ADAPTIVE CLOSED-LOOP CONTROL STRATEGY FOR PARALYZED SKELETAL MUSCLES

Clara IONESCU and Robin DE KEYSER
Ghent University
EeSA Department of Electrical energy, Systems and Automation
Technologiepark 913, B9052, Gent-Zwijnaarde, Belgium
Email: {clara,rdk}@autoctrl.UGent.be

ABSTRACT
Artificial control of paralyzed or paretic muscles became a challenging field and lately an important research topic, stimulated by the advances in technology. In this contribution, classical PI control is compared to the DIRAC (DIRect Adaptive Control) strategy. The DIRAC algorithm is both an auto-tuning and an adaptation method of the controller parameters. Since paralyzed or paretic muscles are time-varying systems, an adaptive/auto-tuning method is necessary for control. Simulations on a muscle model adapted from literature are performed and the controller’s performances are compared. Some of the implementation aspects are also discussed.

KEY WORDS
Paralyzed muscle, simulation, PI controller, adaptive control

1. Introduction
Remarkable research efforts are directed nowadays towards developing various muscle models [1,2] based on the cross-bridge theory [3] for muscle contraction. A simple model of muscle energetics characterizing the metabolic activity of muscle during movement [4] or a parametric model reflecting changes in the mechanical output of skeletal muscle with aging [5] are only a few examples. Another tendency in today’s research is to obtain human locomotion simulations making use of (human) musculo-skeletal models and optimal control strategies for large-scale biomechanical systems [6,7].

With models available, artificial control of paralyzed or paretic muscles became an important research topic. One of the most used methods to perform artificial control of muscles is based on functional electrical stimulation (FES) which enables restoration of movement (e.g. arm, knee). FES-based devices use electric current pulses to stimulate and excite the (intact) peripheral nerves. They produce muscle contraction, generate joint torques, and thus, joint movements. These (invasive) methods are not adopted routinely for the treatment of most patients due to limitations in the available technology. However, therapeutic electrical stimulation for paralyzed or paretic muscles has proved to be an efficacious rehabilitation tool in clinical trials on humans [8]. The recent introduction of radio-frequency controlled, injectable microstimulators (BIONs – BIOnic Neuron) has provided a novel method to deliver electrical stimulation in a way that overcomes some of the problems associated with the conventional stimulation systems (e.g. percutaneous intramuscular systems). BIONs are electrical stimulators that can be injected in one or more muscles through a hypodermic needle. Power and digital command signals are transmitted across the skin to one or more BIONs from an external device [9].

A manifold of control strategies have been applied on nonlinear and linearized muscle models with satisfactory results. Classical control – PI, PID – is simple and generally used for simulation purposes [10,11]. Advanced control techniques – such as adaptive control techniques [10,11] or sliding-mode control [12] – are usually tested in simulations and if successful, implemented afterwards in real-life FES devices.

In this contribution, classical PI control is compared to the DIRAC control strategy (DIRect Adaptive Control). Application of adaptive control is fully justified since the muscle has time-varying properties. The FES devices are usually applied to restore the (initial) properties of skeletal muscles and eventually to perform movements. Another reason to use adaptive control is to obtain a device (e.g. BIONs) that can be used on any patient, without specifying a priori a model of the muscle (as necessary in classical control design). Simulations on a linear (constrained) model from literature [3] are performed and results compared.

In the second section the muscle model and its dynamic characteristics are depicted. Classical PI controller designed via a CAD-package is described, along with the DIRAC strategy in the 3rd section. Comparison between the classical PI controller and adaptive PI controllers designed via the DIRAC method is discussed in the 4th section. Finally a conclusion section summarizes the main outcome of this investigation and formulates some ideas.
concerning the future work. Some of the implementation aspects and their effective impact on results are mentioned in Appendix.

2. A Linear Muscle Model

An impressive number of experimental studies have been performed on amphibians and mammals to obtain information upon skeletal muscle behaviour. Several competing models [1,2,4,5] have been proposed and although abundant experimental data exists, no model is sufficient to explain muscle behaviour under all conditions.

Skeletal muscles contract when activated, and this contraction works against any mechanical load that may be present. If the load is sufficiently high then the movement is prevented and the muscle is considered to be active isometrically (i.e. the muscle length is held constant). Under such conditions, the muscle is considered to perform an isometric contraction. Most of the experimental studies are conducted under isometric conditions. Given these conditions, if a single action potential is delivered to a muscle and a brief burst of force is observed, then the result is termed as isometric twitch.

A linear model capturing the properties of a muscle under isometric conditions can be represented by a 2nd order transfer function:

\[ M(s) = \frac{450}{(s + 5)(s + 20)} \]  

(1)

The parameters (450, 5, 20) are a set of nominal parameters and their value can change (considerably) from person-to-person. These parameters depend on the physical condition of the muscle, age [5] etc. In the case of a paralyzed muscle, important variations are observed during the rehabilitation period. The time delay occurring between the nervous activation and calcium release in muscle to obtain contraction has been ignored in order to avoid implementation and numerical complexity (from control engineering standpoint, it is of minor importance, because this time-delay is very small compared to the system time constants).

The input to such a model is a stimulus occurring with a certain frequency and the output is the force given by the contraction of the muscle, as in Figure 1. The input frequency is limited between 5Hz – 50Hz (the frequencies for which the static characteristic is linear and corresponds to equation (1)).

From a mathematical standpoint, the response \( y(t) \) of the process \( M(s) \) to an input \( u(t) \) (= frequency \( F \)) is the effect of a series of impulses with period \( T (= 1/F) \) applied as input. Starting from the time-domain equation of the input (i.e. a sum of impulses):

\[ s(t) = \sum_{k} \delta(t - kT) \]  

(2)

its Laplace-domain equivalent is:

\[ S(s) = \sum_{k} e^{-kTs} \]  

(3)

The process output is then:

\[ Y(s) = M(s) \sum_{k} e^{-kTs} \]  

(4)

which in time-domain is a sum of delayed impulse responses of system (1):

\[ y(t) = h(t) + h(t - T) + h(t - 2T) + ... \]  

(5)

The value of this signal at the moments corresponding to the impulse series \( (t = kT) \), is:

\[ y(kT) = h(kT) + h((k - 1)T) + ... \]  

or

\[ y_k = h_k + h_{k-1} + ... \]  

(6)

which is approximately:

\[ \frac{1}{T} \int_{0}^{t} h(\tau) d\tau \]  

(7)

Moreover, the integral of the impulse response is the step response, leading to:

\[ y(kT) = \frac{1}{T} g(kT) \]  

(8)

where \( g(kT) \) is the discrete-time step response of \( M(s) \).

The conclusion is thus that a constant (step) input \( u(t) = F \) leads to the output: \( y(kT) = g(kT)F \)  

(9)

In steady-state \( Y = KF \), with \( K \) the static gain of \( M(s) \). This explains why a higher frequency \( F \) leads to a higher force \( Y \) (and the relationship is linear, with gain \( K \)).

A simple experiment (unit impulses) depicted by Figure 2 shows the output of the process corresponding to (9) and, having the time constants: 1/20=50 ms and 1/5=200 ms, a total open-loop settling time of about 1 second.

![Fig.1: Block Scheme of Muscle Model Simulation.](image1)

![Fig.2: Single Twitch (dotted line) and Summation of Twitches (continuous line) at 5Hz (unit) impulse frequency.](image2)
Fig. 3: Unfused Tetanus at 15Hz (dotted line) and Fused Tetanus at 45Hz (continuous line) impulse frequencies.

If a muscle is activated by repeated action potentials (stimulus) the twitches can merge into each other, as depicted by Figure 2. For example, if two (or more) action potentials are close in time, before the force of the first twitch becomes zero, the next twitch is superimposed on the previous. Obviously, the peak force of such a summation will be greater than that of a single twitch. Finally, decreasing the period between the stimuli tends to increase the amplitude of the summation (= output force). When the stimuli period is large, the force rises and falls between the stimuli and this is called un-fused tetanus (tetanus = sustained contraction of a muscle, when induced experimentally; un-fused = discontinuous; fused = continuous) as observed in Figure 3.

Details on the muscle model defined by (1), as well as an overview upon isometric muscles properties are in [3].

3. PI and DIRAC – Controller Design

Controller design is based on 2 approaches: 1) using the CAD (Computer Aided Design) methodology, requiring an available dynamic model of the system and 2) using the auto-tuning principle, which automatically finds a set of PI(D) parameters without an a priori process identification (i.e. no model required).

A brief description is provided in this section and more details can be found in [13,14]. Some implementation aspects with impact on results are discussed in Appendix.

3.1 Computer Aided Design (CAD)

A CAD-software based on Frequency Response techniques (FR-tool) has been used. This ‘in-house’ developed methodology is based on specifications such as: robustness, settling time, overshoot and is a very interactive and visual design tool [13]. The principle is the following: ‘play’ with the PI(D) – zero(s) to reshape the Nichols curve, so that it fits into the requirements of the given specifications. It is necessary to provide a dynamic model of the system.

The dynamic model of the system required by the FR-tool is given by (1). The PI controller design procedure consists of playing with the PI - zero so that the controller proportional gain $K$ is as large as possible, to reduce the settling time ($Ts<1s$), but still satisfying the required specification for overshoot ($OS<15\%$). The zero is located at $z= -5.3$ and the corresponding gain is $K=0.6503$, as depicted by Figure 4.

The PI(D) parameters can further be used in a discrete-time control scheme, with a software implemented controller:

$$ u(t) = u(t-1) + c_0 e(t) + c_1 e(t-1) + c_2 e(t-2) \quad (10) $$

with the error:

$$ e(t) = w(t) - y(t) \quad (11) $$

Denoting the shift-operator: $q^{-1}e(t) = e(t-1)$, results:

$$ u(t) = \frac{c_0 + c_1 q^{-1} + c_2 q^{-2}}{1 - q^{-1}} e(t) \quad (12) $$

and the control loop is depicted in Figure 5.

The sampling period of the controller was $Ts=10$ ms.

The dynamic model of the system, required by the FR-tool is given by (1). The PI controller design procedure consists of playing with the PI - zero so that the controller proportional gain $K$ is as large as possible, to reduce the settling time ($Ts<1s$), but still satisfying the required specification for overshoot ($OS<15\%$). The zero is located at $z=-5.3$ and the corresponding gain is $K=0.6503$, as depicted by Figure 4.
need for a priori specifying a model of the process, thus functioning as an auto-tuning method. Secondly, if used on-line, the PI(D) parameters are adapted continuously, resulting thus in a direct adaptive controller.

The use of auto-tuning or adaptive control seems appropriate for control of skeletal muscles, since they are known to be time-varying [5]. Moreover, if the control is intended on paralyzed muscles, the electrical stimulation is often used not only to produce contraction of the paralyzed muscle, but also to restore its initial properties. The adaptive control method described in this section is easy to understand and simple to apply.

In the context of an unknown process model, the assumption that the muscle (and stimulator) is described by an unknown (discrete-time) transfer function \( M(s) \) leads to the closed loop transfer function (ref. Fig.5):

\[
y(t) = \frac{C(q^{-1})M(q^{-1})}{(1-q^{-1})+C(q^{-1})M(q^{-1})}w(t) \tag{13}
\]

The design performance of the closed loop is specified by a reference model, \( R(q^{-1}) \), given a priori. For example, one of the desired characteristics of the closed loop response can be the settling time. For the present simulation, a first order and a second order reference transfer function has been used and the resulting controllers tested in section 4. The reference models were:

\[
R_1(s) = \frac{1}{1+\tau s} \quad \text{and} \quad R_2(s) = \frac{1}{(1+\tau s)^2} \tag{14}
\]

The task of controller tuning is to find \( C(q^{-1}) \) (i.e. \( c_0, c_1 \) and \( c_2 \)) such that the closed-loop transfer function from (13) approximates the desired reference model \( R(q^{-1}) \). This can be written as:

\[
C(q^{-1})(1-R(q^{-1}))M(q^{-1})y(t) \equiv (1-q^{-1})R(q^{-1})w(t) \tag{15}
\]

Applying (15) to the time-signal \( u(t) \), results in:

\[
C(q^{-1})(1-R(q^{-1}))M(q^{-1})u(t) \equiv (1-q^{-1})R(q^{-1})u(t) \tag{16}
\]

and becomes:

\[
C(q^{-1})(1-R(q^{-1}))y(t) \equiv (1-q^{-1})R(q^{-1})u(t) \tag{17}
\]

Defining the filtered signals:

\[
u_f(t) = (1-q^{-1})R(q^{-1})u(t), \quad y_f(t) = (1-R(q^{-1}))y(t) \tag{18}
\]

and introducing the error signal \( \varepsilon(t) \), (17) becomes:

\[
u_f(t) = C(q^{-1})y_f(t) + \varepsilon(t) \tag{19}
\]

The final step is to estimate (e.g. via least-squares estimator) the parameters in the polynomial \( C(q^{-1}) \) such that the errors \( \varepsilon(t) \) are minimized [15]. A schematic overview of the DIRAC strategy is given in Figure 6.

![Fig.6: Block-scheme of the DIRAC Strategy](image)

Notice that for the simulation presented in this contribution, DIRAC algorithm has been used off-line, for initial tuning of a PI controller. However, the method can be easily implemented on-line, as a direct adaptive controller.

4. Comparison of Different Controllers

The purpose of this section is to evaluate the controllers designed as described in previous section, in order to compare the performance of the auto-tuned controllers to the performance of the CAD-designed controllers. After designing the PI controller by use of the CAD package, two PI controllers have been auto-tuned using the DIRAC strategy.

The reference transfer functions required in the DIRAC strategy, each corresponding to a PI controller, were:

\[
R_1(s) = \frac{1}{1+0.1s} \quad \text{and} \quad R_2(s) = \frac{1}{(1+0.02s)^2} \tag{20}
\]

The experiment consisted in changing the reference set-point (= force) from 45N (10Hz) to 95N (20Hz). On a scale of 22.5N-225N (corresponding to 5Hz-50Hz), a set-point change of 50N is about 25%. The results given by the controllers are depicted in Figure 7 and the corresponding control input given in Figure 8.

It can be easily concluded that similar performance is obtained for the classical control and for the adaptive control. However, as depicted, defining the reference transfer function holds great importance since it represents the desired closed-loop performance. The PI controller designed with the second order reference transfer function has a settling time about 1000 ms but some overshoot (5%). A much simpler reference transfer function – a first order – gives the same settling time (1100 ms) and no overshoot.

As expected, DIRAC1 (with 1st order reference transfer function) reaches the set-point more slowly than DIRAC2 since this was the desired closed-loop performance, given by (20).
Regarding the PI controller, it has similar settling time as the DIRAC controllers. However, the main disadvantage consists of the one-time tuned parameters, and thus not able to cope with a time-varying system as a (paralyzed) skeletal muscle.

In summary, applying adaptive control techniques for skeletal muscle control is strongly motivated by the time-varying character of the system. It has been shown that a similar performance can be obtained with an adaptive control strategy as with a controller which is designed by means of a CAD package. The big advantage is that the DIRAC strategy does not require a model of the process. The controller’s parameters are estimated based on the system response to an excitation signal (e.g. a pseudo-random binary signal) and a reference transfer function with the desired closed-loop characteristics. Thus the time constant(s) of the reference transfer function is a design parameter. This allows the user to design controllers that are specific to the application, and therefore performant.

A next step would be to test the DIRAC strategy on a real-life skeletal muscle model, including time delay. Notice however that the model is necessary in this case only to be able to perform the simulation. In real-life, the DIRAC strategy could be applied directly on the system without a preliminary identification of a model (which is usually required when tuning classical controller parameters).

Another interesting application would be to use a model based predictive control scheme. It has the advantages of tackling the constraints posed on the system (in this case limited frequency) and can deal with non-linear time-varying systems (if extended to a nonlinear muscle model). The disadvantage is that it requires a model of the system and the complexity of the algorithm requires a high performance technology.

### 5. Conclusion

This paper deals with the simulation of a dynamic skeletal muscle model and comparison between the classical PI controller and adaptive PI controllers.

The use of CAD package software to design the classical PI controller required a model of the process (muscle). An advantage of the FR-tool is that of being a very interactive and visual design tool.

The advantage of the DIRAC strategy is that it does not *a priori* require a model of the process. The controller’s parameters are estimated based on the system response to an excitation signal (e.g. a pseudo-random binary signal) and a reference transfer function with the desired closed-loop characteristics. Thus the time constant(s) of the reference transfer function is a design parameter. This allows the user to design controllers that are specific to the application, and therefore performant.

A next step would be to test the DIRAC strategy on a real-life skeletal muscle model, including time delay. Notice however that the model is necessary in this case only to be able to perform the simulation. In real-life, the DIRAC strategy could be applied directly on the system without a preliminary identification of a model (which is usually required when tuning classical controller parameters).

Another interesting application would be to use a model based predictive control scheme. It has the advantages of tackling the constraints posed on the system (in this case limited frequency) and can deal with non-linear time-varying systems (if extended to a nonlinear muscle model). The disadvantage is that it requires a model of the system and the complexity of the algorithm requires a high performance technology.

### References:


Appendix

The software implementation of the stimulator in Figure 1 posed interesting problems, with significant impact on the results. Some of the most important will be described in this section. The platforms used for programming were Matlab® and Simulink®.

In practice, an impulse is considered as an input vector with only one non-zero element. To implement a series of impulses with duration 1ms and variable period it was necessary to define a buffer vector (Fig.1A). Every 10 ms the buffer was adapted to the new frequency (Fig.1A). Notice that implementation of the impulse series at different frequencies requires conversion from frequency to a value which denotes the impulse period T. Since this value has the meaning of a vector’s length, it cannot be fractional thus it is rounded to its nearest integer. For example, if the controller asks for a frequency of 15Hz, the T is 1000/15=66.7 (in ms). Rounding to its nearest integer gives T=70. As a consequence, part of the accuracy of controller’s output is lost.

For the first iteration, a 10-element vector, with the first element non-zero, is sent to the process. At the second iteration, the frequency given by the controller is converted to a period T of length 1000/frequency and the first element non-zero. Since the 1000/frequency value is not always integer, it is rounded to its closest integer value. The position of the last non-zero element in the buffer is found and the new T is added in the buffer starting from that position (the buffer is not restricted to the initial 200 elements length). The new 10 values from 201-210 are send to the process, and the first 10 values of the buffer are cut, shifting everything to the next 10 elements. The procedure is repeated at the next iteration. The procedure proved to be incorrect when the current period T has to be changed to a smaller value (see further Table 1A).

To avoid this situation, the following strategy has been considered. If the position of the last non-zero element in the buffer is smaller than 200 (thus if the above described situation occurs) the buffer is filled with zeros until position 200 and from 201 the new T will be added. In this way, the impulse is send immediately and the period between last non-zero element position in buffer and the new impulse (position 201) will be a transition period, whereas:

old T < transition period < new T

For example, in the situation depicted in Table 1A, the transition period is 90 (Table 1A, bold line: 67+201-178).

Figure 2A depicts the impact on the results of the incorrect software implementation and of the improved strategy.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>position non-zero value</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>249</td>
<td>121 221</td>
<td>100</td>
</tr>
<tr>
<td>250</td>
<td>111 211</td>
<td>100</td>
</tr>
<tr>
<td>251</td>
<td>101 201</td>
<td>100</td>
</tr>
<tr>
<td>250</td>
<td>111 178 245</td>
<td>67</td>
</tr>
<tr>
<td>251</td>
<td>235</td>
<td>67</td>
</tr>
<tr>
<td>252</td>
<td>225</td>
<td>67</td>
</tr>
<tr>
<td>253</td>
<td>215</td>
<td>67</td>
</tr>
<tr>
<td>254</td>
<td>205</td>
<td>67</td>
</tr>
</tbody>
</table>

|Fig.2A: Effect of Correct (dotted line) and Incorrect Implementation (continuous line) on Results|

Notice that implementation of the impulse series at different frequencies requires conversion from frequency to a value which denotes the impulse period T. Since this value has the meaning of a vector’s length, it cannot be fractional thus it is rounded to its nearest integer. For example, if the controller asks for a frequency of 15Hz, the T is 1000/15=66.7 (in ms). Rounding to its nearest integer gives T=70. As a consequence, part of the accuracy of controller’s output is lost.