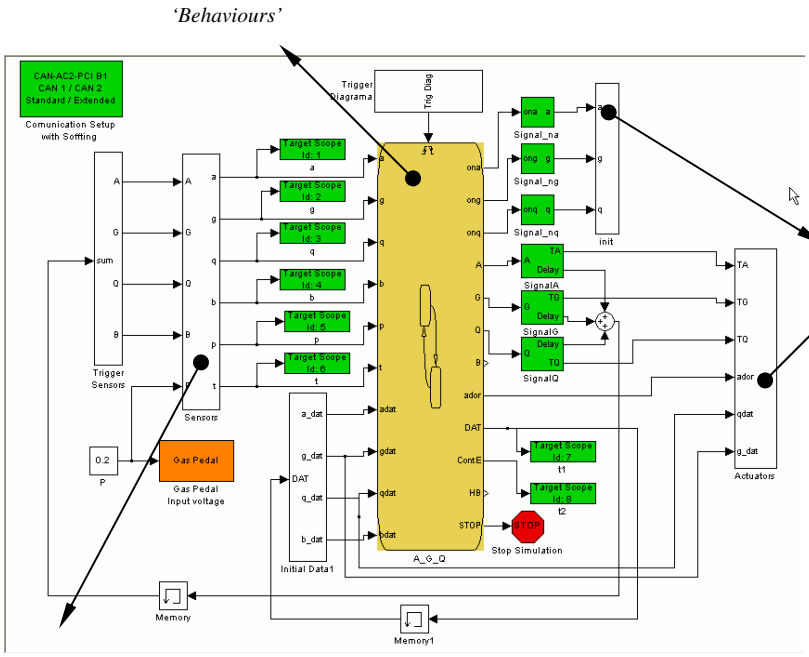


Figure 8
The control program made from Stateflow and Simulink objects

Conclusions

The aim of the paper was to present the control program architecture that we have imagined for the ACC mobile robot. After we have designed and realized the mechanical and electrically subsystem of the robot the next challenge was to implement the control program. Because the control task is complex, we intend to solve it approximating the human driver behaviour. More precisely we don't intend to model how the driver is steering, pushing the gas pedal etc. Our intention is to model the driver decisions process: the human (robot) decides to brake; the human (robot) driver decides to steer etc.

In actual state of work only a part of this construction is made: de operational level and the structure of the 'Behaviours' (a part of the tactical level). In this system we implemented several behaviours (bricks): start the car, enable the motor, put the car in zero position, etc. Future developments are focus in the tactical level: creating new bricks, creating the program manager, the filters etc.

References

[1] Mizukami, R.: Development of Autonomous Ground Vehicle Design of Control System. www.ee.ualberta.ca

[2] Bengtsson, J.: Adaptive Cruise Control and Driver Modeling. Department of Automatic Control Lund Institute of Technology Lund, November 2001

[3] Al-Shihabi, T., Mourant, R.: A Framework for Modeling Human-like Driving Behaviors for Autonomous Vehicle in Driving Simulators. Proceedings 5th International Conference on Autonomous Agents, June 2001, 286-291

[4] Quispel, L.: Automan, a Psychologically Based Model of Human Driver. Experimental and Work Pscychology. Department of Psychology, University of Groningen Grote Kruisstraat 2/1, 9751 MN Groningen

[5] Ungoren, A.: An Adaptive Lateral Preview Driver Model. Vehicle System Dynamic

[6] Liu, A.: Modeling and Prediction of Human Driver Behavior. Proc. of the 9th International Conference on Human Computer Interaction, Aug, 2001

[7] Huang, S.: Design and Performance Evaluation of Mixed Manual and Automated Control Traffic. IEEE Transaction on Systems Man and Cybernetics Nov. 2000

[8] Salvucci, D.: Modeling Driver Behavior in a Cognitive Architecture. In Press, Human Factors, February 16, 2005

[9] Dobrescu, R.: Autovehicule Inteligente, Editura MATRIX ROM Bucuresti 1995

[10] Antsalkis, P.: An Introduction to Intelligent and Autonomous Control, Kluver Academic Publisher Norwell, 1993

[11] Ackerman, J.: Robust Control for Automatic Steering, Proc. Amer. Conf., San Diego, CA, 1990, pp. 795-800

[12] Layne, J. K.: Fuzzy Learning Control for Antiskid Braking System, IEEE Trans. on Control Systems Techn. Vol. 1, No. 2, June 1993, pp. 122-129

[13] Jang, J.: Neuro-Fuzzy and Soft Computing, Prentice Hall, 1997

[14] Kosko, B.: Neural Network and Fuzzy Systems, Prentice Hall, 1992

[15] http://www.wordreference.com/definition/behavior

[16] Troester, F.: The ACC Fahrautomat Project. The International Conference Robotica Timisoara 2004

[17] Pozna, C., Troester, F.: Simulate the Inverse Cinematic of the ACC Mobile Robot. SYROM 2005, Bucharest, 2005

Statistical Analyses and Artificial Neural Networks for Prognoses in Hepatitis C

Andreea Drăgulescu[1], Adriana Albu[2], Cătălin Gavriluță[3], Ştefan Filip[4], Karoly Menyhardt[5]

[1] Medicine and Pharmacology University, Timişoara, Romania, adragulescu@cardiologie.ro
[2] Department of Automation and Applied Informatics, Politehnica University, Timişoara, Romania, adriana.albu@aut.upt.ro
[3] Politehnica University, Timişoara, Romania, tr.trope@gmail.com
[4] Politehnica University, Timişoara, Romania, stefy.filip@gmail.com
[5] Politehnica University, Timişoara, Romania, karoly_m@cmpicsu.utt.ro

Abstract: This paper analyses a large database with hepatitis C virus infected patients. There are made a lot of statistical analyses on the records of this database in order to determine the evolution of biological parameters during the treatment. That's for what it was also implemented a system which offeres predictions about the same parameters, using artificial neural networks. The results of the statistical analyses and the predictions of the system indicate to the same conclusions. It encourages the use of such a system to facilitate the physicians' work.

Keywords: hepatitis C virus infection, statistical analysis, artificial neural networks

1 Introduction

Hepatitis C is a serious and frequent disease and its evolution has to be carefuly overseen during the treatment. Even the efficency of the hepatitis C treatment inproves continously, the burden of this infection will remain a major issue for the next several decades.

The patients fro this study have been kept under observation for 12 months to establish the treatment's influence on the evolution of the biological indicators. Three different treatment schemes have been instituted:

- Simple Interferon (IFN);
- Peg interferon α-2a;
- Peg interferon α-2b.

The biological parameters were determined every three months and their evolution in time was monitored, trying to establish the relations between the biological indicators values (*TGP, TGO, GGT, ARN VHC*) and time, on patient groups sampled on their answer to the treatment. There are six types of reactions to the treatment considered as *answer-code*: 0-responder, 1-no responder to IFN, 2-no responder to Peg IFN, 3-backslide in treatment with IFN, 4-backslide in treatment with Peg IFN, 5-recess any treatment.

The correlations *biological indicators – time – code of reaction* have been 3D represented as functions of *z=f(x,y)* type. The obtained results are presented in Chapter 2.

On the other hand, the article presents (in Chapter 3) an artificial neural network trained to predict the evolution of the biological indicators. By introducing the patients personal data, as well as the results of biological tests at the treatment start, the implemented system can indicate the evolution in time of the illness. The use of the neural network presents the same conclusions as the statistical analysis.

2 Statistical Analysis

The investigated population contained 193 patients registered at the Emergency Clinical Hospital in Timisoara, Gastroenterology Department between 2003 and 2005 (120 women and 73 men aged between 14 and 67 years in old).

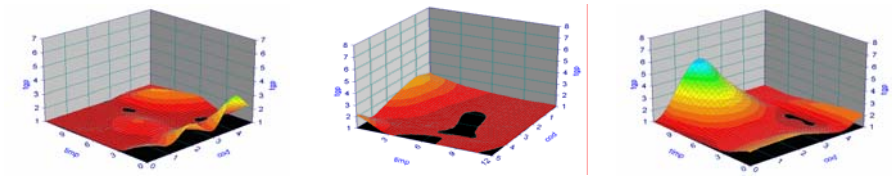
The study of the 3D correlations was made for the main biological indicators. For data processing the *TableCurve 3D* software was used. It can make the nonparametric interpolation of some 3D data multitudes by using the homogeneous grid method. Different calculating algorithms have been used (Akima [1], [2], [3], Bicubic [2], B-Spline [4], Preusser [6, 7], Renka [8], [9], [10], [11] and Watson [12]) and it has been observed that the Watson algorithm gave the biggest values of the correlation coefficient R^2. Therefore all the 3D representations and regressions established for the *z=f(x,y)* functions were represented by using this algorithm.

Even the information in the database is incomplete (there are patients which didn't follow the treatment for 12 months or were not periodically showed up for analyses) the indicated processing method can overpass it. Based on Watson algorithm the missing data have been calculated by extrapolation [12].

2.1 The TGP Biological Indicator Analysis

The *TGP=f(time, answer-code)* indicator analysis, as a answer- to the *IFN* treatment (Figure 1a), shows, for answer-code 0 patients, an emphasized decrease

of the *TGP* indicator in the first three months of treatment from the initial value *TGP_0=2.190* to the value *TGP_3=1.109,* followed by a milder decrease until the end of investigation where *TGP_12=1*. For the other answer-codes, the representation shows some differencies. For example, the maximum value of the *TGP* indicator *TGP_0= 3.525* for answer-code 4 and a smaller value *TGP_0=2.590* for answer-code 2. The represented surface allows the predicted evolution of all the answer-codes to be followed. The missing values can be obtained from the graphic representation. Therefore, for answer-code 5 for example, where there were values only for the initial phase and for the 3 month period, it can be observed that the representation gives calculated information even for the following months. For the 6 months phase the *IFN* treatment leads to a value *TGP_6=1.5,* then it rises to *TGP_9=1.8* and decreases again in the next period of time. Such values are readable on the graphic for all answer- codes in all temporal phases of the study.



a) patients treated with *IFN* b) patients treated with *Peg INF α-2a* c) patients treated with *Peg IFN α-2b*

Figure 1
The variation as function of time and answer code of the *TGP* indicator

The analysis of the same indicator for *Peg IFN α-2a* treatment (Figure 1b) shows a practically continuous decrease of the indicator's value from *TGP_0=2.018* to *TGP_12=1.093* for patients with answer-code 0.

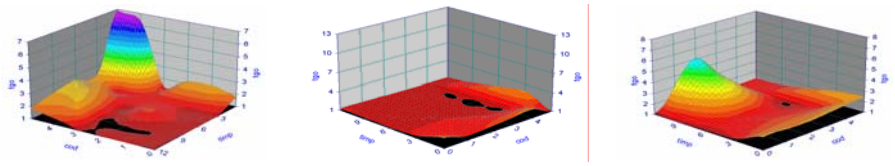
The representation verifies correctly enough the data for answer- code 4, where *TGP_0=1.190* and once with 6th month of the study it should stabilize at the indicator's value of 1. However, it can be observed from the representation that the technique of data extrapolation changes the *TGP* value in the period of 9-12 month. Therefore, at the end it is not obtained *TGP* measured value 1, but an estimated value of *1.008*, which represents an error of 0.8%. In conclusion, the verification of the existent data in the representation leads to the certitude that the estimated data are valid, having an acceptable error coefficient.

The same observations are valid for the surface *TGP=f(time, answer-code)* for the treatment with *Peg IFN α-2b* (Figure 1c). It can be observed that patients with answer- code 2 do not react to this treatment and that the indicator's value rises from *TGP_0=2.184* to *TGP_12=5.258*. For patients with answer- codes 1 and 3, for whom there is no information in the database, the representation in Figure 1b allows a relatively correct prediction, its correlation coefficient being $R^2=0.75$.

This way of approaching the problem makes the estimation of the *TGP* indicator's values a 70% true. It is calculated through the summing up of the probabilities induced by the correlation coefficients values and the statistical belief used.

2.2 The TGO Biological Indicator Analysis

The *TGO=f(time, answer code)* indicator analysis has been done in the same manner. It is represented in Figure 2 (a, b and c), for the three types of treatments.



a) patients treated with *IFN* b) patients treated with *Peg INF α-2a* c) patients treated with *Peg IFN α-2b*

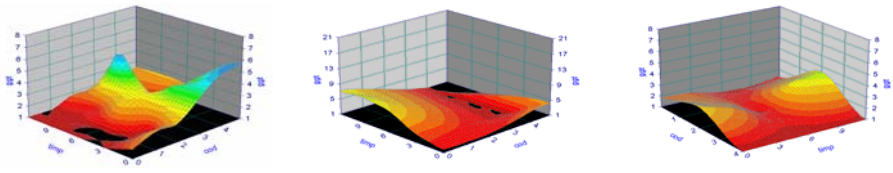
Figure 2
The variation as function of time and answer code of the *TGO* indicator

By comparing Figures 2b and 2c it can be observed that the *Peg IFN α-2a* treatment is more efficient than the *Peg IFN α-2b* treatment because the decrease of the indicator's value is continuous for the former and it is predicted a lower final value than for the latter. The representations allow the *TGO* indicator's value to be assessed for patients with answer-codes 3 and 5 which are not present in the database and which have been estimated through the Watson algorithm. These data are very important, especially for the answer-code 5 patients that have stopped the treatment because of collateral effects.

2.3 The GGT Biological Indicator Analysis

The *GGT=f(time, answer-code)* indicator was similarly analyzed. The obtained graphical representations are presented in Figure 3 (a, b and c) for the three types of treatment.

It is proved that the *Peg IFN α-2b* treatment is the most efficient, as the *GGT* indicator's value decreases from *GGT_0=4.974* to *GGT_12=1.314*. The same treatment has a fluctuant influence on the code 2 patients. In the first 6 months the treatment leads to a decrease value of the indicator from *GGT_0=2.843* to *GGT_6=1.158,* followed by a come back to *GGT_12=3.472*.



a) patients treated with *IFN* b) patients treated with *Peg INF α-2a* c) patients treated with *Peg IFN α-2b*

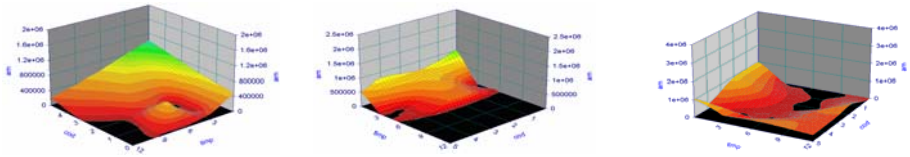
Figure 3
The variation as function of time and answer code of the *GGT* indicator

The data in Figure 3b definitely show the inefficiency of the *Peg IFN α-2a* treatment over the value of the *GGT* indicator for answer-code 0 patients. The representation presents an estimation of the reaction of the answer-code 2 patients, for whom there was no data starting with the 6th month of the study. For the same patients, it is estimated the value *GGT_12=5*, the correlation coefficient of the representation being $R^2=0.74$. Answer-code 2 patients are no-responders to the *PegIFN* treatment, therefore it is important to know what is the predicted value of the indicator on the areas in which the database is incomplete.

2.4 The ARN VHC Biological Indicator Analysis

The variation of the *ARN VHC=f(time, answer-code)* indicator is represented in Figure 4 (a, b and c) for the three types of treatment. Regarding the evolution of this indicator, all treatments are efficient.

It is observed that the patients which have been no-responders to the *IFN* treatment (answer-code 1) have a positive evolution in the first three months of treatment, followed by a decrease in the next period, but at a lower value of the indicator than the initial. From now the missing data can be completed by predicting that an eventual continuation of the treatment could lead to a significant decrease of the *ARN VHC* value. As a result, the patient would become a responder.



a) patients treated with *IFN* b) patients treated with *Peg INF α-2a* c) patients treated with *Peg IFN α-2b*

Figure 4
The variation as function of time and answer code of the *ARN VHC* indicator

In addition to this, the figure gives probable information of the indicator's values for answer-code 3 patients, information that does not exist in the database. For these, with a probability of 65%, the initial value of the indicator could be *ARN VHC_0≅ 750000*. So, this information could be taken into consideration together with another indicators, in order to attach a code, as realistic as possible, to the patients considered in a future study. Figures 4b and 4c are useful for the prediction of the values of the *ARN VHC* indicator for patients with answer-codes 4 and 5, for whom the database is also incomplete.

3 Predictions using Artificial Neural Networks

The same conclusions as before regarding the evolution of the biological indicators during the hepatitis C treatment have been obtained throw the implementation of a system based on artificial neural networks. This system will predict the values of the *TGP*, *TGO*, *GGT* and *ARN VHC* after 3, 6, 9, and 12 months of treatment.

Artificial neural networks are a branch of the artificial intelligence and they have been developed to reproduce human reasoning and intelligence. The initial idea was that, in order to reproduce intelligence, it would be necessary to build systems with architecture similar to the brain one. Therefore, artificial neural networks are built by the interconnection of certain primary elements, whose structure is similar to the biological neuron. Like the human brain, these artificial neural networks are able to recognize patterns, manage data and, most important, they have the ability of learning [5] (ability to adapt to the informational environment specific to the problem they are solving).

The system presented here was developed using feed-forward neural network with back-propagation learning algorithm. Such a network receives a series of inputs and its outputs are the results of the problem. Between the two levels (input and output) there can be a number of hidden levels. The elements on each level (neurons) are interconnected throw links called synapses. These have a weight, which can be modified along the training of the network.

In this case, the system has been designed as a network of neural networks. Each neural network has a layer of 10 hidden neurons, a single output unit and a variable number of inputs. For each of the four biological indicators that have been studied, there are four layers of neural networks. The networks on the first layer receive as inputs: patient's age, sex, location (rural/urban), treatment scheme, Knodell score, hepatic fibrosis score and value of the parameter for which the prediction is made, at the initial moment (before the treatment start). These networks have as output the value of the biological parameter at 3 months. On the following layers the networks have the same structure as the first layer ones, but

they have in addition, as inputs, the outputs of the networks on the former layers; therefore, the networks on the last layer will have not 7 inputs (as the networks on the first layer) but 10 (the initial inputs and the values of biological indicators at 3, 6, and 9 months). Figure 5 describes the architecture of this neural network.

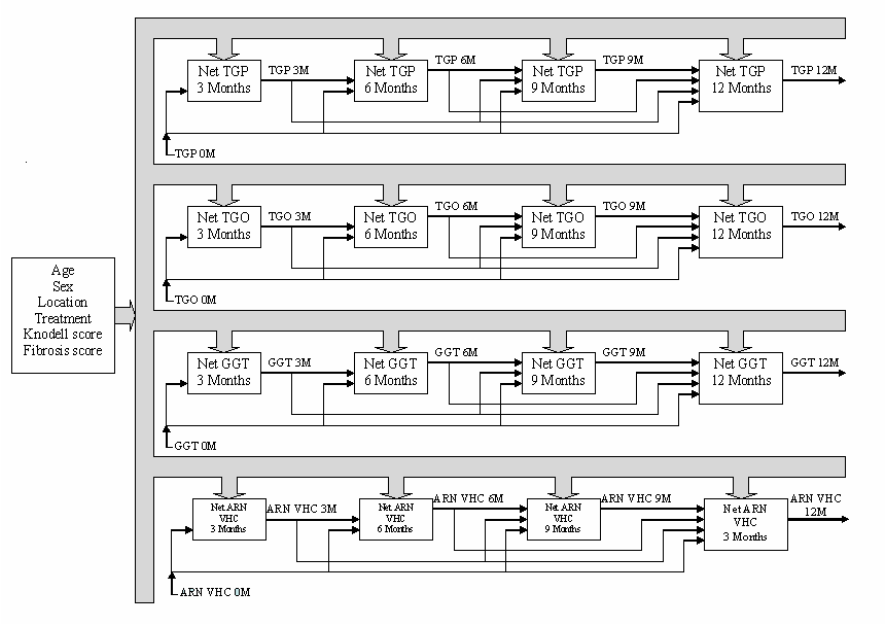


Figure 5
The architecture of the artificial neural network

The advantage of this architecture is that the input data are processed separate for each biological indicator. The disadvantage is that the errors are propagated through the system because the result of the networks from the first level (together with their errors) are used in the following levels. But this disadvantage can be minimised by lerning process.

The application has been projected in the Matlab 7.0 environment, which has a toolbox totally dedicated to the neuronal calculus. The system offers for each evaluated biological indicator predictions regarding the next 12 months evolution, indicating its growing tendency, its stabilizing or decreasing tendency (Figure 6).

The user has to choose a range regarding the age of the patient, the sex, the location where the patient lives (rural/urban), the treatment (IFN, Peg interferon α-2a or Peg interferon α-2b) and has to introduce the values of the Knodell score and of the fibrosis score. It is also necessary to introduce the values of the biological indicators before the treatment.

Looking at the predicted tendency of the biological indicators during the treatment, a physician can estimate if the patient will respond to a treatment or not.

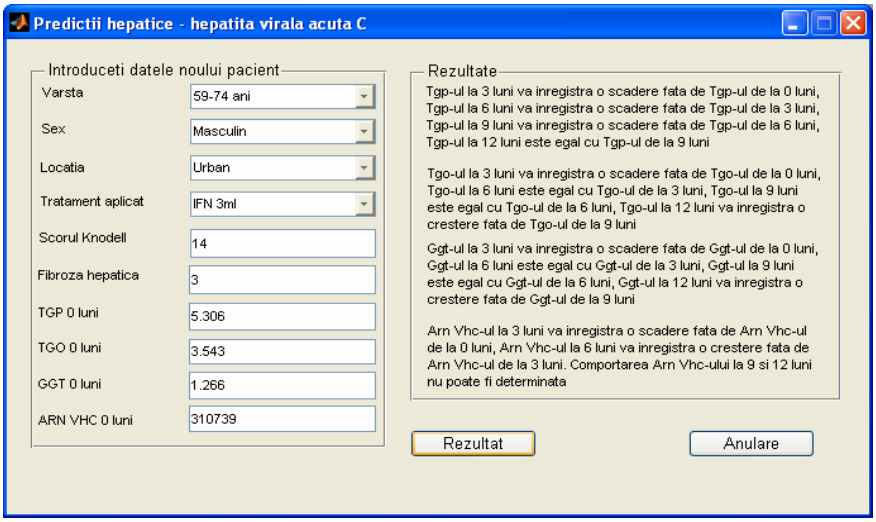


Figure 6
The prediction of biological indicators evolution

Conclusions

The statistical analysis, as well as, the implemented system offers the possibility to predict the evolution of patients in time. The hepatitis C treatment is very expensive and severe side effects can appear very often. Therefore, it is important to identify those patients who most probably can react to the treatment, so that the others can be protected from a treatment with no benefits. That's for what the use of such a system can support the physician decision concerning the treatment.

References

[1] Akima H.: A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points, ACM Transactions on Mathematical Software, Vol. 4, 1978, pp. 144-159

[2] Akima H.: Algorithm 760: Rectangular-Grid-Data Surface Fitting that has the Accuracy of a Bicubic Polynomial, ACM Transactions on Mathematical Software, Vol. 22, No. 3, Sept. 1996, pp. 357-361

[3] Akima H.: Algorithm 761: Scattered-Data Surface Fitting that has the Accuracy of a Cubic Polynomial, ACM Transactions on Mathematical Software, Vol. 22, No. 3, Sept. 1996, pp. 362-371

[4] De Boor C: A Practical Guide to Splines, Springer-Verlag, 1978, pp. 332-346

[5] Maiellaro P. A., Cozzolongo R., Marino P.: Artificial Neural Networks for the Prediction of Response to Interferon Plus Ribavirin Treatment in Patients with Chronic Hepatitis C, Current Pharmaceutical Design, 2004, Vol. 3, pp. 2101-2109

[6] Preusser A.: Efficient Formulation of a Bivariate Nonic C2-Hermite Polynomial on Triangles, ACM Transactions on Mathematical Software, Vol. 16, No. 3, Sept. 1990, pp. 246-252

[7] Preusser A.: Algorithm 684: C1 and C2-Interpolation on Triangles with Quintic and Nonic Bivariate Polynomials, ACM Transactions on Mathematical Software, Vol. 16, No. 3, Sept. 1990, pp. 253-257

[8] Renka R: Algorithm 660: QSHEP2D: Quadratic Shepard Method for Bivariate Interpolation of Scattered Data, ACM Transactions on Mathematical Software, Vol. 14, No. 2, June 1988, pp. 149-150

[9] Renka R: Multivariate Interpolation of Large Sets of Scattered Data, ACM Transactions on Mathematical Software, Vol. 14, No. 2, June 1988, pp. 139-148

[10] Renka R: Algorithm 751: TRIPACK, A Constrained Two-Dimensional Delaunay Triangulation Package, ACM Transactions on Mathematical Software, Vol. 22, No. 1, March 1996, pp. 1-8

[11] Renka R: Algorithm 752: SRFPACK, Software for Scattered Data Fitting with a Constrained Surface under Tension, ACM Transactions on Mathematical Software, Vol. 22, No. 1, March 1996, pp. 9-17

[12] Watson D.: Nngride, An Implementation of Natural Neighbor Interpolation, David Watson, P.O. Box 734, Claremont, WA 6010, Australia, 1994

Iterative Feedback Tuning in Fuzzy Control Systems. Theory and Applications

Stefan Preitl, Radu-Emil Precup

Department of Automation and Applied Informatics,
"Politehnica" University of Timisoara
Bd. V. Parvan 2, RO-300223 Timisoara, Romania
E-mail: stefan.preitl@aut.upt.ro, radu.precup@aut.upt.ro

János Fodor

Institute of Intelligent Engineering Systems, Budapest Tech
Bécsi út 96/B, H-1034 Budapest, Hungary
E-mail: fodor@bmf.hu

Barnabás Bede

Department of Mechanical and System Engineering, Budapest Tech
Népszínház utca 8, H-1081 Budapest, Hungary
E-mail: bede.barna@bgk.bmf.hu

Abstract: The paper deals with both theoretical and application aspects concerning Iterative Feedback Tuning (IFT) algorithms in the design of a class of fuzzy control systems employing Mamdani-type PI-fuzzy controllers. The presentation is focused on two-degree-of-freedom fuzzy control system structures resulting in one design method. The stability analysis approach based on Popov's hyperstability theory solves the convergence problems associated to IFT algorithms. The suggested design method is validated by real-time experimental results for a fuzzy controlled nonlinear DC drive-type laboratory equipment.

Keywords: Iterative Feedback Tuning, PI-fuzzy controllers, experiments

1 Introduction

PI and PID controllers are widely used in more then 80% of industrial applications worldwide due to the good control system (CS) performance they offer [1]. Since the main tasks in control, the achievement of good CS performance in reference input tracking and the regulation in the presence of disturbance inputs, are difficult to be accomplished by means of PI and PID controllers in one-degree-of-freedom structures, an alternative is to develop two-degree-of-freedom (2-DOF) controllers which have advantages over the one-degree-of-freedom ones [2, 3]. However, the main drawback of 2-DOF control structures in the linear case is that although they ensure regulation, the reduction of the overshoot is paid by slower responses with respect to the modification of reference input. The presentation in the paper will be concentrated on the 2-DOF PI controller case.

Another solution with to ensure good CS performance in the conditions of complex or even ill-conditioned plants is represented by fuzzy control. The development of fuzzy control systems (FCSs) is usually performed by heuristic means, incorporating human skills, with the drawback in the lack of general-purpose design methods. A major problem, which follows from this way to design fuzzy controllers (FCs) is the analysis of several properties of the FCS including stability, controllability, parametric sensitivity and robustness [4, 5].

If low cost automation solutions are required then systematic design methods devoted to relatively simple FCs. One approach is to design firstly 2-DOF PI controllers for the plants characterized by simplified linearized models. Then, the transfer of results from the linear case to the fuzzy one resulting in 2-DOF PI-fuzzy controllers (PI-FCs) is done in terms of the modal equivalence principle [6] accepting the well acknowledged equivalence in certain conditions between FCSs and linear / linearized CSs [7].

Iterative Feedback Tuning (IFT) [8, 9] is a gradient-based approach, based on input-output data recorded from the closed-loop system. The CS performance indices are specified through certain cost functions (c.f.s). Optimizing such functions usually requires iterative gradient-based minimization, implemented as IFT algorithms, observing that the c.f.s can be complicated functions of the plant and of the disturbances dynamics. The key feature of IFT is that the closed-loop experimental data are used to compute the estimated gradient of the c.f. Several experiments are performed iteratively and the updated controller parameters are obtained based on the closed-loop input-output data obtained during system operation.

In this context, the aim of combining IFT algorithms with fuzzy control in terms of transferring the results from the linear case to the fuzzy one is to obtain new and attractive low cost fuzzy control solutions ensuring FCS performance enhancement.

One problem associated to any gradient-based optimization algorithm is related to its convergence. Solving this problem is done in a transparent way by the stability analysis. The paper presents a stability analysis approach based on Popov's hyperstability theory. This approach results in sufficient stability conditions that guarantee the convergence of the IFT algorithms.

The paper is organized as follows. An overview on the IFT algorithms used in tuning the linear 2-DOF PI controllers is presented in Section 2. Then, Section 3 is focused on a new design method for a class of Mamdani-type two-degree-of-freedom PI-fuzzy controllers (2-DOF PI-FCs). Section 4 deals with the stability analysis of the considered class of FCSs in order to guarantee the convergence of IFT algorithms. Section 5 is dedicated to the validation of the method by applying it in one case study regarding the speed control of a nonlinear DC drive-type laboratory equipment, and the last Section 6 to conclusions.

2 Overview on Iterative Feedback Tuning Algorithms

Two versions of equivalent control system structures can be used in case of 2-DOF control structures to ensure either simultaneous tuning of controller parameters [10] or their separate tuning for each of the controller blocks [11]. The first version will be presented as follows, with the nomenclature according to Fig. 1(a), where the two controller blocks are characterized by the transfer functions $C_r(s)$ and $C_y(s)$. The basic details in the 2-DOF control structure case are presented in [12].

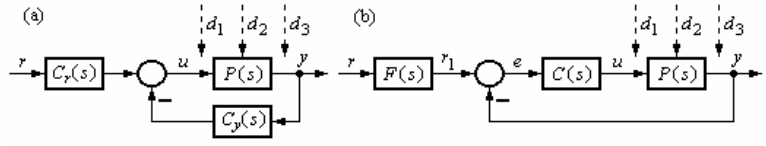


Figure 1
2-DOF control system structure used in IFT (a), and used feedforward filter (b)

The IFT method consists of the steps A) … E) to be presented in the sequel in relation with the 2-DOF PI controller having the structure presented in Fig. 1(b), where: $C(s)$ – transfer function of the main PI controller:

$$C(s) = k_c(1+sT_i)/s = k_C[1+1/(sT_i)],\ k_C = T_i k_c, \tag{1}$$

with k_C – controller gain and T_i – integral time constant, $F(s)$ – transfer function of the feedforward filter:

$$F(s) = 1/(1+T_i s), \tag{2}$$

r – reference input, y – controlled output, $e = r_1 - y$ – control error, u – control signal, r_1 – output of block $F(s)$ (filtered reference input), d_1, d_2, d_3 – load disturbance input scenarios, with the general disturbance input $d \in \{d_1, d_2, d_3\}$, and the connections between the controller blocks in Fig. 1(a) and (b) are:

$$C_r(s) = C(s)F(s),\ C_y(s) = C(s). \tag{3}$$

In these conditions, the steps of the IFT approach are:

A) A controller, of desired complexity, which stabilizes the system, has to be chosen. A discrete form of the controller is needed. The parameterization of the controller is such that the transfer functions $C_r(s, \boldsymbol{\rho})$ and $C_y(s, \boldsymbol{\rho})$ are differentiable with respect to its parameters, $\boldsymbol{\rho}$ being the parameter vector.

For the sake of highlighting the controller tuning parameters, the parameter vector $\boldsymbol{\rho}$ has been added as additional input variable to the transfer function. This nomenclature will be used in the sequel in both continuous- and discrete-time not only for the transfer functions but also to the variables regarding the plant (the control signal u and the controlled output y).

B) A reference model must be chosen, prescribing the desired CS behaviour observed in y. This model is typically chosen with first- or second-order dynamics and, for the sake of simplicity and better CS performance, it can be also chosen to be without dynamics, having the transfer function equal to the unity.

C) The general expression of the c.f. J is proposed in (4) in the framework of this optimization problem:

$$\boldsymbol{\rho}^* = \arg\min_{\boldsymbol{\rho} \in SD} J(\boldsymbol{\rho}),\ J(\boldsymbol{\rho}) = \frac{1}{2N} \cdot \sum_{k=1}^{N} \{[L_y(q^{-1})\, \delta y(k, \boldsymbol{\rho})]^2 + \lambda\, [L_u(q^{-1}) u(k, \boldsymbol{\rho})]^2\}, \tag{4}$$

where: N – length of each experiment, L_y, L_u – weighting filters, introduced to emphasize certain frequency regions, λ – weighting constant, δy – output error, the difference between the actual output (y) and the desired output (y_d):

$$\delta y = y - y_d. \tag{5}$$

D) The update law must be set by which the next set of parameters will be computed. This law corresponds usually to a Gauss-Newton scheme of type (6), other versions being also used to avoid the computation of second-order derivatives:

$$\boldsymbol{\rho}^{i+1} = \boldsymbol{\rho}^i - \gamma^i (\mathbf{R}^i)^{-1} est\left[\frac{\partial J}{\partial \boldsymbol{\rho}}(\boldsymbol{\rho}^i)\right], \tag{6}$$

where: i – index of the current iteration, $est[x]$ – estimate (generally) of the variable x, $\gamma^i > 0$ – parameter to determine the step size.

E) The regular matrix $\mathbf{R}^i$ in (6) is a positive definite matrix, usually the Hessian of $J(\boldsymbol{\rho})$:

$$\mathbf{R}^i = \frac{1}{N}\sum_{k=1}^{N}\left(est\left[\frac{\partial y}{\partial \boldsymbol{\rho}}(k,\boldsymbol{\rho}^i)\right] est\left[\frac{\partial y}{\partial \boldsymbol{\rho}}(k,\boldsymbol{\rho}^i)\right]^T + \lambda\, est\left[\frac{\partial u}{\partial \boldsymbol{\rho}}(k,\boldsymbol{\rho}^i)\right]\cdot \right. \\ \left. \cdot est\left[\frac{\partial u}{\partial \boldsymbol{\rho}}(k,\boldsymbol{\rho}^i)\right]^T \right) \tag{7}$$

Choosing the identity matrix for $\mathbf{R}^i$ ensures the negative direction of the gradient, but it is recommended to compute $\mathbf{R}^i$ by a quasi-Newton method or as the Hessian of the c.f.

IFT algorithms are employed to implement to solve the optimization problem (4), where several additional constraints can be imposed regarding the plant or the closed-loop system. One necessary constraint concerns the stability of the closed-loop system, and *SD* in (4) stands for stability domain. In addition, the expression of the c.f. can be modified by adding quadratic terms with the output sensitivity functions defined in the time domain, accordingly weighted to reduce the sensitivity of the CS with respect to the parametric variations of the controlled plant (see the situation in [13] for FCSs).

The iterative character of IFT algorithms in case of 2-DOF PI controllers considered here results from three real-time experiments performed with the CS, the first and the third one referred to as normal ones and the second one referred to as the gradient one. The normal experiments are characterized by the reference input fed to the CS while in case of the gradient experiment the role of reference input fed to the CS is played by the control error in the first experiment. The input-output data recorded from these three experiments are employed to compute the estimated gradients of the controlled output and of the control signal required in computing the estimated gradient of the c.f. $J(\boldsymbol{\rho})$.

The IFT algorithms considered here contain the steps 1 … 8 to obtain the next set of parameters:

Step 1 The three experiments are done and the input-output data (u_1, y_1), (u_2, y_2) and (u_3, y_3) are recorded.

Step 2 The output of the reference model is generated, y_d, and the output error δy is computed by (5).

Step 3 The estimated gradient of the output is computed based on the data recorded form the real-time experiments. Before applying this approximation, the sensitivity function S and the complementary sensitivity function T must be

expressed in discrete-time, with the following definitions according to the CS structure illustrated in Fig. 1(a):

$$\begin{aligned} &S(q^{-1},\boldsymbol{\rho}) = 1/[1+P(q^{-1})C_y(q^{-1},\boldsymbol{\rho})], \\ &T(q^{-1},\boldsymbol{\rho}) = P(q^{-1})C_r(q^{-1},\boldsymbol{\rho})/[1+P(q^{-1})C_y(q^{-1},\boldsymbol{\rho})]. \end{aligned} \tag{8}$$

The analytical expression of the gradient of δy is obtained using (8) and taking the derivatives with respect to $\boldsymbol{\rho}$ [10]:

$$\begin{aligned} est\left[\frac{\partial \delta y}{\partial \rho}(k,\boldsymbol{\rho})\right] = \frac{1}{C_r(q^{-1},\boldsymbol{\rho})}\cdot\left[\frac{\partial C_r}{\partial \boldsymbol{\rho}}(q^{-1},\boldsymbol{\rho})y_3(k,\boldsymbol{\rho}) - \right. \\ \left. -\frac{\partial C_y}{\partial \boldsymbol{\rho}}(q^{-1},\boldsymbol{\rho})y_2(k,\boldsymbol{\rho})\right]. \end{aligned} \tag{9}$$

Step 4 The control signal is a perfect realization of the control signal in the first experiment:

$$u = u_1 . \tag{10}$$

Step 5 The estimated gradient of the control signal u is computed using (8), (10) and taking the derivatives with respect to $\boldsymbol{\rho}$:

$$\begin{aligned} est\left[\frac{\partial u}{\partial \rho}(k,\boldsymbol{\rho})\right] = \frac{1}{C_r(q^{-1},\boldsymbol{\rho})}\cdot\left[\frac{\partial C_r}{\partial \boldsymbol{\rho}}(q^{-1},\boldsymbol{\rho})u_3(k,\boldsymbol{\rho}) - \right. \\ \left. -\frac{\partial C_y}{\partial \boldsymbol{\rho}}(q^{-1},\boldsymbol{\rho})u_2(k,\boldsymbol{\rho})\right]. \end{aligned} \tag{11}$$

Step 6 The c.f. $J(\boldsymbol{\rho})$ is computed according to (4) with the estimated and its estimated gradient results as follows:

$$\begin{aligned} est\left[\frac{\partial J}{\partial \boldsymbol{\rho}}(\boldsymbol{\rho})\right] = \frac{1}{N}\sum_{k=1}^{N}\{L_y(q^{-1})\delta y(k,\boldsymbol{\rho})est\left[\frac{\partial \delta y}{\partial \boldsymbol{\rho}}(k,\boldsymbol{\rho})\right] + \\ + \lambda L_u(q^{-1})u(k,\boldsymbol{\rho})est\left[\frac{\partial u}{\partial \boldsymbol{\rho}}(k,\boldsymbol{\rho})\right]\}, \end{aligned} \tag{12}$$

with the values of the estimated gradients obtained as steps 3 and 5.

Step 7 The matrix $\mathbf{R}^i$ is computed in terms of (7).

Step 8 The next set of parameters is obtained by a Gauss-Newton scheme according to (6).

3 Design Method for Mamdani-type Two-degree-of-freedom PI-fuzzy Controllers

The structure of 2-DOF PI fuzzy controller is constructed by fuzzifying the main linear PI controller with the transfer function $C(s)$ in Fig. 1(b). The 2-DOF PI-FC, with the structure presented in Fig. 2, represents a discrete-time controller involving a basic fuzzy controller without dynamics (B-FC). The dynamics is inserted by the numerical differentiation of the control error e_k expressed as the increment of control error, $\Delta e_k = e_k - e_{k-1}$, and by the numerical integration of the increment of control signal, Δu_k.

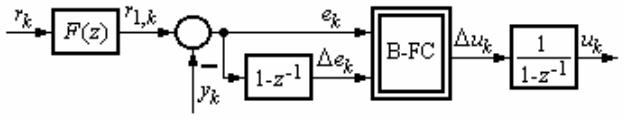


Figure 2
Structure of two-degree-of-freedom PI-fuzzy controller

B-FC is a nonlinear two inputs-single output system which includes among its nonlinearities the scaling of inputs and output as part of its fuzzification module. The fuzzification is solved in terms of the regularly distributed input and output membership functions presented in Fig. 3. Other distributions of the membership functions can modify in a desired way the controller nonlinearities. The inference engine in B-FC employs Mamdani's MAX-MIN compositional rule of inference assisted by the rule base presented in Table 1, and the centre of gravity method for singletons is used for defuzzification.

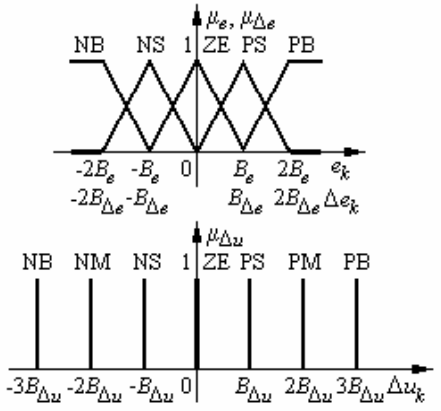


Figure 3
Membership functions of B-FC in Fig. 2

Table 1
Decision table of B-FC

Δe_k	e_k				
	NB	NS	ZE	PS	PB
PB	ZE	PS	PM	PB	PB
PS	NS	ZE	PS	PM	PB
ZE	NM	NS	ZE	PS	PM
NS	NB	NM	NS	ZE	PS
NB	NB	NB	NM	NS	ZE

Other binary operators can be introduced instead as t-norms and t-conorms instead of the MIN and MAX operators, respectively. Several representations of uninorms [14], nullnorms [15] or distance-based operators [16] can be used with this respect resulting in several versions of inference engines that can lead to FCS performance enhancement after serious analyses [17].

The design method for this class of Mamdani-type 2-DOF PI-FCs consists of the following design steps:

Step A Design the continuous-time 2-DOF PI controller by a specific method to the linear case depending on the class of considered controlled plants (in simplified mathematical models for the design) and on the desired / imposed CS performance indices.

Step B Set the value of the sampling period T_s chosen in accordance with the requirements of quasi-continuous digital control, express the discrete-time equation of the-point filter $F(z)$, the discrete-time equation(s) of the digital PI controller $C(z)$ in its incremental version:

$$\Delta u_k = K_P \cdot \Delta e_k + K_I \cdot e_k = K_P(\Delta e_k + \alpha \cdot e_k), \tag{13}$$

and compute the parameters $\{K_P, K_I, \alpha\}$, exemplified in (14) in case of Tustin's method:

$$K_P = k_C[1 - T_s/(2T_i)],\ K_I = k_C T_s / T_i,\ \alpha = K_I / K_P = 2T_s/(2T_i - T_s). \tag{14}$$

Step C Apply the modal equivalence principle that enables the computation of two fuzzy controller parameters, $B_{\Delta e}$ and $B_{\Delta u}$:

$$B_{\Delta e} = \alpha B_e, B_{\Delta u} = K_I B_e, \tag{15}$$

where the third parameter, B_e, represents designer's option.

4 Stability Analysis Method

To perform the stability analysis of the FCS the controlled plant must be transformed to the following *n*-th order discrete-time SISO linear time-invariant state mathematical model (MM) including the zero-order hold:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A} \cdot \mathbf{x}_k + \mathbf{b} \cdot u_k \\ y_k &= \mathbf{c}^T \cdot \mathbf{x}_k \end{aligned}, \tag{16}$$

where: u_k – control signal; y_k – controlled output; $\mathbf{x}_k$ – state vector, dim $\mathbf{x}_k = (n,1)$; $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}^T$ – matrices having the dimensions as follows: dim $\mathbf{A} = (n, n)$, dim $\mathbf{b} = (n, 1)$, dim $\mathbf{c}^T = (1, n)$. In this context it is necessary to transform the initial FCS structure into a multivariable one because B-FC (see Fig. 2) represents a two inputs-single output system. This modified FCS structure is illustrated in Fig. 4(a), where the dynamics of the fuzzy controller is transferred to the controlled plant resulting in the extended controlled plant (ECP, a linear one). The vectors in Fig. 4(a) have the following representation: $\mathbf{r}_k$ – reference input vector, $\mathbf{e}_k$ – control error vector, $\mathbf{y}_k$ – controlled output vector, $\mathbf{u}_k$ – control signal vector. Generally speaking, in both discrete- and continuous-time, the index k may be omitted, and these vectors are defined in (17):

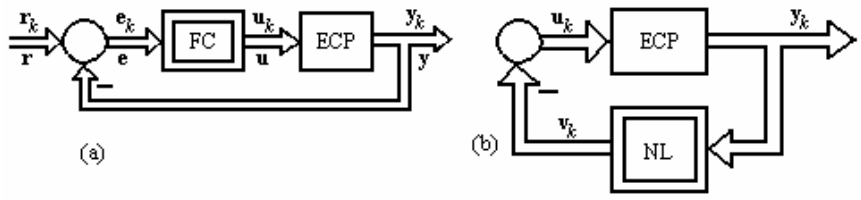


Figure 4
Modified structure of FCS (a) and structure used in stability analysis (b)

$$\mathbf{r}_k = \begin{bmatrix} r_k & \Delta r_k \end{bmatrix}^T, \ \mathbf{e}_k = \begin{bmatrix} e_k & \Delta e_k \end{bmatrix}^T, \ \mathbf{y}_k = \begin{bmatrix} y_k & \Delta y_k \end{bmatrix}^T, \tag{17}$$

with the general notation $\Delta v_k = v_k - v_{k-1}$ for the increment of v_k.

The block FC in Fig. 4(a) is characterized by the nonlinear input-output static map $\mathbf{F}$:

$$\mathbf{F} : R^2 \to R^2, \ \mathbf{F}(\mathbf{e}_k) = [f(\mathbf{e}_k), \ 0]^T \ , \tag{18}$$

where $f : R^2 \to R$ is the input-output static map of B-FC in Fig. 2.

All variables in Fig. 4(a) have two components, and this requires the introduction of a fictitious control signal, supplementary to the outputs of B-FC, to obtain equal numbers of inputs and outputs as required by the hyperstability theory in the multivariable case. In addition, the structure involved in the stability analysis of an unforced nonlinear control system ($\mathbf{r}_k = \mathbf{0}$ and the disturbance inputs are also zero)

is presented in Fig. 4(b). The block NL in Fig. 4(b) represents a static nonlinearity due to the nonlinear part without dynamics of the block FC in Fig. 4(a). The connections between the variables of the two control system structures in Fig. 4 are:

$$\mathbf{v}_k = -\mathbf{u}_k = -\mathbf{F}(\mathbf{e}_k),\ \mathbf{y}_k = -\mathbf{e}_k, \tag{19}$$

with the second component of **F** being always zero to neglect the effect of the fictitious control signal.

The MM of the ECP can be derived by firstly defining the additional state variables $\{x_{uk},x_{yk}\}$ according to [18], and the (n+2)-th order discrete-time state MM of the ECP can be expressed as:

$$\begin{aligned} \mathbf{x}_{k+1}^E &= \mathbf{A}^E \cdot \mathbf{x}_k^E + \mathbf{B}^E \cdot \mathbf{u}_k^E \\ \mathbf{y}_k^E &= \mathbf{C}^E \cdot \mathbf{x}_k^E \end{aligned}, \tag{20}$$

with the matrices $\mathbf{A}^E$ (dim $\mathbf{A}^E$ = (n+2,n+2)), $\mathbf{B}^E$ (dim $\mathbf{B}^E$ = (n+2,2)) and $\mathbf{C}^E$ (dim $\mathbf{C}^E$ = (2,n+2)):

$$\mathbf{A}^E = \begin{bmatrix} \mathbf{A} & \mathbf{b} & \mathbf{0} \\ \mathbf{0}^T & 1 & 0 \\ \mathbf{c}^T & 0 & 0 \end{bmatrix},\ \mathbf{B}^E = \begin{bmatrix} \mathbf{b} & \mathbf{1} \\ 1 & 1 \\ 0 & 1 \end{bmatrix},\ \mathbf{C}^E = \begin{bmatrix} \mathbf{c}^T & 0 & 0 \\ \mathbf{c}^T & 0 & -1 \end{bmatrix}. \tag{21}$$

Simple computations from the second equations in (19) and (20) lead to:

$$\mathbf{e}_k = -\mathbf{C}^E \cdot \mathbf{x}_k,\ \mathbf{x}_k = \mathbf{C}^b \cdot \mathbf{e}_k, \tag{22}$$

where the matrix $\mathbf{C}^b$, dim $\mathbf{C}^b$ = (n+2,2), can be calculated relatively easy as function of $\mathbf{C}^E$.

The core of the suggested stability analysis method can be expressed in terms of one theorem offering a certain sufficient stability condition [18]. This theorem states that nonlinear system, with the structure presented in Fig. 4(b) and the MM of its linear part (20), is globally asymptotically stable if the three matrices **P** (positive definite, dim **P** = (n+2,n+2)), **L** (regular, dim **L** = (n+2,n+2)) and **V** (any, dim **V** = (n+2,2)) fulfil (23):

$$\begin{aligned} (\mathbf{A}^E)^T \cdot \mathbf{P} \cdot \mathbf{A}^E &= -\mathbf{L} \cdot \mathbf{L}^T \\ \mathbf{C}^E - (\mathbf{B}^E)^T \cdot \mathbf{P} \cdot \mathbf{A}^E &= \mathbf{V}^T \cdot \mathbf{L}^T \\ -(\mathbf{B}^E)^T \cdot \mathbf{P} \cdot \mathbf{B}^E &= \mathbf{V}^T \cdot \mathbf{V} \end{aligned}, \tag{23}$$

introducing the matrices **M** (dim **M** = (2,2)), **N** (dim **N** = (2,2)) and **R** (dim **R** = (2,2)) defined in (24):

$$\begin{aligned} \mathbf{M} &= (\mathbf{C}^b)^T \cdot (\mathbf{L} \cdot \mathbf{L}^T - \mathbf{P}) \cdot \mathbf{C}^b \\ \mathbf{N} &= (\mathbf{C}^b)^T \cdot [\mathbf{L} \cdot \mathbf{V} - (\mathbf{A}^E)^T \cdot \mathbf{P} \cdot \mathbf{B}^E - 2(\mathbf{C}^E)^T] \\ \mathbf{R} &= \mathbf{V}^T \cdot \mathbf{V} \end{aligned}, \tag{24}$$

the Popov-type inequality (25) holds for any value of the control error e_k:

$$f(\mathbf{e}_k) \cdot \mathbf{n}^T \cdot \mathbf{e}_k + (\mathbf{e}_k)^T \cdot \mathbf{M} \cdot \mathbf{e}_k \geq 0\,, \tag{25}$$

where **n** represents the first column in **N**.

For a given value of PI-FC parameter B_e the stability analysis methbod is concentrated on checking the stability condition (25) for any values of PI-FC inputs in operating regimes considered as significant to FCS behaviour.

5 Case Study

The case study considered in this Section aims the validation of the suggested design method dedicated to the new class of 2-DOF PI-fuzzy controllers presented in Section 3. The case study is focused on a fuzzy controller design for the class of plants with the transfer function $P(s)$ characterizing simplified mathematical models used in servo system control as part of mechatronics systems and embedded systems:

$$P(s) = k_P / [s(1 + T_\Sigma s)]\,, \tag{26}$$

where k_P is the controlled plant gain and T_Σ is the small time constant or an equivalent time constant as sum of parasitic time constants.

One solution to control this class of plants is represented by PI control [1]. A simple and efficient way to tune the parameters of the 2-DOF PI controller controlling the plant (16) is represented by the Extended Symmetrical Optimum (ESO) method [19], characterized by only one design parameter, β. The choice of the parameter β within the domain $1 < \beta < 20$, leads to the modification of the CS performance indices (σ_1 – overshoot, $\hat{t}_r = t_r / T_\Sigma$ – normalized rise time, $\hat{t}_s = t_s / T_\Sigma$ – normalized settling time defined in the unit step modification of r, φ_m – phase margin) according to designer's option and to a compromise to these performance indices using the diagrams presented in Fig. 5 in the situation without feedforward filter. The presence of the feedforward filter with the transfer function $F(s)$ improves the CS performance indices.

The PI tuning conditions, specific to the ESO method, are:

$$k_c = 1/(\beta\sqrt{\beta} T_\Sigma^2 k_P),\ T_i = \beta T_\Sigma\,. \tag{27}$$

These tuning conditions highlight the presence of only design one parameter, β, which simplifies the application of the IFT algorithms mentioned in Section 2 because the parameter vector becomes a scalar:

$$\boldsymbol{\rho} = \beta\,. \tag{28}$$

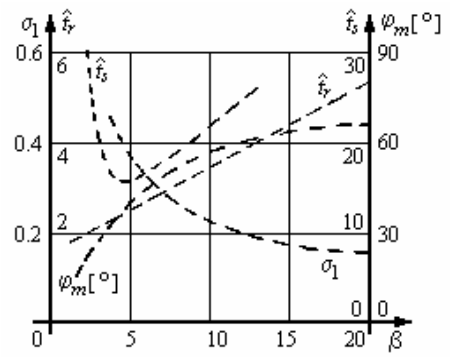


Figure 5
Control system performance indices versus β in the situation without feedforward filter

The experimental setup consists of speed control of a nonlinear laboratory DC drive (AMIRA DR300). The DC motor is loaded using a current controlled DC generator, mounted on the same shaft, and the drive has built-in analog current controllers for both DC machines having rated speed equal to 3000 rpm, rated power equal to 30 W, and rated current equal to 2 A. The speed control of the DC motor is digitally implemented using an A/D-D/A converter card. The speed sensors are a tacho generator and an additional incremental rotary encoder mounted at the free drive-shaft. The block diagram of the hardware station is presented in Fig. 6.

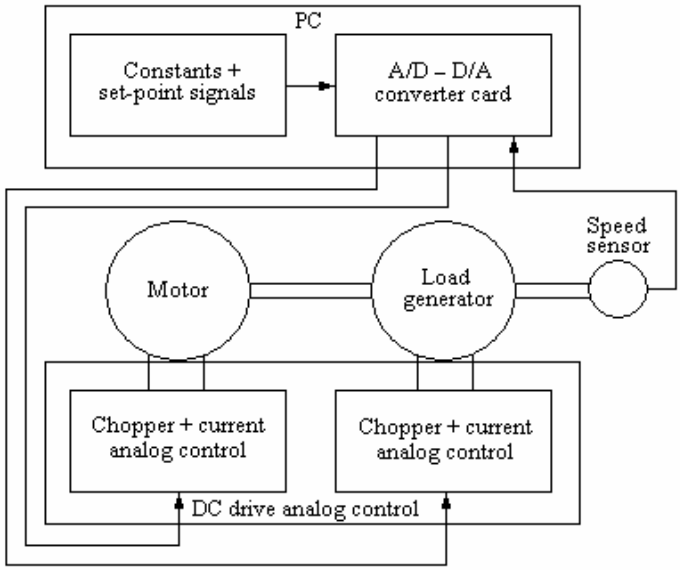


Figure 6
Block diagram of hardware station

The mathematical model of the plant can be well approximated by the transfer function $P(s)$ in (26), with $k_P = 4900$ and $T_\Sigma = 0.035$ s. The development method proposed in the previous Section is applied, and for the sake of simplicity only the main parameter values are presented. The method starts with the choice of the initial value of the design parameter $\beta = 6$. Then, a version of IFT algorithm presented in Section 2 is applied in the condition of the following c.f., J:

$$J = \frac{1}{2N}\sum_{k=1}^{N}(\delta y^2(k)), \tag{29}$$

in the conditions of a third-order reference model corresponding to the closed-loop linear system for the same value of β and a smaller value of T_Σ. The following 'optimal' values of the PI-FC tuning parameters have been obtained after six iterations: $B_e = 0.3$, $B_{\Delta e} = 0.03$, $B_{\Delta u} = 0.0021$, for $\beta^* = 5.76$. These values guarantee the FCS stability according to Section 4.

Part of the real-time experimental results – the variations of r and y versus time – are presented in Fig. 7 and Fig. 8 for the linear CS (with 2-DOF PI controller) and for the fuzzy CS (with 2-DOF PI-FC) in the conditions:

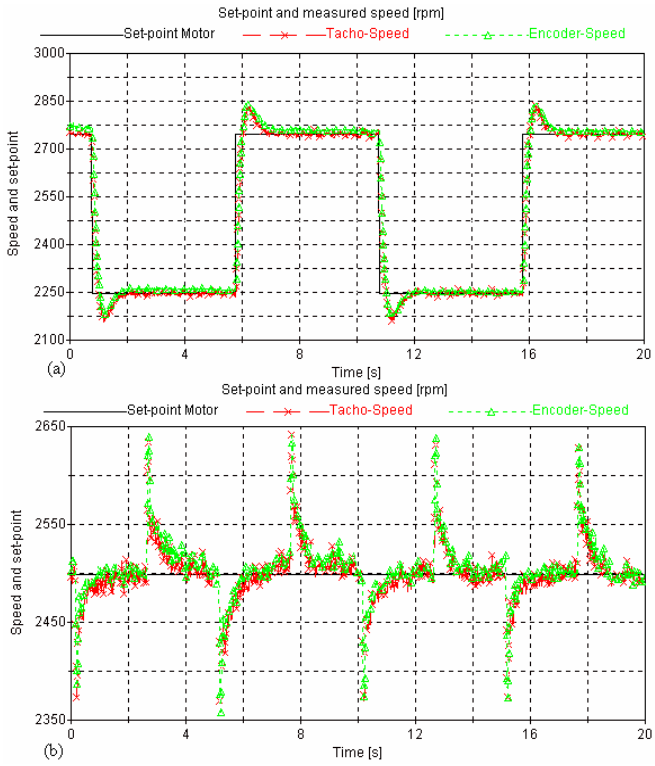


Figure 7
Control system response with 2-DOF PI controller

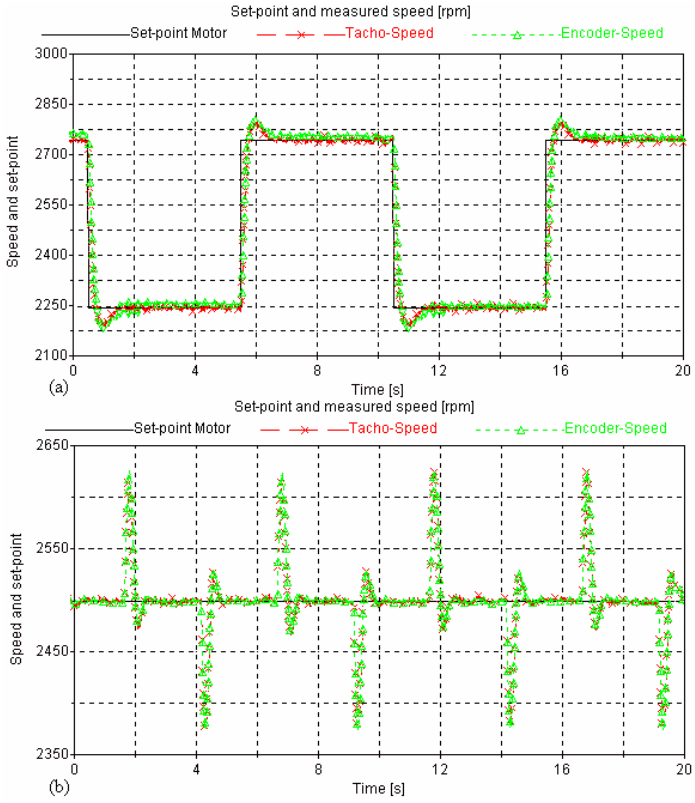


Figure 8
Control system response with 2-DOF PI-fuzzy controller

- without load in Fig. 7(a) and Fig. 8(a),
- 5 s period of 10% rated $d = d_2$ type load and $r = 2500$ rpm in Fig. 7(b) and Fig. 8(b).

Conclusions

The paper presents theoretical and application aspects concerning Iterative Feedback Tuning algorithms employed in the design of a class of fuzzy control systems with Mamdani-type PI-fuzzy controllers. It is proposed a new design method for the PI-fuzzy controllers validated by real-time experiments related one fuzzy control solution dedicated to a class of plants applied in servo systems as part of mechatronics systems and embedded systems.

The design method illustrates the potential of IFT employed in connection with fuzzy control in complex plants. The convergence of the IFT algorithms is guaranteed if the stability analysis method suggested in the paper is applied.

Future research will focus on the on-line implementation of IFT algorithms to control other laboratory equipment. This concerns discrete-event systems including robots and manufacturing systems [20, 21, 22].

Acknowledgement

The support stemming from the cooperation between Budapest Tech Polytechnical Institution and "Politehnica" University of Timisoara (PUT) in the framework of the Hungarian-Romanian Intergovernmental Science & Technology Cooperation Program no. 35 ID 17, from two CNCSIS grants, and from the cooperation in automatic control between University of Bremen and PUT is acknowledged.

References

[1] K.-J. Åström, T. Hägglund: PID Controllers Theory: Design and Tuning, Instrument Society of America, Research Triangle Park, NC, 1995

[2] M. Araki, H. Taguchi: Two-degree-of-freedom PID Controllers, in Int. Journal of Control, Automation, and Systems, Vol. 1, 2003, pp. 401-411

[3] A. Visioli: A New Design for a PID Plus Feedforward Controller, in Journal of Process Control, Vol. 14, 2004, pp. 457-63

[4] D. Driankov, H. Hellendoorn, M. Reinfrank: An Introduction to Fuzzy Control, Springer-Verlag, Berlin, Heidelberg, New York, 1993

[5] K. M. Passino, S. Yurkovich: Fuzzy Control, Addison-Wesley, Reading, MA, 1998

[6] S. Galichet, L. Foulloy: Fuzzy Controllers: Synthesis and Equivalences, in IEEE Transactions on Fuzzy Systems, Vol. 3, 1995, pp. 140-148

[7] B. S. Moon: Equivalence between Fuzzy Logic Controllers and PI Controllers for Single Input Systems, Fuzzy Sets and Syst., Vol. 69, 1995, pp. 105-113

[8] H. Hjalmarsson, S. Gunnarsson, M. Gevers: A Convergent Iterative Restricted Complexity Control Design Scheme, in Proc. 33rd IEEE Conf. on Decision and Control, Lake Buena Vista, FL, 1994, pp. 1735-1740

[9] H. Hjalmarsson, M. Gevers, S. Gunnarsson, O. Lequin: Iterative Feedback Tuning: Theory and Applications, in IEEE Control Systems Magazine, Vol. 18, 1994, pp. 26-41

[10] O. Lequin, M. Gevers, M. Mossberg, E. Bosmans, L. Triest: Iterative Feedback Tuning of PID Parameters: Comparison with Classical Tuning Rules, in Control Engineering Practice, Vol. 11, 2003, pp. 1023-1033

[11] K. Hamamoto, T. Fukuda, T. Sugie: Iterative Feedback Tuning of Controllers for a Two-mass-spring System with Friction, in Control Engineering Practice, Vol. 11, 2003, pp. 1061-1068

[12] C. Ardelean, A. Graeser, M. Mihajlov, R.-E. Precup: Applications of Iterative Feedback Tuning in PID Controller Design, in Buletinul Stiintific al Universitatii "Politehnica" din Timisoara, Transactions on Automatic Control and Computer Science, Vol. 50 (64), No. 1, 2005, pp. 11-16

[13] R.-E. Precup, S. Preitl: Optimisation Criteria in Development of Fuzzy Controllers with Dynamics, in Engineering Applications of Artificial Intelligence, Vol. 17, 2004, pp. 661-674

[14] J. Fodor, R. Yager, A. Rybalov: Structure of Uninorms, in International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 5, 1997, pp. 411-427

[15] T. Calvo, B. De Baets, J. Fodor: The Functional Equations of Frank and Alsina for Uninorms and Nullnorms, in Fuzzy Sets and Systems, Vol. 120, 2001, pp. 385-394

[16] I. Batyrshin, O. Kaynak, I. Rudas: Fuzzy Modeling based on Generalized Conjunction Operations, in IEEE Transactions on Fuzzy Systems, Vol. 10, 2002, pp. 678-683

[17] M. Takács: Approximate Reasoning in Fuzzy Systems Based on Pseudo-Analysis, PhD Thesis, University of Novi Sad, 2004

[18] R.-E. Precup, S. Preitl: Stability and Sensitivity Analysis of Fuzzy Control Systems. Mechatronics Applications, in Proc. 6th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, 2005, pp. 130-143

[19] S. Preitl, R.-E. Precup: An Extension of Tuning Relations after Symmetrical Optimum Method for PI and PID Controllers, in Automatica, Vol. 35, 1999, pp. 1731-1736

[20] L. Horváth, I. J. Rudas, O. Kaynak: Virtual Manufacturing Oriented Generic Manufacturing Process Model, in Recent Advances in Mechatronics, Springer-Verlag, 1999, pp. 336-348

[21] L. Horváth, I. J. Rudas, C. Couto: Integration of Human Intent Model Descriptions in Product Models, in Digital Enterprise - New Challenges Life-Cycle Approach in Management and Production, Kluwer Academic Publishers, 2001, pp. 1-12

[22] L. Horváth, I. J. Rudas, J. F. Bitó, A. Szakál: Adaptive Model Objects for Robot Related Applications of Product Models, in Proc. 2004 IEEE International Conference on Robotics & Automation ICRA 2004, New Orleans, LA, USA, 2004, pp. 3137-3142

A Multi-Agent Framework for Execution of Complex Applications

Alexandru Cicortas, Victoria Iordan

Computer Science Department
Faculty of Mathematics and Computer Science
University of the West Timisoara
e-mail: cico@info.uvt.ro, iordan@info.uvt.ro

Abstract: Complex applications execution needs a lot of conditions starting with hardware and software resources until the task sequencing and verifying the results of the execution. The needed resources can be found locally or on the Web and Grid. For an efficient usage the needed resources application and the effective resources found must be described in an appropriate and way and we propose the XML. The matching between the resources of the application and the Web/Grid resources is done in our proposal by the agents. As a new approach in the last years is the Service Oriented Architecture (SOA) that is developed as an efficient solution. The services are furnished by the providers and these are used by the clients in some specific conditions. The realization of the SOA technology presents a real opportunity to improve effectiveness. On the Web and Grid we dispose for services and these can be used in a given contest. SOA provides the application of well-founded concepts that exploits the ability of modern system resources to collaborate, independent of location across heterogeneous technologies.

Keywords: multi-agent systems, grid, resource management, complex systems design, service oriented architecture.

1 Introduction

Real world systems grow in complexity. The execution of complex applications implies adequate tools and also support for execution. The Web and Grid offer an efficient way for executing the requirements of companies that must interact in a very complex manner.

The grid model, with its use of resources tends to be heterogeneous and distributed across multiple management domains. For example, in a grid environment shared resources must remain accessible, not but that the hardware configuration is dynamic, the resources can be available in a time instant, but in an another time instant these can disappear. In the grid key infrastructure services must be available, and virtual organizations must be maintained. It must also be

possible to detect report and deal with faults that may occur in any of the member domains.

Effective system management is only possible if resources are manageable, and if tools are available to manage them. However, these tools tend to operate independently and to use proprietary interfaces and protocols to manage a limited set of resources, making it difficult for an organization to build an efficient, well-integrated management system. In order to allow the development of manageability standards that will enable conforming management tools to manage conforming resources in a uniform manner, and to interoperate with each other. In turn this will enable system administrators to choose their management tools and suppliers in the knowledge that, regardless of their origin, the tools can work cooperatively in an integrated management environment.

As stated in [15] the resource management includes:

- reservation, brokering and scheduling;
- installation, deployment and provisioning;
- metering;
- aggregation (service groups, WSDM collections, etc.);
- VO management;
- security management;
- monitoring (performance, availability, etc.);
- control (start, stop, etc.);
- problem determination and fault management.

Some of other major problems are to use and discovery the resources.

Beside resources, the grid offers also services. These services can be simple services or complex ones and use resources. Like resources the services must be represented in an adequate manner in order to be found and used. In many grid organizations the registers are used in order to furnish the resources and services that can be used or invoked.

The problem of execution of complex applications that must be solved in an efficient manner using the grid requires a lot of complex tools. Some of aspects of the problem concern:

- formulating the problem that is proposed to be solved;
- expressing an adequate decomposition of the application in the tasks;
- for every task to express the hardware and software resources that are needed;

- giving an appropriate expression of the task sequencing and conditioning;
- defining the services that will be invoked;
- searching and discovering on the grid for the appropriate resources and services;
- if there is the case to allow the migration of the software agents (SA) on the grid and also to install the needed (adequate) software;
- defining the data that must be partially hosted for the future tasks (temporary storage of partial data or needed data for recovering in the case of failure);
- defining the recoveries when some failure occurs.

The execution of such grid application can be expressed in terms of workflow. This subject will be detailed in another paper. Concerning the workflow of the complex application on the grid, there are many works like [4], [16] that describe how the workflow must be designed.

There exist a lot of works that threat partially or totally such a kind of approach in the grid context. The GLOBUS [11] UNICORE [18] and the works done for the Fraunhofer Resource Grid [10] solve in some partially sense the above requirements.

Being in many cases a complex problem the jobs execution needs to overcome some of the exception situations as rollbacks and resuming in the incidents. In a framework it implies some additional activities like:

- localization of the failed task;
- analysis of the task execution;
- the resuming of the task from an intermediary point if there exists a such point;
- resuming the data bases implied in the same node or other nodes.

There exist many frameworks that allow to the application designers to dispose for:

- tools, standards and languages to define express and use the information concerning the evolution of the application during its execution;
- an adequate Grid framework that allows to:
    - explore and find the grid resources;
    - establish in the grid context, the conditions for the task execution and in the failure cases the conditions for resuming the task (like the temporary storage of data);
    - launching and controlling the task execution;

- in order to obtain the desired results the framework must allow the rollbacks.

We propose also a framework.

Multi-agent systems are used in many domains and their strength allows developing complex systems. The agent abilities i.e., self adaptation, learning from own experience or from collectivity experience, the communication and collaboration are very expressive tools in order to reach the system goals.

The following concepts [20] are closely related:

- *Choreography* describes required patterns of interaction among services and templates for sequences (or more structures) of interactions;
- *Orchestration* describes the ways in which business processes are constructed from Web services and other business processes, and how these processes interact;
- *Workflow* is a pattern of business process interaction, not necessarily corresponding to a fixed set of business processes. All such interactions may be between services residing within a single data center or across a range of different platforms and implementations anywhere.

The XML is used for expressing the appropriate information that can be processed using various platforms in most recent initiatives concerning the usage of adequate standards that are applied in many domains. In many of real projects the XML [22] is an adequate mean in order to allow expressing the needs and it in platform independent. Based on the previous experience in the following the XML will be used.

The paper proposes:

- a framework that uses the XML that allows to express and furnish the required information to the designer. The information concerns the grid its resources, its services, its location and also the details of the application concerning its decomposition in tasks, the sequencing of the tasks and consequently the resources and services needed for tasks execution;
- a multi-agent system as a solution for solving such kind of problems;
- a workflow model for the execution of the applications in the grid context. It will be the subject of another work and papers due the particularities of the workflow.

The paper is structured as follows. The information used is shortly revised in the next Section. The third Section presents the concepts and tools used for modeling. The fourth Section, propose the multi-agent system as a possible solution. In Section 5 the framework is sketched and some details are given. The last Section contains some conclusions and future works.

2 The Information for the Grid Resources

Some of the resource Grid components usually are not described in the necessary detailed form in order that its description can be efficient used, i.e., all its attributes that specify if the resource or service can be used by certain service or software. For overtaking it we must define a function that allows to the user to receive the desired and detailed hardware resource attributes.

The resources are basically hardware resources and software resources. The hardware resource is characterized by:

- the resource type;
- the resource name, in some cases can be used an ID(entifier);
- the resource location;
- a set of its attributes;
- a set of its components; we can define in turn a structured resource.

The software resource is characterized by:

- the resource type;
- the resource name, in some cases can be used an ID(entifier);
- the needed support that is composed by a set of hardware resources every being quoted with appropriate attributes.

The information concerning the authorization, access and security is not detailed here in order to simplify the presentation.

The information that describes the hardware resource, software resource and the services, given as XML files, furnished by the grid will be processed in an appropriate way by the specific tools in order to fulfill the system requirements. In order to find the node where a task can be executed, a matching between the resource node in formation and the resource needed for the software resource (of the task) will be done.

In [7] is given a detailed resource Schema as a XML file where the authors describe a resource that can contain a structured resource one, its components, and every component having its own attributes and being also a structured one. The resource Schema will be used for describing the resource on the Grid or on the Web and also the needed resource for a software resource or for a service offered by the Web or by the Grid. The main feature of the proposed schema is that it can describe a complex resource. Concerning its usage depending on the level of detail that is given the user that will be some specific agent must be able to parse it and match with the given information offered by the requester.

3 The Concepts and Tools used in Modeling

A complex application is composed from a set of tasks that generally must interact and that must be processed (executed) in a specified sequencing order. For its execution every task needs a set of specified resources. The resources are specific and every resource has particular properties. In order to define a complete specification for a resource we will use some conventions from [12] that will be extended.

In many other grid projects are used various means for describing the dynamic behavior of complex grid applications. These means consist from coupled software components and appropriate data. The data is used for:

- mapping the grid application onto the underlying grid middleware;
- controlling the workflow and data flow of the grid application.

Based on the previous, we need tools that allow describing:

- resources of the grid;
- basic resources that are required to define the grid job;
- the model of the grid job workflow using various concepts like Directed Acyclic Graphs (DAG) [18] or Petri nets derivatives [1], [17].

Directed Acyclic Graphs (Figure 1) are widely spread due their structure but they have some disadvantages like: being directed it is impossible to define bidirectional coupling schemes, it is also impossible to define loops. The DAGs describe only the behavior but not the state of the system. The Petri nets can be used to control the workflow of complex applications [1]. Details concerning Petri nets and their properties can be found in [17]. The models for workflow that use Petri nets can be found in papers of van der Aalst [1]. In Figure 2 is given an example of Petri net that is equivalent to the DAG from Figure 1.

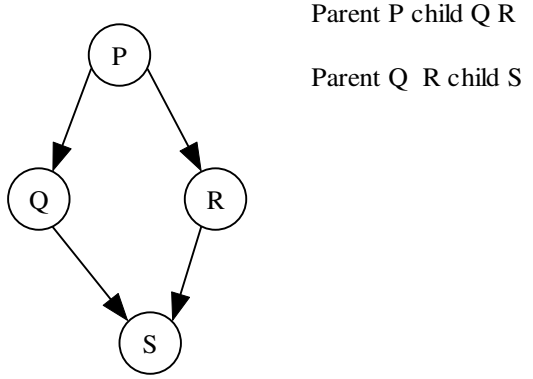


Figure 1
Example of a Directed Acyclic Graph (DAG)

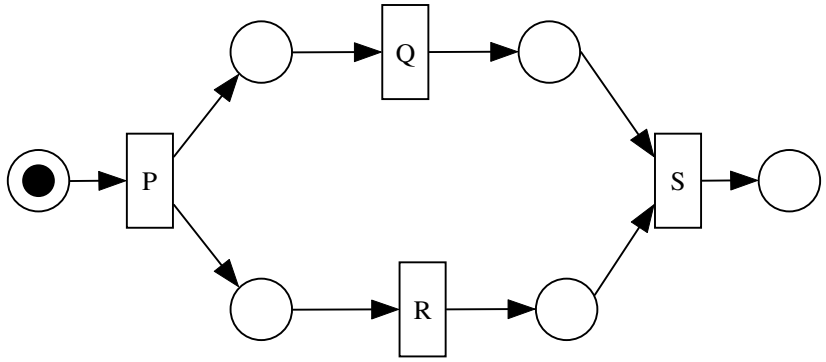


Figure 2
Example of a Petri Net equivalent to the DAG in Figure 1

The Petri nets can be used for modeling the workflow and also to control the workflow of complex applications. In many cases the workflow within grid applications can be equivalent to the data flow. That means that the decision when a software component can be launched in execution depends upon the availability of the input data. As a consequence the tokens in the Petri net represent real data that is exchanged between the software components in the grid. So, the Petri net can be used to model the interaction between software components in the grid represented by software transitions and data resources represented by data places.

In order to control the workflow due to the fact that in some cases the workflow is independent from the data flow is necessary to introduce control transitions and control places. In this case the tokens (that represent the process state) need some additional information that is the color of the token so the colored Petri nets [13]. The Petri nets used for modeling the workflow must have some particular properties that can be seen in [1]. These properties are:

- the net has one input location and one output location;
- the initial marking is that contains only one token in the input location;
- the final marking contains only one token in the output location;
- from any marking (state) between the initial marking and final marking there is a path to the final marking;
- by placing one transition that has an input arc form the output location and an output arc to the input location the Petri net is safe.

A formal verification of the modeled system can be used and it is based of the Petri net properties. This verification can be easy done and expressed in a mathematical form. Basic features that can be expressed like: conflict confusion, contact, trap, deadlock are well-defined properties of Petri nets that can be very useful during of the process of optimization of the workflow of a complex grid

application. We do not enter into the details concerning the features that can be given for express in Petri nets terms of the execution in parallel, sequential execution, definitions of the conditions or loops. There exist many approaches to describe Petri nets with XML-based language, e.g., the Petri Net Markup Language (PNML) [14].

Recently, UML activity diagrams [21] have been used for workflow modeling. There exist some approaches [9] that use the UML's activity diagrams those was added some especially semantic in order to improve a better utility in workflow modeling. The ebXML [8] is used to model e-business services. In the ebXML the event features of activity diagrams are used quite extensively: events being the standard means of communication between different business partners.

Business Process Modeling [23] offers also a set of standards and languages that allow expressing the workflow. Based on WfMC [23] documents was developed Business Process Modeling Notation (BPMN) [6] as a language that allows to describe a process that in our case can be a workflow.

4 The Service Oriented Architecture

The Service Oriented Architecture (SOA) can be understood as the necessary structure that supports communications between services. The service can be defined as a unit of work to be performed on behalf on an entity that can be human user or another program. Service interaction is loosely coupled and self-contained, every interaction being independent one another. The implementation of SOA can use the Simple Object Access Protocol (SOA). The protocol independence of SOA allow to the different consumers to communicate with the service in different ways. A management layer between the providers and consumers ensures a real flexibility regarding implementation protocols.

For an efficient usage of SOA, the Service Oriented Integration (SOI) and Service Oriented Management (SOM) were developed. SOI is used instead of the traditional Enterprise Application Integration (EAI). SOI has as significant characteristics:

- well defined standardized interfaces-consumers are provided with well understood and consistent access to the underlying service;
- opaqueness- the technology and location of application providing the functionality is hidden behind the service interface;
- flexibility- the service is constant both the provider of the service and the consumer of the service can change.

SOM constitutes the operational management of service delivery within a SOA. One of the major features of the SOM is to provide a differentiated service delivery capability during operation using specific objectives for driving the system behavior. A SOM solution supervises and controls the delivery of a service from a service provider to a service requester. A SOM solution can also be viewed as supervising and controlling the consumptions of services by a requester from a number of providers. A SOM solution should by able to manage any service from any technology without requiring code changes, special deployment, or special development environments.

Using a SOA solution we obtain a set of advantages: lower costs, flexibility, the web services that are point to point and it can not solve all the problems, a higher security.

5 The Multi-Agent Systems

Due of the complexity of problems around the development of grid applications an adequate and efficient solution can be developed using the agents that act in a multi-agent system. The agents must be designed in order to fulfill their own goals and also the system goals. The agents must cooperate and concur in order to obtain the needed resources. In many approaches is used the negotiation.

Negotiation between intelligent agents is one of the basic issues in Distributed Artificial Intelligence and Multi-Agent Systems [3]. A negotiation process tends to modify the own plan of each agent in order to achieve agreement among a subset of agents in the system. One of the main works in this area Contract Net Protocol (CNP) [19] serves as a basis for a lot of variants that are already frequently used. The CNP was initially developed for decentralized task allocation and is a distributed negotiation model based on the notion of call for bids on markets. The relation between clients (managers, customers, buyers) and suppliers (bidders, contractors, sellers) is created in a call for bids and evaluation of the proposals submitted by the bidders to the managers. This CNP has several limits:

As a multi-agent system is distributed, several managers can concurrently call for bids, so an agent may have to manage several negotiation processes in parallel in order to reduce the length of its negotiation processes. Some of the applications of the CNP force the contractors to sequence their negotiation processes, i.e., to answer with a single bid at a time. Sequencing the processing of contractor answers to the calls emitted by the various managers may make the contractors miss some contracts. When the multi-agent system is equipped with delay failure detectors, it may also force the managers to consider these contractor agents as failures.

CNP-based applications enable the agent to break its commitments when the agent receives an offer for a better task in comparison with those for which the agent is committed. However breaking the commitments is not always a good solution because it makes managers call again for bids to find anew contractors for their tasks.

6 The Proposed Framework Based on Multi-Agent and SOA

SOA is an architectural manner that has as objective to achieve loose coupling among interacting software agents. A service being a unit of work done by a service provider done by achieves desired results for a given consumer, both provider and consumer are roles played by software agents on behalf of their owners. The results of a service are usually the change of state for the consumer, or for the provider or for both. SOA achieve loose coupling among interacting software agents using as architectural constraints:

- a small set of interface to all participant software agents. Only generic semantics are encoded at the interfaces. The interfaces should be universal available for providers and consumers;
- descriptive messages constrained by an extensive schema delivered through the interfaces. An extensible schema allows new versions of services to be introduced without breaking existing services.

Interfacing in SOA and in distributed applications is important and also expensive and error-prone. The interface prescribes the system behavior and this is very difficult to implement correctly across different platforms and languages. In some cases is recommended to reuse generic interfaces, but this implies that the designer expresses application-specific semantics in messages. Due to the fact that the architecture is service oriented the messages sent must comply with some specific rules:

- the messages must be descriptive, rather that instructive, due to the fact that the service provider is responsible for solving the problem;
- the vocabulary used must be clear for all (provider and consumer) and the structure of the messages must be also well defined for an efficient communication;
- the extensibility allows to evolve the system and it is very important;
- the SOA must give to the consumer a mechanism that enables the consumer to discover a service provider under the context of a service sought by the consumer. This mechanism can be flexible and it (if this is

possible) does not have to be a centralized registry (in the Grid the registry is frequently used especially for nodes).

In order to improve its scalability, performance and reliability of the SOA the following constraints are needed:

- stateless service - the message that a consumer sends to a provider must contain all necessary information for the provider to process it. This makes a service more scalable because the provider does not have to store state information between requests. Each request can be treated as generic. In a way is improved visibility because the monitoring software can inspect one single request and figure out its intention. Due to the fact that there are no intermediate states to worry about the recovery from partial failure is relatively easy and the service becomes more reliable.
- stateful service – between the particular situations that must be carefully designed are: the existence of a session between a consumer and a provider due to the reasons of efficiency; providing of a customized service. This kind of services requires both the consumer and the provider to share the same consumer-specific context that is either included in or referenced by messages exchanged between the provider and the consumer. The drawback of this constraint is that it may reduce overall scalability of the service provider because it may need to remember the shared context for each consumer. It also increases the coupling between a service provider and a consumer.
- idempotent request – (it can be treated carefully due to the fact that means) duplicate requests received by a software agent have the same effect as a unique request. This allows providers and consumers to improve the overall service reliability by simply repeating the request if faults are encountered.

Concerning the service it can be considered in SOA as a piece of functionality whose properties are:

- the interface contract to the service is platform independent; the service can be consumed by a client from anywhere, on any OS and in any language;
- the service can be dynamically located and invoked; it hints that a discovery service is available. The directory service enable a look-up mechanism where consumers can go to find a service based on some criteria;
- the service is self contained, that is the service maintains its own state.

In Figure 3 is illustrated the usage of a directory service.

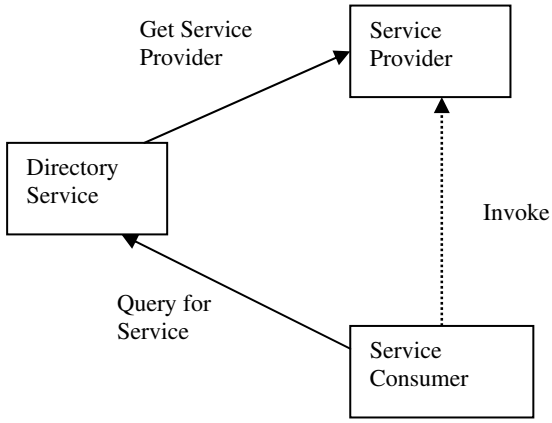


Figure 3
The usage of a directory service

In [12] is proposed a very complex model that uses a Grid Application Definition Language (GADL). It contains a set of XML-based description languages in order to define and assemble complex grid applications for mapping these applications onto the available hardware and software resources and to control the workflow during the execution. The GADL is composed from description languages for resources, interfaces data and jobs. Based on these the user interfaces with by Task mapping, grid job builder and Grid job handler.

The complex application executed on the grid (and not only) imposes a strong analysis concerning:

- its decomposition in tasks with the specification of the resources that are needed during the task execution;
- the tasks sequencing and where the case is the tasks correlation. It is a major challenge the tasks parallelization during their execution;
- the failures and erroneous task execution that imposes: the rescheduling or reminded tasks for execution in the new contest; the temporary data storage imposed by the failure and the supplementary services that are required for replying the rest of the tasks execution.

Concerning the execution, a task in characterized by:

- its identifier;
- the preconditioning of the task execution;
- the post conditioning (that is unleashed by its execution);
- the needed resources and their amount.

All these are given in some forms, usually as XML files.

The sequencing can be done in some ways, one of these being based on Colored Petri Nets. In this paper it is done in a simplified way using Place/Transition Petri Nets. Using such a representation we are able expressing the: concurrency (the parallelism) and synchronization between tasks (and tasks sequences).

In the following figures, we present some of the possibilities that illustrate the modeling of the application execution. In the first one is represented the data flow. Tokens in the places are the data and the transitions are the tasks. The application execution in time is given by tokens flow in the net. Also a tool that manages the Petri net that manages the application execution is needed. The Petri nets used are timed Petri nets. In Figure 4, the data flow and the tokens in the places represent data and the transitions are the tasks execution. Here the sequence of tasks t_2, t_4 and the sequence t_3, t_5 can be executed concurrent (in parallel). The task t_6 is synchronization (instant transition). The initial marking of Petri net is $(1,0,0,0,0,0,0,0,0,0)$ i.e., the start of the application. This marking enables [17] the transition (the task t_1) that fires and based on the Petri net rules the transitions t_2 and t_3 are enabled and also these transitions can fire.

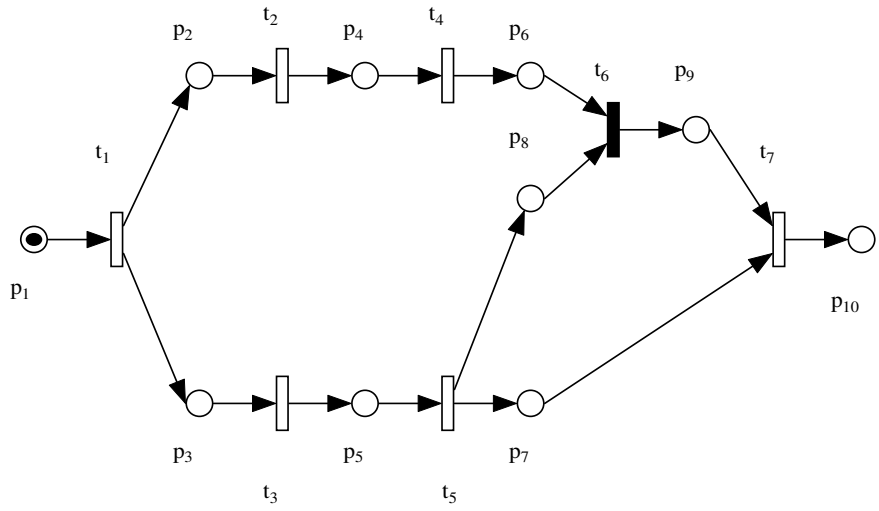


Figure 4
The Petri Net for an application

A major lack of workflow models is that the case instantiation is treated in isolation. In real life that is not true. The major problem concerns to the resources their contention and sharing (these are not treated by workflow models). Based on an analysis done we propose that the following features to be used:

- the time must be taken into consideration, i.e., all the activities evolve in time (the workflow also);
- as an intuitive representation we can imagine that every workflow of an application is represented in a plane and the application are represented in parallel planes;
- the resources that are available in the grid and are used in the application execution are situated between these planes. As an explanation we propose the following. Somewhere a resource is used by a task of an application. This resource is unavailable to other tasks because the token that represents the resource in its appropriate place does not exist, being used by the task. When the task execution is finished then the token that represents the resource is placed back into its appropriate place and can be used by another task.

The other one possibility, where the resources are represented, can be done in two ways. The first one, given in Figure 5 the resource that is in the place p is in the total amount #r and the task t_{1i} needs an amount a_{1i} for its execution that is acquired when it is available in the location p and when the transition t_{1i} fires (the appropriate task is executed) after that this amount is returned back to the location p. The same is the execution of the task represented by the transition t_{2j} from the Application 2.

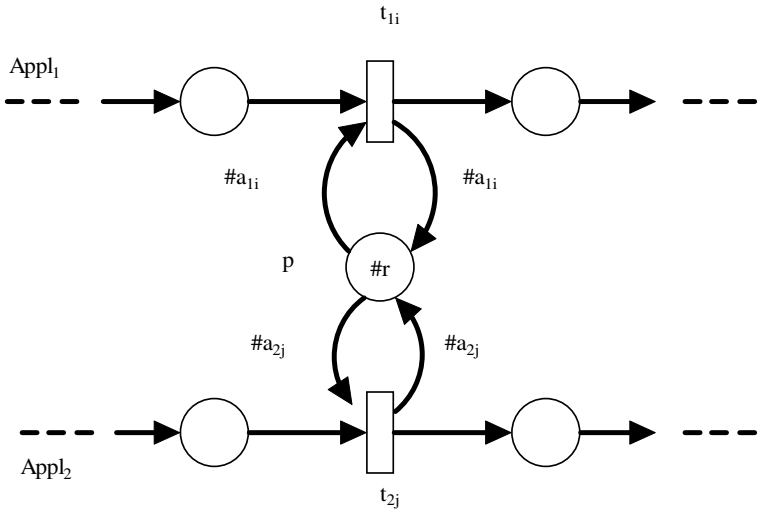


Figure 5
The first way of the representation of task execution

Another way is given in Figure 6, where the task execution is represented in a more complicated manner. Here the transition t_s represents the start of the task execution and the transition t_e represents its end. The time spent by the token into the place between the two transitions t_s and t_e. So, the transitions t_s start the task execution when the resource is available in the desired amount (as in the previous figure) and at the end of the execution, the amount used during the task execution is released and returned to the appropriate place of the resource (denoted by #r).

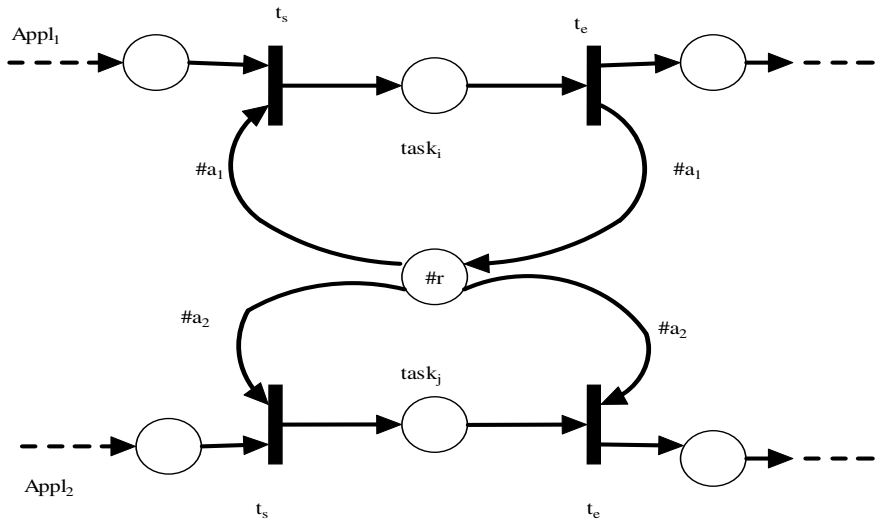


Figure 6
The second way of the representation of task execution

The proposed framework is basically a multi-agent system (MAS), where the agents execute appropriate actions in order to improve their goals and also the general objective: the execution of the application (job) on the grid. In the following, for specific actions are proposed agents that can play different roles. These requirements will be allocated to the roles of the agents. The agents that will be interacting in the system are: Descriptive Agent (DA), Explorer Agent (EA), Broker Agent (BA) and Supervisor Agent (SA).

The complexity domain constitutes a challenge in design due to the multiple factors that influence it. In the following we will consider that an application can be decomposed in tasks, a task being an atomic piece of the application that uses resources during its execution. First there are a set mo major problems concerning the requirement of complex applications like: a set of tools that must be used during the design and also during the execution. These tools are influenced by platforms that support these tools and also by the standards used.

Second, during the execution of a task it can fail. The failure imposes an analysis that must evaluate the effect on the whole execution of the application and the impact of the failure. based on this analysis there is need to take a decision concerning the fact that the execution of the application can be resumed or the execution of the application must be restarted (and in this case when).

Based on the application and failed task there are some aspects that will influence the reply of the execution:

- the conditions for the reply of failed task;
- data that must be partially stored as a consequence of the execution until the failed task and used in the reply of failed task;
- the node or location of the reply of the execution and in this case the time instant in that the reply will be started.

In these conditions we dispose for a set of agents that collaborate one to other and also that will help the user in order that in the proposed framework allows to the users to:

- describe and decompose the application in the tasks, their sequencing during the execution;
- describe the services that are needed for the execution of the application (including the resources needs);
- dispose for the capabilities that allow to reply the execution after a failure of a given task;
- exploring the Internet and/or the Grid in order to discovery the needed resources and services that are necessary for the execution of the application;
- to starting the execution of the application;
- dispose for an efficient tool for overseeing task execution in order to allow to analyses and reply in the case of the failure of a task;
- optimize in some sense the usage of the resources concerning the owners (of the resources) and also optimize the execution after the user criteria.

The Descriptive Agent (DA) allows to the user describes the application. As the result of the description of the application are:

- the tasks and their sequencing for the execution of the application;
- for every task the needed resources and where is the case their amount;
- the reply conditions in the case of the failure of some (every) task.

This description is given as XML files.

The Explorer Agent explores the Internet or the Grid in order to discovery the needed resources and services. It explores the grid and where exist registers it uses their content.

The result of its discoveries is expressed also as XML files.

The Supervisor Agent (SA) as main functions:

a) to receive the requirement for an execution of an application in the form of appropriate XML files that describes the application as we stated above;

b) send to a Broker Agent the requirements concerning the execution of the application;

c) oversees the application execution and in the case of a failure analyses and in the case of reply modifies (updates) the reminded execution adding the needed and supplementary activities (services);

d) interact with the user during the execution (especially in the case of failure that requires the user decision).

The Broker Agent (BA) using the application description and the results of the discoveries made by the EA parses these and negotiate with the Grid/Internet owners in order to allow that the execution can be done and when it can be scheduled at the location (for execution).

During the execution of the application based on the evolution of the execution, SA decides (if it is the case the user is consulted) when a failure occurs and updates the reminded files of the application, contacting a BA for the replies. In the case of a Failure the BA must send some notifications to the Grid/Internet owners concerning the fact that the rest of execution will be canceled and in accordance with the future evolution of the execution the new requirements.

In Figure 7 is given a generic schema of the framework. The user requires an application description (flow 1) that is done by DA. Using it the user requires (flow 2) to the SA to initiate a request for an execution. Based on it the BA requires to EA exploring the grid or the web and furnishes the appropriate information. BA negotiates (flow 3) with the owners of the grid or web the conditions for the execution informing the SA. After the confirming done by the SA (with the user accept eventually), the execution is initiated (flow 4) or the negotiation is repeated until the SA requirements will be satisfied. In the case of a failure for a task (flow 5), the new conditions are stated by the SA and the BA and EA evaluate these conditions and the flows 3, 4 (and eventually 5) are replayed. By numbers are presented main flows:

1-Requirement for the description of the application;

2-Requirement for the execution;

3-Negotiation;

4-Execution;

5-Failure.

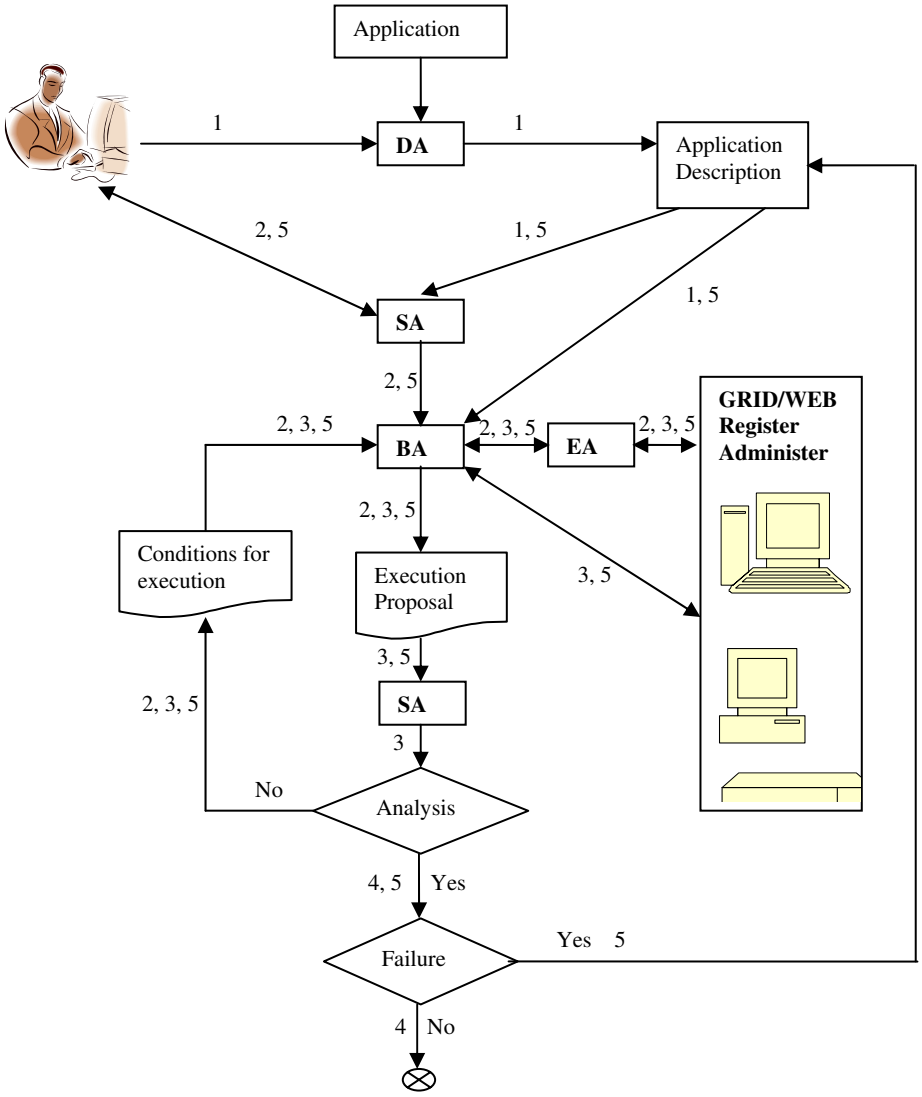


Figure 7
Generic framework schema

7 Applications

The first application is designed in Java. It is based on a client server protocol the client send a request and the server returns a XML file with requested information. If we intend to make a remote software installation the client will be able to interact with the obtained process as the result of the execution of the program that was transmitted for being installed. The client will interact with the standard input and standard output for possible errors of the process. The client will be able to examine only one sub-interval of IPs (or all the IP that are in the same sub network with him). The clients do not depend on the server and they have a Graphical Interface.

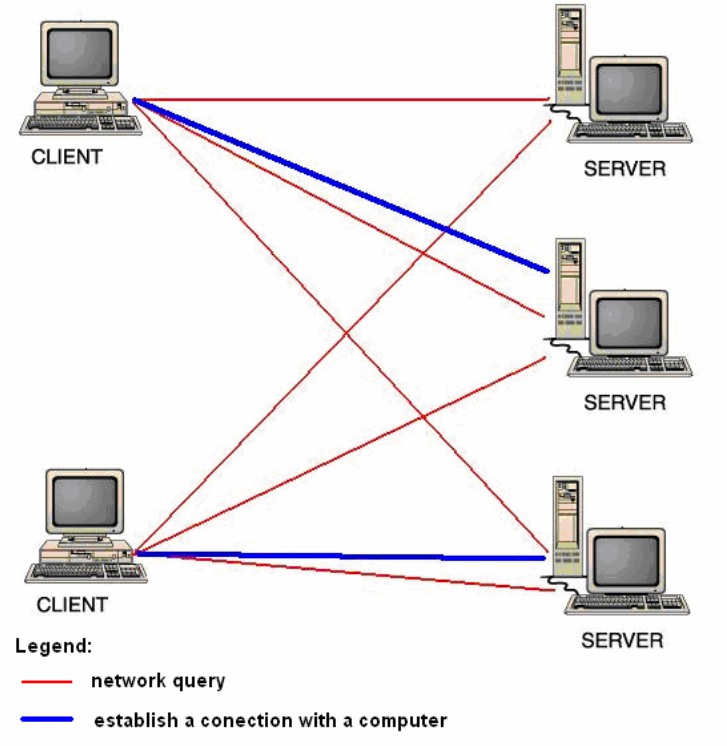


Figure 8
The possible requests of a client

Figure 8 illustrates the possible actions in the first applications. The client can made as requests connections with a computer and send queries.

The second application is designed in Microsoft platform (Visual C++) and allow to finding solutions for mapping of software resources over the available hardware resources. The user defines and can update for every hardware resource its characteristics and also for every software its requirements.

Having the hardware resources and the software with its requirements concerning the hardware that is necessary the user can verify if in the available hardware there the required resources are concerning the hardware for specified software. The user can save the project in an XML file. As a small example of such project is the following.

```
<root>

<resource id="software1" type="software">
        <resourceRef type="cpu-mhz" value="1000" />
        <resourceRef type="ram-mb" value="64" />
        <resourceRef type="os" value="linux" />
        <resourceRef type="kernel-version" value="2.6" />
</resource>
<resource id="software2" type="software">
        <resourceRef type="cpu-mhz" value="1200" />
        <resourceRef type="ram-mb" value="128" />
        <resourceRef type="os" value="linux" />
</resource>
<resource id="software3" type="software">
        <resourceRef type="cpu-mhz" value="2200" />
        <resourceRef type="ram-mb" value="512" />
        <resourceRef type="dot-net-framework" value="1" />
</resource>
<resource id="grid-node-01" type="hardware">
        <resourceRef type="cpu-mhz" value="1700" />
        <resourceRef type="ram-mb" value="512" />
        <resourceRef type="os" value="linux" />
        <resourceRef type="kernel-version" value="2.6" />
</resource>
<resource id="grid-node-02" type="hardware">
        <resourceRef type="cpu-mhz" value="2700" />
        <resourceRef type="ram-mb" value="1024" />
        <resourceRef type="os" value="windows" />
        <resourceRef type="dot-net-framework" value="1" />
</resource>

</root>
```

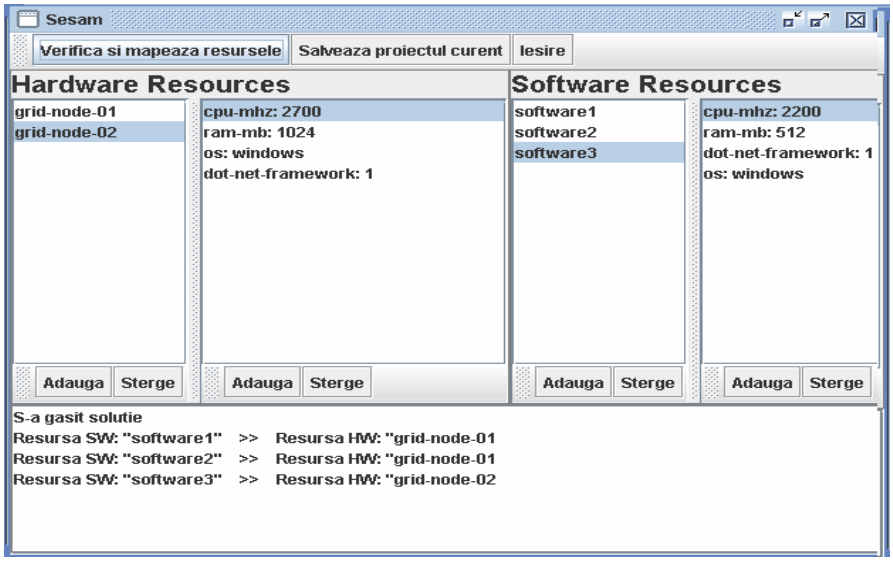


Figure 9

The Graphical interface for controlling the existence of the required software in the available hardware resources

In Figure 9 is given the graphical interface that allows to the user to update and verify that the needed hardware resources for specified software exist in the real available context of hardware.

Conclusions

The grid and its applications is one of the most dynamic and challenging domains. Most of companies and academic organizations are involved in grid and its applications. It is very difficult to develop a complex and complete framework for the execution of applications on the grid due to complexity of problems. The paper intended to propose a framework that is based on a multi-agent system and uses the XML as basis for information that is used in to the framework.

The design in the nearest future will be greatly influenced by the SOA. The services are most used in complex applications and in the application that uses the Grid.

References

[1] van der Aalst, W. M. P.: The Applications of Petri Nets to Workflow Management, The Journal of Circuits, Systems and Computers 8, pp. 21-66, 1998

[2] Activex-Grid: http://www.clientserverhelp.com/activex/activex_grid.html

[3] Akine, s., Pinson, S., Shakun, M.: An Extended Multi-Agent Negotiation Protocol, Autonomous Agents and Multi-Agent Systems, Vol. 8(1), pp. 5-45, 2004

[4] Brown, J.L, Ferner, C.S., Shipman, W.J: GridNexus and JXPL: A Grid Services Workflow System, http://www.globusworld.org/program/, 2006

[5] Business Process Management Initiative, http://www.bpmi.org

[6] Business Process Modeling Notation http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf, 2004

[7] Cicortas, A., Iordan,V.: Multi-Agent Systems for Resource Allocation, SACI 2005 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, Timisoara, Romania, pp. 221-232, 12-14 May 2005

[8] ebXML http://www.ebxml.org, 2003

[9] Ershuis, R., Wieringa R.: Comparing Petri Net and Activity Diagram Variants for Workflow modeling- A Quest for Reactive Petri Nets, In H. Ehrig, W. Reisig, and G. Rozenberg, editors, Petri Net , Technologies for Communication Based Systems, LNCS, Springer-Verlag, Berlin, pp. 2325-354, 2002

[10] Fraunhofer Resource Grid http://www.fhrg.fhg.de, 2003

[11] The Globus Project: The Globus Resource Specification Language (RSL v.1.0), http://www.globus.org.gram/rsl_spec1.html, 2000

[12] Hoheisel, A., Der, U,: AN XML-Based Framework for Loosely Coupled Applications on Grid Environment, P.M.A. Sloot et al. (Eds.),ICCS 2003, LNCS 2657, pp. 245-254, 2003

[13] Jensen, K.: Colored Petri Nets, Vol1-3 Springer, 1997

[14] Jungel, M., Kindler, E., Weber, E.: The etri Net Markup Language, Philippi, S., (ed) Workshop Algorithmen und Werkzeuge fur Petri Netze. Univ. Koblenz-Landau, pp. 47-52, 2000

[15] Maciel, et.all: Resource Management in OGSA, http://www.ggf.org/documents/GFD.45.pdf, pp. 1-24, 2005

[16] Kacsuk, P., Dosza, G., Kovacs, J., Lovas, R., Podhorszki, N., Balaton, Z., Gombas, G: P-GRADE: A Grid Programming Environment, J. Grid Comput. Vol. 1, No. 2, pp. 171-197, 2003

[17] Petri, C.A.: Kommunication mit Automaten, Ph.D. Dissertation, Insitut fur Instrumentale Mathematik, Bonn, 1962

[18] Romberg, M.: The UNICORE Architecture: Seamless Access to Distributed Resources, Proc. of the 8th Int'l Symposium on High

Performance Distributed Computing (HPDC-8), Los Alamitos, pp. 287-293, 1999

[19] Smith, G. R: The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver, IEEE Transactions on Computers, Vol. 12, 1980

[20] Treadwell, J.: Open Grid Services Architecture Glossary of Terms, http://www.ggf.org/documents/GFD.44.pdf, pp. 1-15, 2005

[21] Introduction to OMG's UML, http://www.omg.org, 2005

[22] Walsh, N.: A Technical Introduction to XML, http://www.xml.com/pub/a/98/10/guide0.html, 1998

[23] Workflow Management Coalition, http://www.wfmc.org

Specification-based Retrieval of Software Components through Fuzzy Inference

Ioana Şora, Doru Todinca

Department of Computer Science, "Politehnica" University of Timişoara
Bd. V. Parvan 2, 300223 Timisoara, Romania
E-mail: {ioana.sora, doru.todinca}@ac.upt.ro

Abstract: In the component based software engineering approach, a software system is viewed as an assembly of reusable independently developed components. In order to produce automated tools to support the selection and assembly of components, rigorous specifications of components and performant retrieval and selection strategies based on these specifications are needed. Classical approaches for automatically retrieving components that match a set of required properties result mostly in very complex solutions. In this article we propose an efficient fuzzy logic based solution for the specification and retrieval of software components. We describe the basic principles of the proposed solutions and illustrate them on example scenarious.

Keywords: fuzzy logic, fuzzy inferece, software components

1 Introduction

In the component based software engineering approach [1], [11], a software system is viewed as an assembly of reusable independently developed components. The set of components and the manner in which these are connected with each other determines the properties (functionality and behaviour) of the assembled system.

Constructing a system with certain required properties starts with the compositional decision: which components to select from a large repository of components and which topology to give to their assembly? During this decisional process, the selection is made based on the matching between the required properties and the known properties of the components. In order to use automatic tool support, a systematic compositional model is needed. Such a compositional model must comprise a component specification scheme and formalism, and a coordinated, well defined requirements driven composition strategy. It must have the expresiveness to describe many types of requirements and properties (functional, non-functional, structural, behavioural), it must be sufficiently formal

to be used in automatic tools and still its complexity should permit reasonable implementation.

Many different approaches have been proposed for the specification of components: interface description, formal specification methods, behavioural specification, architectural description. In previous work, we also developed a compositional model at the architectural level, the composable components model and its specification formalism, CCDL [8], [10]. A critical issue are the non-functional properties of components, as speed, memory consumption, usability, etc. These are difficult to specify and match [3], most often the matching of a set of such requirements is based on a series of trade-offs. This work proposes a solution by applying fuzzy logic in this field of software engineering.

The remainder of this article is organized as follows: Section 2 resumes the basic concepts of our component model, CCDL. Section 3 presents the extension of CCDL with fuzzy attributes and fuzzy logic based selection of components. All concepts are illustrated on a running example, built incrementally over all sections. Section 4 discusses our work in the context of other works. The last section summarizes the concluding remarks.

2 Core CCDL Component Model

This section resumes the basic concepts of our component model. Only these aspects of the complex composable components model (CCDL) that are dirrectly affected by the extension proposed in this work are presented here, many details beeing omitted. The CCDL model has been extensively presented in [8] and [10].

2.1 Principles Used in the Specification of Components

A software system is viewed as a set of components that are connected to each other through connectors. As defined in mainstream component bibliography [11], [1] a software component is an implementation of some functionality, available under the condition of a certain contract, independently deployable and subject to composition.

In our approach, each component has a set of ports as logical points of interaction with its environment. We distinguish between input ports and output ports and consider that syntactically every input port is plug-compatible with every output port. The logic of a component composition (the semantic part) is enforced through the checking of component contracts.

Components may be simple or composed. A simple component is the basic unit of composition that is responsible for certain behaviour and has one input port and

one output port. Composed components introduce a grouping mechanism to create higher abstractions and may have several input and output ports.

Components are specified by means of their provided and required properties. Properties in our approach are facts known about the component, in a way similar to Shaw's credentials [7]. A property is a name from a domain vocabulary set and may have refining subproperties (which are also properties) or refining attributes that are typed values. For example, a property that does data compression will be described through a property named compression. An attribute of this property can be defined as the average compression rate, expressed as the real type attribute compression-factor.

The component contracts specify the services provided by the component and their characteristics on one side and the obligations of client and environment components on the other side. Most often the provided services and their quality depends on the services offered by other parties, being subject to a contract. In the CCDL model contracts are expressed as provides and requires clauses containing sets of provided and required properties. The component as a whole provides certain services, defined by global provides clauses in the component specification. In order to provide these services it requires that other services are provided by the environment. These services must be provided in certain defined interaction points, thus the requires clauses are attached to the ports.

2.2 Automatic Requirements-driven Selection and Composition

A component assembly is valid if it provides all user required services and if the contracts of all individual components are respected. A contract for a component is respected if all its required properties have found a match. The criterion for a semantically correct component assembly is matching all required properties with provided properties on every flow in the system.

A match between a required and a provided property is established first by matching the properties names, then recursively matching subproperties and matching of the attributes values. A property present in the requirement must not specify all attributes of the property present in the provides clause for a match.

In our approach, it is not necessary that a requirement of a component is matched by a component dirrectly connected to it. It is sufficient that requirements are matched by some components that are present on the flow connected to that port, these requirements are able to propagate.

The internal structure of a composable target can be established at runtime through automatic requirements-driven composition. The requirements for the composable target result from its invariant structural constraints and from the current requirements imposed by the external environment. The overall process of

generating the structure of the target is driven by the requirements. The required properties for the target are put on the main flow of the target and propagated from that point on, while adding components. The addition of new components on the flow occurs according to the current requirements, which are those propagated from the initial requirements together with those of the new introduced components. A component is added to the solution if it matches at least a subset of the current requirements. A solution is considered complete when the current requirements set becomes empty. It is possible that for a certain set of requirements no solution can be found or that several component assemblies are found.

The mechanism of propagation of requirements briefly sumarised here was formally described in [9], paper that also gives a complete description of the automatic composition strategy.

2.3 Issues

The compositional model presented above can be improved in regard with the specification and matching of properties and attributes.

In the core CCDL model, a property most often consists of a name describing functionality and attributes that are typed values describing the non-functional characteristics of the functional property.

For example, a component providing data compression can be characterized by a property named compression. The non-functional characteristics are described by attributes like compression_rate, speed, memory_consumption. For the specification of the component, these attributes must be first evaluated. The compression_rate can be expressed as a real type value, representing the average measured compression rate for different data inputs. Not all attributes can be as easily evaluated and described as crisp values. The exact value of the speed attribute depends essentially on the hardware configuration of the system. Such an attribute should be more adequately be described using terms as ‘fast’, ‘very fast’, ‘slow’, description established as a result of comparing the performance of the current component relative to the performance of other components that provide the same functional property.

In the core CCDL model, a match between a required and a provided property is established first by matching the properties names, then recursively matching subproperties and matching of the attributes values. Let consider components C1 and C2. C1 provides property P with the attributes A1=Value1, A2=Value2. C2 provides the same property P with the attributes A1=Value3, A2=Value4. A client requirement can be for property P with attributes A1=ValueX, A2=ValueY. Then either C1 or C2 will be selected, whether ValueX=Value1 and ValueY=Value2 is true or ValueX=Value3 and ValueY=Value4. If the values do not match exactly, no component will be selected.

In many cases, a client requirement has no precise requirements regarding the values of attributes. For example, a requirement regarding the selection of a compression component will rarely need to specify an exact value for the compression_rate as to require, i.e., compression_rate=0.4. It will rather specify that compression_rate=medium is acceptable.

In order to relax the specification of requirements, in the strict matching mechanism of the core CCDL it is still possible that a property present in the requirement does not have to specify all attributes of the property present in the provides clause for a match. In the previous example of components C1 and C2, a client requirement can omit the specification of required values for all or some of the attributes of property P. For example, a requirement as property P with attribute A2=ValueX will select all components that provide property P with attribute A2 matching the ValueX, regardless of the values of the attribute A1. This possibility of uncomplete requirement specification is not enough to give the desired flexibility. It misses the possibility to specify required ranges, as well as the possibility of doing tradeoffs between the matching degree of several attributes of the same property.

From the examples presented above, we conclude that there are two main issues with the specification and matching of properties:

- certain attributes cannot be exactly evaluated and specified
- the matching of required/provided attributes must not always be precise

We believe that fuzzy logic [13] can help to overcome these problems because it aims at exploiting the tolerance for imprecision and uncertainity. The next chapter proposes our fuzzy based solution.

3 Fuzzy Attributes and Selection

In this article, we propose an extension of the core CCDL model, where attributes can be expressed in fuzzy logic and the properties matching is done by fuzzy inference.

3.1 Fuzzy Attributes

3.1.1 Concepts

A property consists of a name describing functionality and attributes that are either typed values (crisp) or fuzzy terms.

For example, the property compression can be defined with attributes that are both crisp and fuzzy. The atributes compression_rate and memory_consumption are crisp values, the attribute speed is a fuzzy value as it depends on the hardware configuration and performance of the system.

The names used for the properties and for the attributes are established through a domain-specific vocabulary. Such a restriction is necessary because a totally free-text specification makes the retrieval difficult, producing false-positive or false-negative matchings due to the use of a non-standard terminology. Establishing domain specific vocabularies is a common solution [6], [5] and has been used also in the core CCDL model [8].

In our work, the domain specific vocabulary must also describe the domains of the fuzzy attributes (linguistic variables) for each property as well as the membership functions for the fuzzy terms.

The membership functions for all linguistic variable are considered of trapezoidal/triangular shape as in Figure 1:

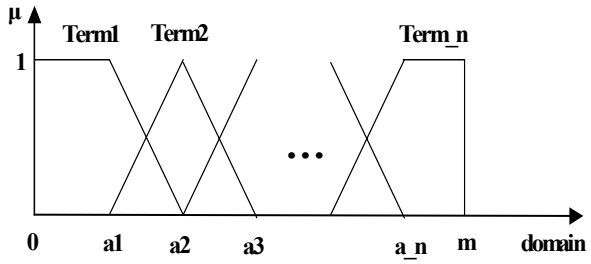


Figure 1
The shape of the membership functions

For each linguistic variable, first the number and the names of the terms of its domain must be declared, and after that the values of the parameters a1, a2, …, a_n must be specified.

3.1.2 Example

A component repository contains several implementations of components that have the functionality of data compression, specified with the provided property compression. They are diferentiated through the values of their non-functional attributes. Let us consider two different components, C1 and C2, specified as follows:

Component C1:

Property compression with attributes

compression_rate =crisp(0.6)

memory_consumption=crisp(512)

speed=fuzzy(slow)

Component C2:

Property compression with attributes

compression_rate =crisp(0.8)

memory_consumption=crisp(1024)

speed=fuzzy(medium)

Each of the three attributes is defined as a linguistic variable with three terms, as follows:

domain(compression_rate) = {low, medium, high}

domain(memory_consumption) = {low, medium, high}

domain(speed) = {slow, medium, fast}

For each linguistic variable, the parameters a1, a2, a3 defining the shape of the membership functions are defined. In our example, in case of the attribute compression_rate, these values are defined as a1=0.3, a2=0.6, a3=0.75.

It is important to note that a linguistic variable that characterizes an attribute can have different meanings in the context of different properties. The domain and the shape of a linguistic variable can be redefined in the context of different properties. For example, the attribute memory_consumption can be attached to very different functional properties: a compression component, a sorting component, a signal processing component, etc. In each of these cases, there are different expectations about the amount of memory needed: an amount of memory that is considered normal (medium) for signal processing is considered very high if it is required by a compression component. Thus the values a1, a2, a3 and maybe also the number of the linguistic terms will be defined different in the context of every property.

3.2 Generation of Fuzzy Rules from Requirements

3.2.1 Principle of Fuzzy Rule Generation

A new property matching mechanism is defined.

In general, a requirement as:

Req property P with attributes A1=V1 and A2=V2 and … An=Vn

is handled in the following manner:

First, the basic functionality is ensured, matching properties names according to the classical composition strategy. Usually several solutions result from thist first step.

Second, the preliminary solutions are selected and hierarchized according to the degree of attribute matching. This is done by fuzzy logic. The given requirement is translated into the corresponding rule:

If A1=V1 and A2=V2 and … An=Vn then decision=select

The generation of the fuzzy rules is done automatically starting from the requirements.

Very often, the required attributes are not values, but rather are required to be at least (or at most) a given value, A>=V or A<=V. For example, the speed is usually not required to be medium, but at least medium.

In general, a requirement containing the attribute expression A>=V will be translated into a set of i rules, for all Vi>=V:

If A=Vi then decision=select

3.2.2 Automatic Extension of the Fuzzy Rules Set

Several rules are generated from one requirement. In order to relax the selection, it is considered a match even if one of the linguistic variables in the premises matches only a neighbor of the requested value (the predecessor or the succesor). In this case the decision of selection is a weak one. In the case that more than one linguistic variable in the premise matches only neighbor values (while the rest match the requested fuzzy terms), the decision is a weak reject. In the extreme case that all linguistic variables in the premises match neighbor values, the decision is a weak reject. In all the other cases, the decision is a strong reject.

For example, in the case of a requirement containing two attributes, A1=V1 and A2=V2, the complete set of generated rules is:

The dirrectly generated rule is:

If A1=V1 and A2=V2 then decision=strong_select

The rules generated if one of the linguistic variables in the premises matches only a neighbor of the requested value are (maximum 4 rules):

If A1=pred(V1) and A2=V2 then decision=weak_select

If A1=succ(V1) and A2=V2 then decision=weak_select

If A1=V1 and A2=pred(V2) then decision=weak_select

If A1=V1 and A2=succ(V2) then decision=weak_select

In this case there are a maximum number of four generated rules, if neither V1 nor V2 are extreme values of their domains. If a value is the first value in the domain it has no predecessor, if it is the last value in the domain it has no successor.

The rules generated if more then one of the linguistic variables in the premises matches only a neighbor of the requested value are (maximum 4 rules):

If A1=pred(V1) and A2=pred(V2) then decision=weak_rej

If A1=succ(V1) and A2=pred(V2) then decision=weak_rej

If A1=pred(V1) and A2=succ(V2) then decision=weak_rej

If A1=succ(V1) and A2=succ(V2) then decision=weak_rej

For all the rest of possible combinations of values of A1 and A2 the decision is strong reject. The number of rules from this category depends on the number of terms of the linguistic variables A1 and A2.

3.2.3 Specifying the Relative Importance of Attributes

In order to allow the user to specify which attributes are more important and to treat them accordingly, different weights can be declared in the requirement specification. These weights describe how strict is a certain requirement: very strict, strict, normal, or less.

A requirement will be expressed for example in the following manner:

Req property P with attributes

A1=V1/imp=strict and A2=V2/imp=normal and … An=Vn/imp=less

The process of automatic generation of the extended set of fuzzy rules, presented in the previous paragraph considering the case when all attributes have normal importance, is adapted to deal with the attributes of different importance.

The attributes of strict importance will never be replaced by neighbor values in the generated fuzzy rules. The attributes of less importance will be replaced by neighbor and also second neighbor values in the generated rules.

3.2.4 Example

An application requires property compression with attributes having the values compression_rate=*medium* and speed=*medium*:

Req property compression with attributes

compression_rate=medium and speed=medium

No relative importances are specified for these two attributes, thus they are considered of equal importance.

The first rule which is generated is the direct rule:

[R1] If compression_rate=medium and speed=medium then

decision=strong_select

As illustrated in paragraph 3.1.2, the domains for compression_rate and speed contain each three linguistic terms: *low*, *medium*, *high* for compression_rate and *slow*, *medium*, *fast* for speed. For compression_rate, pred(*medium*)=*low* and succ(*medium*)=*high* while for speed pred(*medium*)=*slow* and succ(*medium*)=*fast*.

The rules generated for one different neighbor are:

[R2] If compression_rate=low and speed=medium then

decision=weak_select

[R3] If compression_rate=high and speed=medium then

decision=weak_select

[R4] If compression_rate=medium and speed=slow then

decision=weak_select

[R5] If compression_rate=medium and speed=fast then

decision=weak_select

The rules generated for two different neighbors are:

[R6] If compression_rate=low and speed=slow then

decision=weak_reject

[R7] If compression_rate=high and speed=slow then

decision=weak_reject

[R8] If compression_rate=low and speed=fast then

decision=weak_reject

[R9] If compression_rate=high and speed=fast then

decision=weak_reject

In this case, there are no rules where the decision is strong_reject, because both of the linguistic variables have only two terms and the requested term was the middle term.

We must also remark that the requirement was expressed using only the equality operator =. Thus, bigger values than the requested one are not considered to be best matches. If the user wants to consider better values than the requested one as best matches (for example, value *fast* for speed to be not considered worse that value *medium*) he must explicitly specify this in the requirement as speed >= medium. In this case, the rules that have speed=fast in the premise will have as conclusion a stronger selection decision.

3.3 Using Fuzzy Inference for Component Selection

3.3.1 The Principles

The selection of components corresponding to a set of requirements is done in several steps: First, the matching of properties according to the classic composition strategy resumed in paragraph 2.2 results in a list of potential solutions. Second, all the required attributes are used to generate the set of fuzzy rules as described in paragraph 3.2. Finally, the potential solutions obtained in the first step are hierarchized with help of fuzzy inference over the set of rules.

Given a fact A' and a rule $R_{A\rightarrow B}$, fuzzy inference means the composition A'$\circ$ $R_{A\rightarrow B}$ in order to obtain the conclusion B'= A'$\circ$ $R_{A\rightarrow B}$.

In our case, a fact A' is an expressions containing attributes of one candidate solution combined using logical operators, the premises A of the fuzzy rules are composed of attributes of the requirement, while the conclusion is the linguistic variable decision.

The premises of the rules are composed of several attributes, the degrees of activation of each premise are combined using the corresponding logical operators (AND, OR). The AND operator is implemented by *minimum* in fuzzy logic and the OR operator is implemented by *maximum* in fuzzy logic.

When more than one rule is active, the consequents of all active rules are combined through the union operator, which is implemented as a maximum between the membership functions of the partial conclusions. Most often, the result of the fuzzy inference has to be a crisp value, obtained by an averaging procedure applied on the partial conclusions, process that is called defuzzification. A widely used defuzzification method is the center of gravity.

The results obtained through defuzzification of the conclusions obtained for each candidate solution lead to their hierarchisation.

3.3.2 Example

Let us consider the following scenario to illustrate how the inference process works on selection of components. An application requires property compression with attributes values compression_rate=medium and speed=medium. From this requirement, the set of fuzzy rules illustrated in paragraph 3.2.4 are automatically generated. The component repository contains different component implementations for property compression, having different values for the nonfunctional attributes. Let suppose that there are available N different component implementations, two of them being these specified in the example of paragraph 3.1.2. For each of the N candidate components the value of the selection decision is calculated.

Figures 2-5 illustrate how each of the generated rules is composed with the fact represented by the specification of component C1 (with comp_rate=0.6 and speed=slow). Only four of the generated rules are activated, all other rules don't influence the inference process because their terms are not intersected by the facts resulting from the specification of component C1.

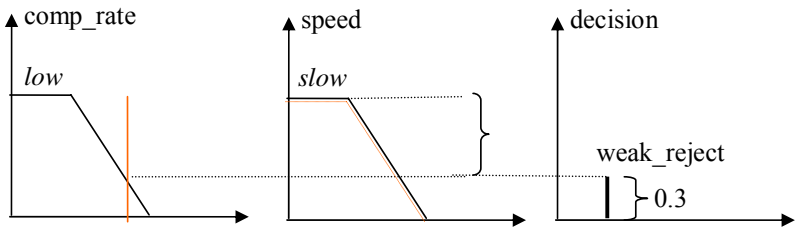


Figure 2

Rule: if compression_rate=*low* and speed=s*low* then decision=*weak_reject*.

Facts: comp_rate=0.6, speed=slow

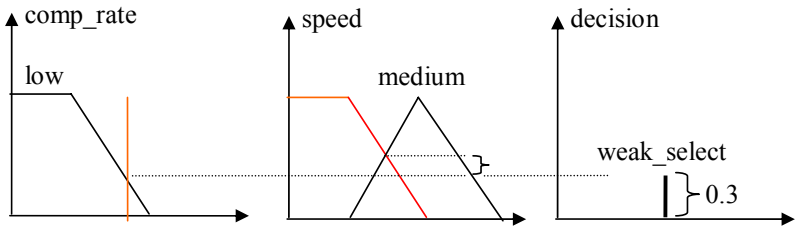


Figure 3

Rule: if compression_rate=*low* and speed=*medium* then decision=*weak_select*.

Facts: comp_rate=0.6, speed=slow

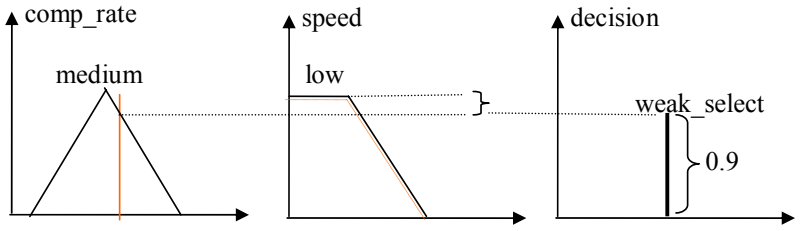


Figure 4

Rule: if compression_rate=*medium* and speed=s*low* then decision=weak_select.

Facts: comp_rate=0.6, speed=slow

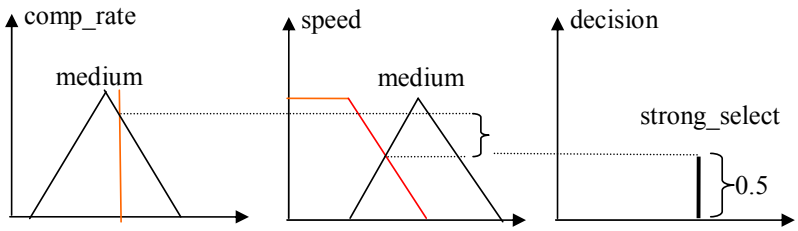


Figure 5

Rule: if compression_rate=*medium* and speed=*medium* then decision=*strong_select*.

Facts: comp_rate=0.6, speed=slow

The decision is the union between the partial conclusions weak_reject=0.3, weak_select=0.9, weak_select=0.3 and strong_select=0.5. Applying as a defuzzification method the the center of gravity for these partial conclusions, it results a decision value that classifies the component C1 as acceptable.

Similarily, fuzzy inference is also done for all the other candidate components. They can be ranked according to their decision values, the component with the decision value placed nearer to the right end of the domain (nearer to strong_select) being the best choice.

4 Related Work

Many different approaches have been proposed for the specification of components: interface description, formal specification methods, behavioural specification, architectural description. In previous work, we also developed a compositional model at the architectural level, the composable components model and its specification formalism, CCDL [8], [10]. A critical issue are the non-functional properties of components, as speed, memory consumption, usability, etc.

Since the use of fuzzy logic for the specification and retrieval of software components is a new approach, only a few experimental results are described in the literature. Two notable approaches are these of Zhang et al [12] and that of Cooper et al [2].

The work of Cooper et al [2] focuses on gathering specification data for individual components by tests and on the process of fuzzification of these specifications (their representation by membership functions). In their article they do not emphasis on the process of quering and decision making process for retrieving components that correspond to certain required properties, which instead is the main focus of our work. Another particularity of our work is that the individual

components do not have to bespecified with fuzzy properties. The advantage of this is that the fuzzyfication information (the shape of the membership functions) is described independent of individual components as a domain specific information.

Zhang et al [12] present a component matching system targeting automatic component searching and matching across the internet. The system is based on XML specifications and supports user queries that are specifications with incomplete/uncertain attributes. The matching approach used in their work is not fuzzy inference as it is used in our work. In [12], a query is parsed into a document object model (DOM) and the DOM is transformed to an internal tree-structured model. After this, fuzzy logic scoring and aggregation algorithms are applied to the internal tree structure to provide a ranked set of candidate approximate matches. In our approach, the matching is based directly on fuzzy logic inference which can be implemented in a generic manner.

Conclusions

In this article we introduced a new approach for the specification and retrieval of software components. This solution is based on fuzzy logic and extends our previous work on architectural level specification. The advantages of this approach are: a natural treatment for certain non-functional attributes that cannot be exactly evaluated and specified, and a relaxed matching of required/provided attributes that don't have to always be precise.

References

[1] Felix Bachman, Len Bass, C Buhman, S Comella-Dorda, F Long, J Robert, R Seacord, Kurt Wallnau: Technical concepts of component-based software engineering, Technical Report CMU/SEI-2000-TR-008, Carneggie Mellon Software Engineering Institute, 2000

[2] Kendra Cooper, Joao Cangusu, Rong Lin, Ganesan Sankaranarayanan, Ragouramane Soundararadjane, Eric Wong: An Empirical Study on the Specification and Selection of Components Using Fuzzy Logic, in Proceedings of 8th International Symposium on CBSE, St. Louis, USA, May 2005

[3] Xavier Franch: Systematic formulation of non-functional characteristics of software, in Proceedings of the 3rd IEEE International Conference on Requirements Engineering, Colorado Springs, USA, April 1998, pp. 174-181

[4] Murat Koyuncu, Adnan Yazici: A Fuzzy Knowledge-Based System for Intelligent Retrieval, in IEEE Transactions on Fuzzy Systems, Vol. 13, No. 3, June 2005, pp. 317-330

[5] Hadef Mili, Estelle Ah-Ki, Robert Godin, Hamid Mcheick: An experiment in software component retrieval, in Information and Software Technology, Elsevier

[6] The Object Management Group: Catalog of Domain Specifications. http://www.omg.org/technology/documents/domain_spec_catalog.htm

[7] Mary Shaw: Truth vs knowledge – the difference between what a component does and what we know it does, in Proceedings of the 8th International Workshop on Software Specification and Design, pp. 181-185

[8] Ioana Sora, Pierre Verbaeten, Yolande Berbers: A Description Language for Composable Components, in Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science LNCS No. 2621, Springer, 2003, pp. 22-37

[9] Ioana Sora, Vladimir Cretu, Pierre Verbaeten, Yolande Berbers: Automating decisions in component composition based on propagation of requirements, in Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science LNCS No. 2984, Springer, 2004, pp. 374-388

[10] Ioana Sora, Vladimir Cretu, Pierre Verbaeten, Yolande Berbers: Managing Variability of Self-customizable Systems through Composable Components, in Software Process Improvement and Practice, Vol. 10, No. 1, Addison Wesley, January 2005

[11] Clemens Szyperski: Component Software: Beyond Object Oriented Programming, Addison Wesley, 2002

[12] Ting Zhang, Luca Benini, Giovanni De Micheli: Component Selection and Matching for IP-Based Design, in Proceedings of Conference on Design, Automation and Test in Europe (DATE), Munich, Germany, 2001, pp. 40-46

[13] H.-J. Zimmermann, Fuzzy sets theory – and its applications. Second, revised edition, Kluwer Academic Publishers, 1991

Above Flux Estimation Issues in Induction Generators with Application at Energy Conversion Systems

Iosif Szeidert, Octavian Prostean, Ioan Filip, Vasar Cristian

Department of Automation and Applied Informatics
Faculty of Automation and Computer Sciences
"Politehnica" University from Timisoara
Av. V. Parvan, No.2, 300223, Timisoara, Romania
Phone: (0040) 256 403237, Fax: (0040) 256 403214, siosif@aut.utt.ro

Abstract: The paper presents issues regarding the flux estimation of induction machines. The electrical machines variable's estimation represents a major problem in the actual context of modern control approaches, especially of sensorless control strategies. There are considered several implementations of induction machine's flux estimators by using the Matlab-Simulink environment and there are drawn the afferent conclusions.

Keywords: flux estimator, observer, induction machines, simulation, Matlab-Simulink environment

1 Introduction

The issue of the state estimation represents a major problem in the actual trend of modern electrical machines control. The state estimation must be solved especially in the case when those variables are not measurable or the transducers are expensive (such as torque, or flux transducers) and the price cost of a control system increases due to this fact. Therefore, appears the necessity of using of control schemes and methods without the usage of some specific transducers/sensors. This case is the one of the sensorless control methods. [1]

The estimator's implementations are based on the usage of control plant models, their aim being to estimate the value of non-measurable variables by using other measurable variables.

Basically there are two major types of estimators: [1]

- Estimators without correction (without feedback).

- Asymptotic estimators or observers (with feedback), that presents a predictive correction in order to assure a faster convergence and a better robustness of estimation at the estimated plant's parameter variations and at exogenous perturbations.

In the context of electrical machines the most commonly used estimators are:

- Flux and torque estimators – are used in the FOC (field oriented control) and the direct torque and flux vector control.
- Speed and acceleration estimators based on measured position – are used in position/speed controllers, state controllers, sliding mode controllers, etc.
- Position and speed estimators based on measured stator variables (currents and voltages) – used in the sensorless control (without position sensors/transducers).
- Perturbation estimators used in equivalent perturbation compensators.

In order to fulfill the requirements of a faster and accurate control (reduced response periods) of the induction machine, the value of the flux variable must be known. This variable's value can be estimated on the basis of voltage, current and rotation speed measurements. There are several control strategies and methods for the induction machine. In technical literature, are especially used the control based on stator flux and the one based on the rotor flux. [2] [3].

The paper describes some estimator structures implemented in the Matlab-Simulink simulation environment. The paper studies those estimator structures having in view the possibility of their implementation in unconventional energetic conversion systems, especially of WECS (wind energy conversion systems). There are studied three induction machine's flux estimators: based on voltage model, based on current model and based on the stator current estimation and a flux observer.

2 Induction Machine's Flux Estimators

2.1 Flux Estimator based on Voltage-model

This type of estimator (flux estimator based on voltage model) can be synthesized by using the voltage-based model, as depicted in relation (1):

$$\hat{\Psi}_s = \int (u_s - R_{se} i_s) dt \tag{1}$$

Where: ˆ – represents the estimated value,

e – represents the estimator parameter,

$\hat{\Psi}_s$ – the stator flux estimated value,

u_S – stator voltage,

R_{SE} – the stator resistance estimator's parameter,

i_S – stator current.

There is considered only the real part of the equation and respectively a zero value rotor current (without load torque), the obtained results are valid also in the case of the imaginary axis (with load torque).

The estimator is defined by the relation (2), for the x-axis:

$$\hat{\Psi}_{sx} = \int (u_{sx} - R_{se} i_{sx}) dt \tag{2}$$

The relation (2) implementation in Matlab-Simulink is represented in Figure 1.

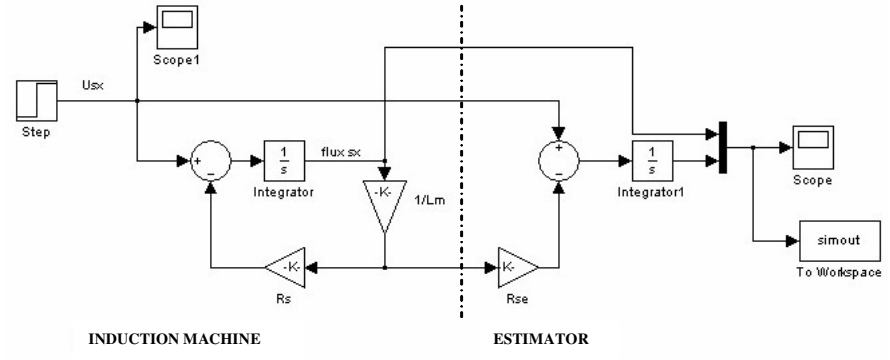


Figure 1

Induction machine's flux estimator based on the voltage model (Matlab-Simulink implementation)

The input variables in the estimator are the machine's current and voltage. In the case that the value of the estimator's parameter stator resistance R_{SE} is identical to the electrical machine stator resistance R_S and the current and voltage are measured without errors (due to noise or offset) then the estimator can be used when the rotor current is not zero. This fact can be noticed in relation (1) where the rotor current doesn't occur.

However, in the case of a slightly different value of the R_{SE} parameter this would lead to errors in the estimator performance especially in the case of low rotation speeds. In Figure 2 is represented the estimator's error in the case when the resistance value is different with a 5% regarding the real value of the resistance (the response at a step signal – the u_{sx} voltage value). It can be noticed, that initially the estimate follows the flux evolution, but further it has a deviation to infinite or to the integrator saturation.

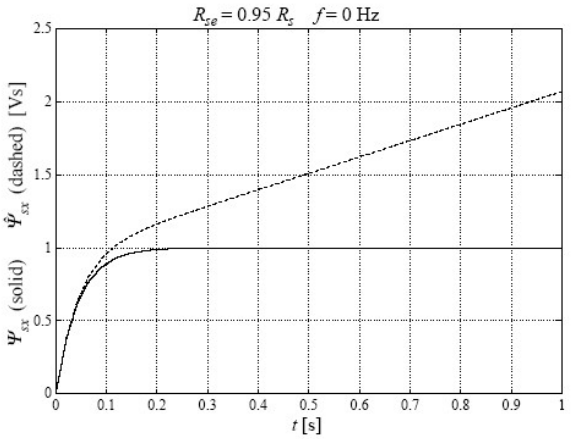


Figure 2
Simulation results (estimator's error)

In Figure 3 there can be noticed that the estimate follows correctly the flux variable, even in the case of a 5 Hz frequency (the resistance error being also 5%). There can be concluded that the influence of the resistance parameter error reduces with the frequency increase. Also, the offset in the current measurement conducts to errors in the flux estimation at any frequency value. Taking in consideration that this estimation method presents difficulties in implementations (especially al low rotation speed), practically it is not used in applications.

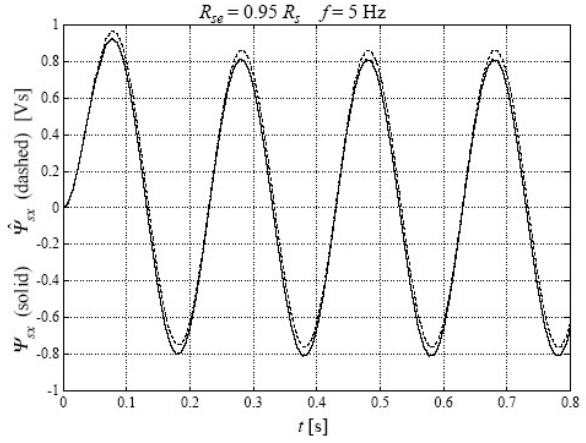


Figure 3
Simulation results (estimator's error at f=5 Hz)

2.2 Flux Estimator based on Current-model

In order to improve to performances of the previous estimator there is used the relation (3).

$$\Psi_s = L_M (i_s + i_r) \tag{3}$$

Where: L_M – magnetization inductance,

i_s –stator current,

i_r – rotor current,

Ψ_s –stator flux

There is considered the rotor current to be zero, and from the relation (3) results the following relation (4):

$$\hat{\Psi}_{sx} = L_{me} i_{sx} \tag{4}$$

Where: L_{me} – the magnetization inductance estimator's parameter.

Therefore, only the stator current represents the input of the estimator structure. In Figure 4, there is represented the Matlab-Simulink implementation of induction machine's flux estimator based on current-model.

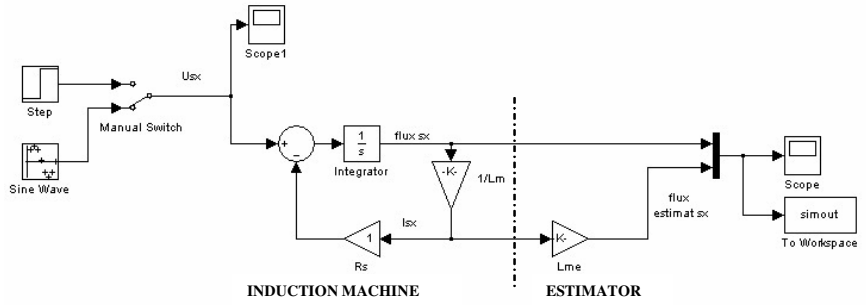


Figure 4

Induction machine's flux estimator based on the current model (Matlab-Simulink implementation)

One of the main disadvantages of this estimator implementation based on the current-model is the fact that a correct estimation of flux requires a very precise value of the magnetization inductance L_M. Another problem is the fact that due to the magnetic saturation phenomenon that occurs in the electrical machines, this parameter (L_M) is not constant. This parameter varies fact that leads to a difficult obtaining of the L_{me} parameter. In Figure 5 there are represented the simulation results – the evolution of the flux estimate in the condition of a 5% error in the magnetizing inductance value. Another disadvantage of this estimator type is the fact that the errors are significant in the case that the rotor current is not zero. In

order to compensate the rotor current effect, it is necessary to measure the rotation speed because the rotor current usually cannot be measured.

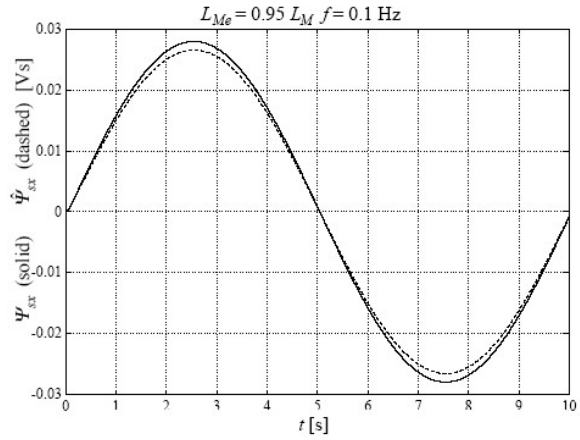


Figure 5
Simulation results (estimator's error)

The previous estimator, based on the voltage-model presents the advantage of being independent to the rotation speed and to the rotor current.

From relation (5) and relation (3) and (4) there is obtained the relation (6).

$$\Psi_r = \Psi_s + L_L i_r \tag{5}$$

Where: L_L – is leakage inductance.

$$\Psi_s = (L_L i_s + \Psi_r)\frac{L_M}{L_L + L_M} \tag{6}$$

In order to calculate the rotor flux, from relation (5), (4), (3) and (7) there is obtained the relation (8).

$$\frac{d\Psi_r}{dt} = jz_p\omega\Psi_r - R_r i_r \tag{7}$$

where: z_p – number of poles pairs,

ω – rotor rotation speed,

R_r – rotor resistance.

$$\frac{d\Psi_r}{dt} = i_s\frac{R_r L_M}{L_L + L_M} - \Psi_r\left(\frac{R_r}{L_L + L_M} - jz_p\omega\right) \tag{8}$$

There results the estimator defined by the relations (9) and (10):

$$\hat{\Psi}_s = (L_{Le} i_s + \hat{\Psi}_r) \frac{L_{Me}}{L_{Le} + L_{Me}} \tag{9}$$

$$\frac{d\hat{\Psi}_r}{dt} = i_s \frac{R_{re} L_{Me}}{L_{Le} + L_{Me}} - \hat{\Psi}_r \left(\frac{R_{re}}{L_{Le} + L_{Me}} - j z_p \omega \right) \tag{10}$$

This estimator has as input the rotor rotation speed and the stator current, as it can be noticed in Figure 6.

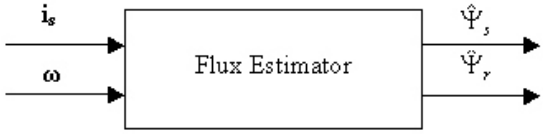


Figure 6
Flux estimator based on current-model

The flux estimator based on current-model presents good performances in the case of low frequencies, but is sensible to the parameter error at high frequencies.

2.3 Flux Estimator based on the Stator Current Estimation

In order to solve the problem of the estimator's based on voltage-model error, there is used the estimate of the stator current instead of the measured value of the stator current variable.

$$\frac{d\Psi_s}{dt} = u_s - R_s i_s \tag{11}$$

From relation (11) and (3), considering the rotor current to be zero, there results the estimator described by relations (12) and (13), represented in Figure 7.

$$\hat{\Psi}_{sx} = \int (u_{sx} - R_{se} \hat{i}_{sx}) dt \tag{12}$$

$$\hat{i}_{sx} = \frac{\hat{\Psi}_{sx}}{L_{Me}} \tag{13}$$

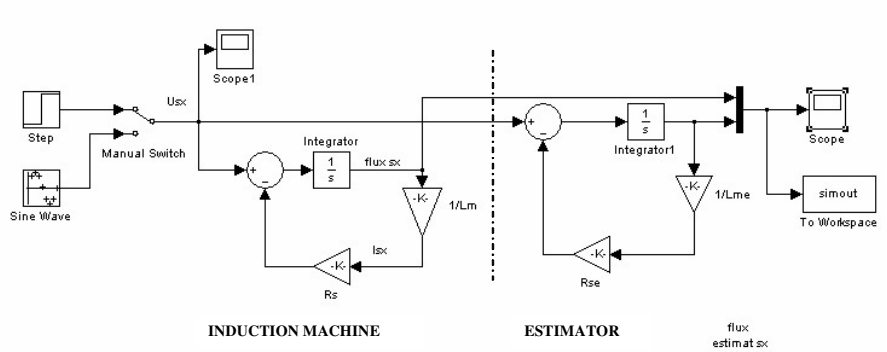


Figure 7
Induction machine's flux estimator based on stator current estimation. (Matlab-Simulink implementation)

This flux estimator is principally a simulation model in open loop, because there are no feedbacks through some measurements. In Figure 8, there can be noticed the reduced value of the estimator's error. However, there is present a steady state regime error in the flux estimate due to the error in the resistance or inductances errors.

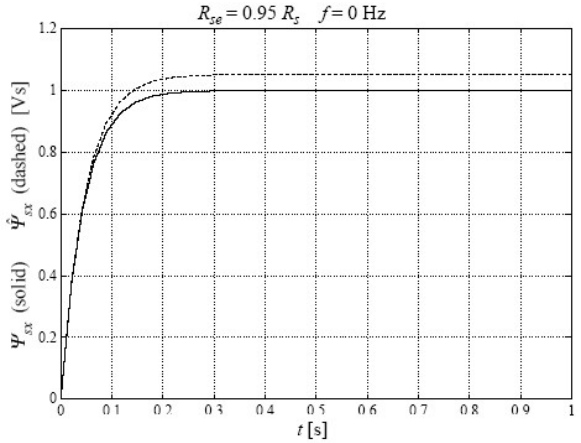


Figure 8
Simulation results (estimator's error)

3 Induction Machine's Flux Observer

3.1 Induction Generator's Flux Observer

Another possibility is the combination of the three estimators' types already considered by using the observer's theory. The Kalman's filter structure can be also applied in the case of the induction machine.

Considering the state representation of a system (described by relations (14), (15)),

$$\frac{dx}{dt} = Ax + Bu \tag{14}$$

$$y = Cu \tag{15}$$

where: x – represents the estimated state, u – represents the system's input, y – represents the system's output described by relation (16):

$$\frac{d\hat{x}}{dt} = A_e\hat{x} + B_e u + K(y - C_e\hat{x}) \tag{16}$$

where: K – represents the gain coefficient (Astrom, 1976), and Ae, Be, Ce - the model's parameter.

In the case of a Kalman filter, the model's parameter are $A=A_e$, $B=B_e$ and $C=C_e$, obtaining the identity observer that follows the entire state vector, in comparison with a reduced set observer that follows only a subset of the state vector (Luenberger, 1979).

There can be obtained a flux observer structure of the induction generator (considering the case when the rotor current is zero) by using the following notations (17), (18), (19), (20), (21), (22), (23):

$$x = \Psi_{sx} \tag{17}$$

$$\hat{x} = \hat{\Psi}_{sx} \tag{18}$$

$$y = i_{sx} \tag{19}$$

$$u = u_{sx} \tag{20}$$

$$A_e = -\frac{R_{se}}{L_{Me}} \tag{21}$$

$$B_e = 1 \tag{22}$$

$$C_e = \frac{1}{L_{Me}} \tag{23}$$

The observer equation is (24):

$$\frac{d\hat{\Psi}_{sx}}{dt} = u_{sx} - \frac{R_{se}}{L_{Me}}\hat{\Psi}_{sx} + K(i_{sx} - \frac{1}{L_{Me}}\hat{\Psi}_{sx}) \tag{24}$$

The choose of the gain coefficient represents a high sensitive task. The literature recommends the study of the poles-zeroes representation. [2] [3] The problem that occurs in the case of the complete order observer (the rotor current is not zero) is that the poles depend by the rotation speed if the *K* coefficient is constant. Another method is represented by the LQG (Linear Quadratic Gaussian) design, in the case that the *K* gain coefficient minimizes the estimate error. This method presents limitations due to the variation of the machine parameter in the functioning regimes. The equation (16) can be reformulated as depicted in relation (25):

$$\frac{d\hat{\Psi}_{sx}}{dt} = u_{sx} - \frac{R_{se}}{L_{Me}}\hat{\Psi}_{sx} + kR_{se}(i_{sx} - \hat{i}_{sx}) \tag{25}$$

Where:

$$k = \frac{K}{R_{se}} \tag{26}$$

$$\hat{i}_{sx} = C_e\hat{\Psi}_{sx} = \frac{1}{L_{Me}}\hat{\Psi}_{sx} \tag{27}$$

In Figure 9 is presented the considered flux observer, described by relations (17) - (23)

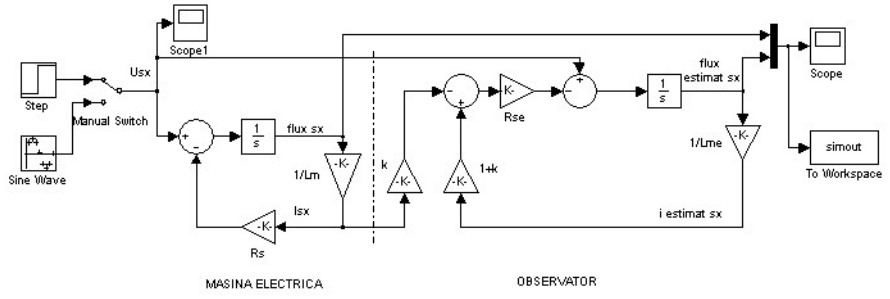


Figure 9
Induction Machine's Flux Observer (Simulink implementation)

Considering the presented Simulink diagram and the equations (17) - (23) and (25), there results:

$$\frac{d\hat{\Psi}_{sx}}{dt} = u_{sx} - R_{se}\left(\frac{\hat{\Psi}_{sx}}{L_{Me}}(1+k) - ki_{sx}\right) = u_{sx} - R_{se}\left((1+k)\hat{i}_{sx} - ki_{sx}\right) \tag{28}$$

The obtained observer is a combination between the flux estimator based on voltage model (Figure 1) and the estimator based on stator current estimation (Figure 7). There can be noticed in Figures 10 and 11, that a choose of an unitary *K* coefficient reduces the steady state regime error of the estimator based on stator current estimation and the deviation of the estimator based on voltage model. The observer pole is the root of the characteristic equation, as depicted in relation (29):

$$\left[(A_e - KC_e) - s\right] = 0 \quad \Rightarrow \left[\left(-\frac{R_{se}}{L_{Me}} - kR_{se}\frac{1}{L_{Me}}\right) - s\right] = 0 \quad \Rightarrow s = -\frac{R_{se}}{L_{Me}}(1+k) \tag{29}$$

If the *K=-1* then the observer becomes the estimator based on voltage model, and respectively if *K=0* then the observer is practically the estimator based on stator current estimation.

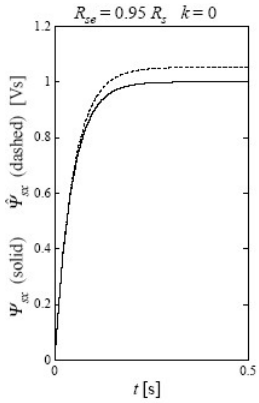


Figure 10
Step response of the observer (*K=0*)

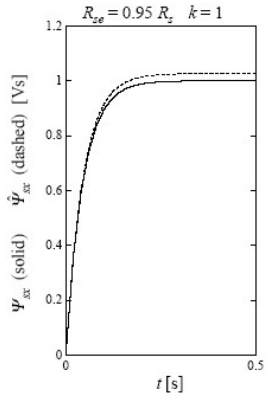


Figure 11
Step response of the observer (*K=1*)

The results obtained by this observer are good, in the case of an +/- 10% variation of the induction's generator parameter, this range covering the issues regarding the machine's parameter variation (due to the temperature, magnetic coupling, etc).

Conclusions

There can be concluded that the solution of using the induction machine's magnetic flux estimation is imposed because this variable must be known in the context of the control structure, especially in the case of FOC structures and due to the fact that this variable cannot be directly measured.

The three estimator structures and the observer presented in the paper present both advantages and limitations that must be considered and studied for each type of application. The problem that must be solved is the design of a flux estimator structure that presents good performances in the case of entire frequency range spectrum. The first presented estimator based on voltage-model presents a better response in the case of higher frequency range; meanwhile the flux estimator based on current-model presents a better response in the range of low frequencies.

In [1] [4], there is described an efficient modality of a combination of those two estimators in order to obtain a estimator with good performances both in case of low frequency range and of high frequency range. The idea that leads to such estimator is to use a low frequency pass filter to select the estimator based on the current-model at low frequency range and of course, a high frequency pass filter to select the estimator based on voltage-model at high frequency range.

Another approach modality that can be used is the observer theory, such as the Kalman filtering that can be applied also in the case of the induction generator. [5] [6]

The above exposed flux estimators and observer present a great significance in the actual trend of the sensorless control structures of the modern windmills based on induction generators. In order to study the windmills control structure performances there is mandatory a detailed analysis of the dynamic behavior of all main components of the wind energy conversion line: wind turbine, electrical generator, converter, grid and other additional elements).

References

[1] H. Siegfried: Grid Integration of Wind Energy Conversion Systems, Publisher: John Wiley & Sons, ISBN 047197143X, 1998

[2] K. D. Hurst, T. G. Habetler, G. Griva, F. Profumo: Speed Sensorless Field-Oriented Control of Induction Machines Using Current Harmonic Spectral Estimation, Proc. IAS '94, 29th Annual Meeting, Denver Colorado, 1994, pp. 601-607

[3] Z. Krzeminski: Speed and Rotor Resistance Estimation in Observer System of Induction Motor, Proc. EPE '91, Firenze, Italy, 1991, pp. 538-542

[4] T. Umeno, Y. Hori, H. Suzuki: Design of the Flux-Observer-Based Vector Control System of Induction Machines Taking into Consideration Robust Stability, Electrical Engineering in Japan, Vol. 110, No. 6, 1990, pp. 53-65

[5] G. C. Verghese, S. R. Sanders: Observers for Flux Estimation in Induction Machines IEEE Transactions on Industrial Electronics, Vol. 35, No. 1, February 1988, pp. 85-94

[6] J. Holtz, J. Quan: Drift and Parameter Compensated Flux Estimator for Persistent Zero Stator Frequency Operation of Sensorless Controlled Induction Motors. IEEE Transactions on Industry Applications, Vol. 39, No. 4, July/Aug. 2003, pp. 1052-1060