ELSEVIER

Medical Engineering & Physics 25 (2003) 3–9

Medical Engineering & Physics

www.elsevier.com/locate/medengphy

Technical note

Advanced modeling environment for developing and testing FES control systems

R. Davoodi *, I.E. Brown [1], G.E. Loeb

A.E. Mann Institute for Biomedical Engineering, University of Southern California, 1042 West 36th Place DRB-B12, Los Angeles, CA 90089-1112, USA

Received 13 December 2001; received in revised form 16 April 2002; accepted 25 April 2002

Abstract

Realistic models of neuromusculoskeletal systems can provide a safe and convenient environment for the design and evaluation of controllers for functional electrical stimulation (FES) prior to clinical trials. We have developed a set of integrated musculoskeletal modeling tools to facilitate the model building process. Simulink models of musculoskeletal systems are created using two software packages developed in our laboratory, Musculoskeletal Modeling in Simulink (MMS) and virtual muscle, in addition to one software package available commercially, SIMM (Musculographics Inc., USA). MMS converts anatomically accurate musculoskeletal models generated by SIMM into Simulink® blocks. It also removes run-time constraints on kinetic simulations in SIMM, and allows the development of complex musculoskeletal models without writing a line of code. Virtual muscle builds realistic Simulink models of muscles responding to either natural recruitment or FES. Models of sensorimotor control systems can be developed using various Matlab® (Mathworks Inc., USA) toolboxes and integrated easily with these musculoskeletal blocks in the graphical environment of Simulink.

Keywords: Biomechanics; Musculoskeletal modeling; Simulink

1. Introduction

A functional electrical stimulation (FES) controller must solve problems of sensorimotor coordination similar to those normally handled by the brain and spinal cord. It must continuously control the activation of a number of redundant, non-linear, and non-stationary muscle actuators to successfully move a similarly complex skeletal system in the face of various loads and internal and external perturbations. Design of FES controllers with such sophistication is obviously non-trivial.

In the past, most of the practical FES controllers have been designed by trial and error in the patient [1–7]. But ad hoc design methods are non-systematic and time-consuming processes. Even for simple applications, different operators may produce different results in the same patient. Such methods are likely to be clinically unacceptable when applied to activities of daily living that involve multiple degrees of freedom, such as reaching, grasping, standing, and walking. Systematic methods to optimize adaptive controllers such as neural networks generally require large sets of pseudorandom training movements that are impractical and may even be hazardous if applied to a real patient. This has motivated us to develop a mathematical modeling environment in which customized and accurate models of specific neuromusculoskeletal systems can be readily constructed and used to simulate and develop FES controllers prior to testing in a given patient.

Mathematical models of the musculoskeletal system have been employed in two different FES strategies. One approach is to incorporate the dynamic model in the control algorithm itself in order to facilitate the use of linear methods such as servocontrollers. Because of the highly non-linear nature of the musculoskeletal system, however, this approach has been applied mainly to simple

* Corresponding author. Tel.: +1-213-821-1114; fax: +1-213-821-1120.

E-mail address: davoodi@usc.edu (R. Davoodi).

[1] Present address: CIHR Group in Sensory-Motor Physiology, Queen's University, Kingston, Ontario, Canada.

1350-4533/03/$22.00.
doi:10.1016/S1350-4533(02)00039-5

postural tasks, with limited success [8,9]. In contrast, other researchers have developed forward dynamic models of the musculoskeletal system to analyze its response to various FES control strategies and disturbances [10–18]. In this approach, one does not need to simplify or linearize the mathematical model because the model is not used directly to formulate the FES commands. Therefore, these forward models can represent all the known complexities of the system and play the role of a virtual subject with precisely controllable experimental conditions for the design and evaluation of controllers prior to human trials. Stability and behavior of the system under various conditions, and sensitivity to variations in the model and control system parameters can also be investigated. Modeled experiments can be repeated using different control schemes before proceeding to clinical trials. Model systems have an advantage over real systems because internal variables such as muscle forces and joint torques can be monitored.

Models of many different musculoskeletal systems have been developed by many researchers for many purposes, including analysis of natural behavior as well as control of FES. Development and validation of such models is time-consuming and requires a high level of biomechanical, mathematical and software engineering skills. Because each model is usually developed for a very specific purpose, it tends to be designed and written in a programming environment that is convenient for the developer but not conducive to sharing or reuse of its component parts. Commercial software packages for simulation of mechanical systems have been used to model musculoskeletal systems: ADAMS (Mechanical Dynamics Inc., USA) [10,15], SD/FAST (Symbolic Dynamics Inc., USA) [11,19], DADS (LMS International, Belgium) [20], and working model (MSC Software Corp., USA) [21]. These software packages, however, lack the specialized components specific to biological systems, such as musculotendinous force production and musculoskeletal moment arms. These model components must then be developed in a compatible format and interfaced, if possible at all, with the specific mechanical simulation software. These packages also lack the capability to generate realistic animations of the motion in musculoskeletal systems.

To address these limitations, SIMM (Musculographics Inc., USA) was developed as a specialized software application for developing anatomically realistic musculoskeletal models [22,23]. SIMM provides a Dynamic Pipeline™ utility that links it to SD/FAST, a commercial kinetic modeling package that uses Kane's formulation [24] to generate the dynamic equations of motion (in C-programming language) for the anatomical model. Development of a complete musculoskeletal model with SIMM still requires a full understanding of the C-code generated by SIMM and SD/FAST and the development of additional C-programs to model other system components such as sensors and controllers. Further, there are run-time limitations on the muscle excitation and external forces, which handicap the use of SIMM to study control algorithms. The models of muscle force generation in SIMM are relatively primitive and do not represent more complex dynamical or pathological states.

We are part of a research consortium that has developed a new class of wireless, injectable electronic modules called BIONs™, which can be configured into multichannel systems to stimulate paralyzed muscles in virtually any part of the body [25]. BION1 stimulators are now in clinical trials [26]. BION2 sensors and back-telemetry systems for FES are under development [27]. Because of the generic nature of this technology, we anticipate the need to develop FES control systems for a wide range of patients and functions. This need has motivated us to develop a modular approach to modeling the various electronic interfaces and the musculoskeletal systems in which they will be used. We describe here an integrated modeling environment based on Simulink blocks that can be linked together in a graphical user interface. Simulink runs on top of Matlab, a powerful and widely used modeling and simulation language that includes toolboxes of functions for constructing and analyzing a wide range of control systems.

2. Modeling environment

The modular component models described herein were built using commercially available SIMM as well as two software packages developed in our laboratory: virtual muscle [28] and MMS [29]. As shown in Fig. 1, a Simulink model of an example FES system typically contains models of muscle actuators (built by virtual muscle), musculoskeletal linkage (built by MMS), sensors, and FES controller. Sensors and FES controllers tend to be application specific and are therefore left to the user, who can take advantage of the built-in libraries in Simulink and control toolboxes in Matlab to reduce considerably the development effort.

3. Creating the musculoskeletal linkage

Building a Simulink model of a musculoskeletal linkage starts with the definition of the anatomical model in SIMM. Surface renderings of the individual bones (from a library of 3D scans of cadaver bones) are assembled into an articulated skeleton. Muscles are added based on their bony attachments (usually measured from the cadavers) and constraints on the paths of their tendons (according to anatomical landmarks or moment arms data) (Fig. 2, top). MMS automatically converts the output of SIMM into a Simulink block. MMS consists of a set of Matlab scripts and C-files that are added to the normal

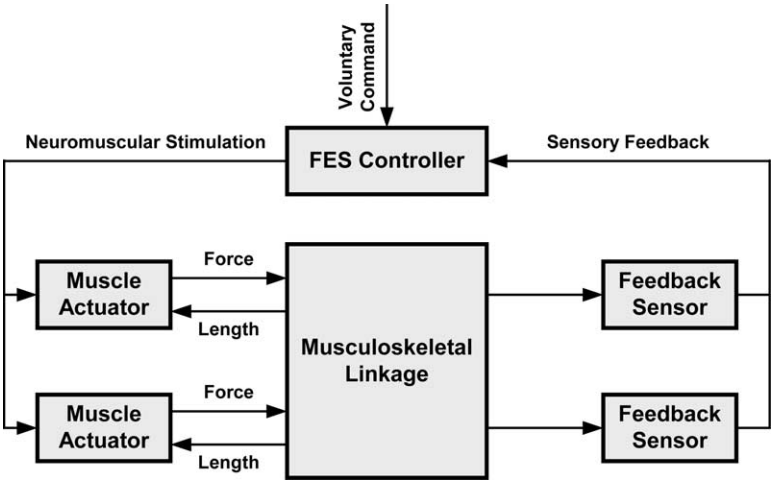

Fig. 1. The Simulink model of a typical FES system. Simulink models of the model components are developed using separate software tools and integrated within Simulink simulation environment.

model building process in SIMM (Fig. 2, middle). MMS generates compiled C-code that calls the SIMM code and files as required. This compiled C-code is wrapped in a Simulink S-function, which permits it to be connected to other Simulink blocks and called during simulations. Within Simulink, the MMS output appears graphically as a large block with intuitively labeled input and output connectors for all of its state variables (Fig. 2, bottom). Once inside Simulink, these connectors can be used to interface the musculoskeletal linkage to other modeled components as shown schematically in Fig. 1.

SIMM operates on its model input files to produce a model-specific parameter file and a skeletal definition file required for SD/FAST. At this point, MMS takes over and automates the rest of the process for building the final Simulink model. MMS first calls SD/FAST to generate C-files representing the equations of motion that compute joint motion caused by external and muscle forces. SIMM's Dynamic Pipeline is a set of C-files that calculate the musculoskeletal geometry for any set of joint angles; MMS can bypass SIMM's default muscle models if requested by the user, permitting the resulting Simulink block to be coupled to other models of muscle force-generation such as virtual muscle. As a result, new customized muscle models can be developed in Simulink and used in place of or in combination with the default SIMM muscle models. (Bypassing of SIMM's default muscle models involves passing musculotendon path length from the MMS-created Simulink block to a separate muscle block which then calculates muscle force and passes it back.) MMS then calls a C-compiler for the C-files generated by SIMM, SD/FAST, Dynamic Pipeline and MMS. The compiled C-files are wrapped into a Simulink S-function and saved in a Simulink model by the MMS Model Builder. MMS runs this intermediate Simulink model only once to extract the parameters of the SIMM model (e.g. the number and names of the muscles and degrees of freedom) to a Matlab M-file. This initial single run also reveals any potential errors in the SIMM model definition files and verifies its integrity before proceeding further.

Next, the MMS code generator uses the inputs from the user to generate a C-file implementing the user-specified external forces and constraints on motion. This implementation eliminates a SIMM limitation that prevents the user from modifying the external forces and motion in run-time. The automatically generated code makes use of the SD/FAST functionality to apply the user-requested forcers and motion. For example, hybrid systems can be modeled by using MMS to prescribe arbitrary motions to the joints or apply joint torques or external point forces to simulate external orthoses, torque motors, external loads and surface contact events. The newly generated files are used in a new round of compilation and model building (dashed lines) to generate the final Simulink model. At run-time, the Simulink model generates a SIMM-compatible motion file that can be used later by SIMM to animate the resulting motion.

The model building process is internally complicated primarily due to our desire to develop MMS without modifying the SIMM and SD/FAST software. This enables current SIMM users to build Simulink blocks from their existing SIMM models without modification. To the external user, the steps required to reconfigure the models and enable the new features are largely invisible.

4. Creating the muscles

Virtual muscle was designed as a stand-alone application that could be used to create Simulink blocks of muscles in a simple, straightforward manner. Although we describe its use here in conjunction with SIMM and MMS, virtual muscle functions independently of these two software packages and can be used with any other Simulink based musculoskeletal linkage model.

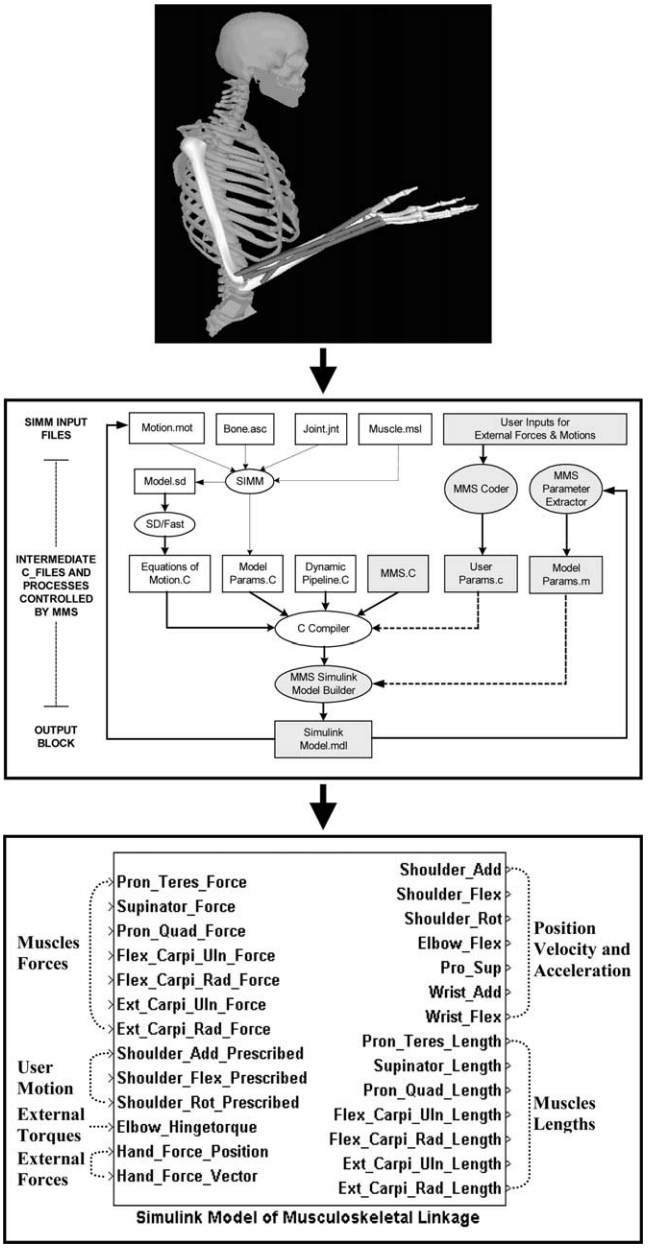

Fig. 2. Procedure for creating the Simulink model of an example musculoskeletal linkage. Top—Anatomical model in SIMM that is usually built using the cadaver measurements taken from the literature. The example model is a 3D model of the human arm with four segments, seven muscles, and seven degrees of freedom, which is built to demonstrate the capabilities of the modeling environment. Middle—MMS structure and its integration with the model building process in SIMM. Shaded boxes and thick lines represent MMS processes. Dashed lines represent iterative processes (see text). Bottom—Final MMS-generated Simulink block. In this example, MMS is instructed to designate the shoulder joint as prescribed, apply a hinge torque to the elbow joint, apply muscle forces to the forearm and wrist joints, and apply a force to the hand to simulate a load. Accordingly, MMS has incorporated the appropriate ports to receive arbitrary values of these inputs from other Simulink blocks in run-time.

Virtual muscle employs a Hill-type model based upon an extensive experimental data set collected from feline muscle [30,31]. The model provides a more accurate description of muscle force production than any model published previously, accounting for the interactive effects of length, velocity and activation (both recruitment and frequency) over the physiological ranges of each. The Hill-type approach of this model, in which active force is an empirical function of activation, length and velocity, was chosen instead of the Huxley-type approach, in which active force is derived from a population of cross-bridges [32], because the Hill-type is computationally much simpler. Some of the more important components of the muscle model are shown in the middle section of Fig. 3. For example, active force is product of the Force–Length (FL), Force–Velocity (FV) and Activation–frequency (Af) relationships.

The Simulink muscle block produced by virtual muscle includes the tendon, the fascicles and the muscle mass between them. This approach permits the model to capture the storage of energy in elastic components and the effects of internal length changes of fascicles on active force-generation and the activity of stretch sensors such as muscle spindles. The Simulink block representing the fascicles is further divided into sub-blocks representing sub-populations of motor units in the muscle. Because a sub-block is either recruited or not, each sub-block represents a sub-populations of motor units that are recruited together; we refer to these sub-blocks as *modeled* motor units to distinguish them from *real* motor units. In the simplest case there would be only one modeled motor unit for each fiber type (i.e. all real motor units of a given fiber type would be recruited together). In order to achieve realistic recruitment and force modulation, however, the total number of modeled motor units in a muscle should probably be 5–10; thus each fiber-type will probably be represented by 1–5 sub-populations of real motor units, each modeled as an independently recruited and frequency modulated modeled motor unit. The only limit on the number of modeled motor units is the available computational time.

The main Simulink muscle block also includes a recruitment block that converts the single activation input to a muscle into firing rates for each of the modeled motor units. Two built-in recruitment strategies are supplied with virtual muscle. Motor unit populations can be recruited and frequency modulated according to a physiological ‘size principle’ or recruited in random order and driven at an externally specified frequency as occurs during intramuscular electrical stimulation [33].

The creation of a Simulink muscle block begins with model descriptions of the fiber-types to be incorporated in the muscle. Fiber-types are defined according to property-specific parameters. For example, each fiber-type’s FL relationship has particular properties, as does each fiber-type’s FV relationship. Users can either use the

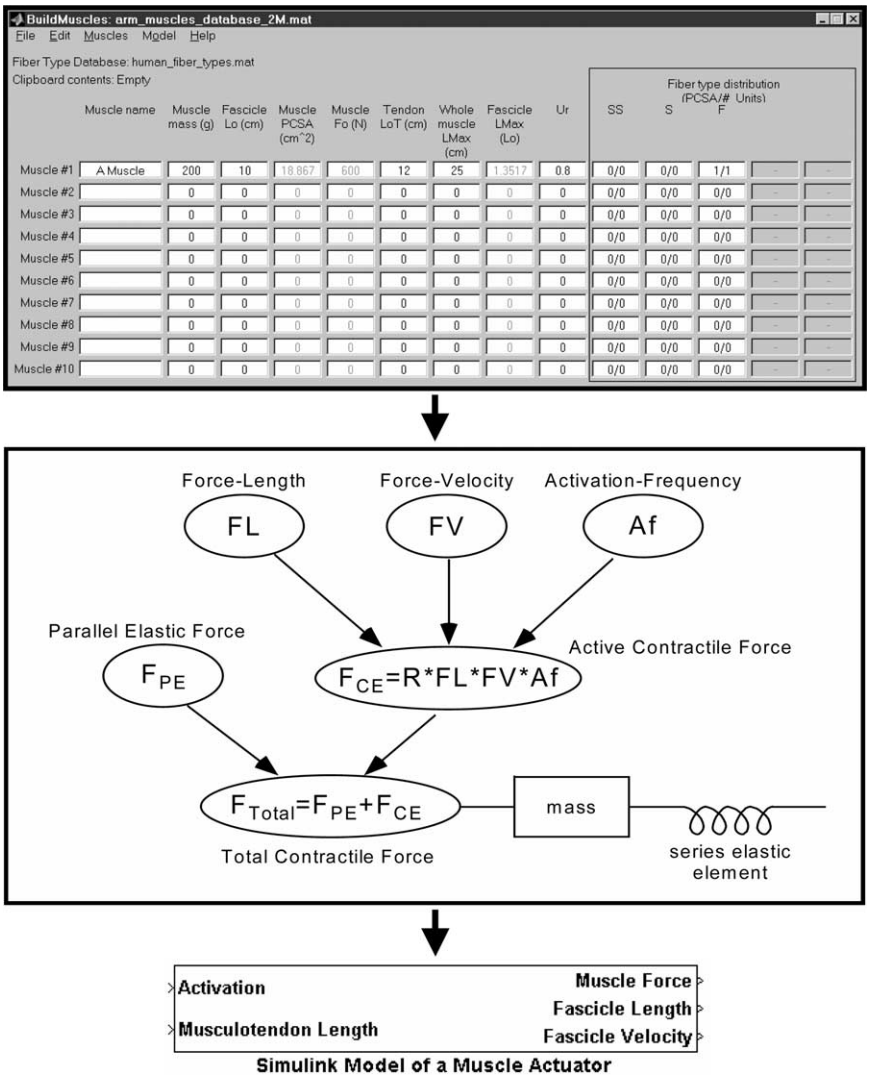

Fig. 3. Procedure for creating the Simulink model of a muscle actuator. Top—Sample screen-shot of virtual muscle's BuildMuscles function. The BuildMuscles function is used to view and manipulate user-defined databases of muscles. Relevant properties that are defined include a muscle's morphometry and fiber-type composition. Middle—The muscle model used by virtual muscle is of the Hill-type variety, and contains the major components shown here: passive elastic element (PE), contractile element (CE), FL, FV, Af. See Cheng et al. [28] for a more complete description of the model. Bottom—Sample Simulink block for a muscle output by virtual muscle. The inputs required by the muscle block are activation and the total musculotendon path length. The minimum output from the block is force, however, additional outputs can be requested such as the two shown here: fascicle length and fascicle velocity.

fiber-types supplied with the basic virtual muscle package or use simple scaling tools in virtual muscle to modify the supplied fiber-types to be faster or slower, or from a different species. The user can also manipulate any parameter of a fiber-type model if so desired.

The second step for creating a Simulink muscle block is to describe a muscle in terms of its geometry, number of motor unit populations and fiber-type description. For example, the optimal fascicle length and mass of each muscle need to be defined. Fiber-type composition is defined as proportions of those fiber-types defined previously and stored in a fiber-type database as described previously. A sample screen-shot of the BuildMuscle function used to describe muscles is shown in the top section in Fig. 3. Once a muscle has been defined, virtual muscle can output a pre-made Simulink muscle block of the muscle. A sample muscle block shown in the bottom section in Fig. 3 includes several outputs. Although MMS and SIMM only require the muscle force output for kinetic simulation, controllers or sensors of a musculoskeletal system may require other outputs, such as fascicle length and fascicle velocity. As shown in the bottom section in Fig. 3, virtual muscle can also output a variety of other relevant data that can then be fed back to a user-defined controller.

Virtual muscle has been designed so that while the experienced user has the choice to manipulate almost all muscle-specific parameters, the naïve user can avoid the

complexities and difficulties of creating and programming their own muscle model, yet still be able to create a complex, realistic model of muscle in a short period of time using simple tools. More importantly, by using a common programming environment such as Simulink, the model can be shared and incorporated easily into larger FES models. The current generation of virtual muscle [28,30,31] uses a model that provides an accurate description of muscle force production, but it provides no way to estimate energy consumption. Energetic efficiency is usually a key performance criterion for optimizing the control of redundant musculoskeletal systems. We are currently rebuilding our muscle model (to be released shortly) in such a way that energy consumption will be easily calculable. Our new approach is to base the excitation dynamics on calcium release/uptake/binding and the activation dynamics of cross-bridge attachment/detachment rates [34]. Such a physiologically based model should provide a strong foundation for models of fatigue, an important property of muscle that is a particularly difficult problem in FES systems. Preliminary simulations indicate that it also accounts well for complex properties such as the activation dependence of the fall time of force, which have been difficult to model accurately in other models to date.

5. Conclusions

Even with the latest commercial software, the development of sophisticated musculoskeletal models is a time-consuming process requiring considerable effort to assemble the necessary morphometric and physiological data and requiring both mathematical and computer programming skills to create the model. The integrated modeling environment presented here can be used by researchers with minimal mathematical and programming skills. It also reduces the amount of new physiological data necessary for a given system by providing the default values for those model parameters that are common in most musculoskeletal models. The user must, however, collect the model-specific parameters such as the segments lengths and inertial parameters, muscle moment arms about the joints, and muscle architectural parameters from cadavers or living subjects. The accuracy of these model-specific parameters and the integrated modeling environment are the two main factors determining the validity of the final musculoskeletal model. The modeling environment is based on SIMM and SD/FAST (extensively tested and validated commercial software), virtual muscle (validated by extensive experimental data [30,31]), and MMS, with the primary role of integrating these validated software to seamlessly work in a single modeling environment. Currently, the most common method for the collection of the model-specific parameters is the use of cadaver measurements because many of the model parameters cannot be measured non-invasively from the living subjects. The main problems with cadaver data are that they are usually from frail subjects, they undergo physiological changes after death, and previously published cadaver measurements are incomplete and the modeler has to mix the data from different sources that may have been taken from different size cadavers. Therefore, cadaver data can be used to develop general models of the musculoskeletal systems to study their behavior and perform sensitivity analysis. Existing measurement techniques can be used to partially customize the musculoskeletal models to individual subjects. Complete customization, however, requires new non-invasive measurement techniques to collect the detailed model parameters from the living subjects.

The modeling environment encourages the development of model systems that are inherently modular and structured, facilitating reuse and sharing of components and tracking of any changes in their internal structure or parameters. The graphical user interface of Simulink makes it easy to inspect and alter the connectivity between model components, avoiding errors that tend to occur if such changes must be performed by rewriting the software code. Simulink automatically handles tasks such as dynamic selection of time-steps to optimize the performance of simulations. Matlab provides a complete set of graphical tools to display quantitative summaries of simulated performance, complementing the animation sequences that can be viewed in SIMM.

This integrated modeling environment is expected to reduce model development time considerably, while improving the traceability of individual components. At the same time, the modeling environment retains the flexibility and expandability that may be required to incorporate models of novel components such as sensors and controllers, pathological components such as atrophied or fatigued muscle, and arbitrarily complex connectivity among them. MMS and virtual muscle are available without charge on our web page at http://ami.usc.edu.

Acknowledgements

Funded by the Alfred E. Mann Institute for Biomedical Engineering.

References

[1] Liberson WT, Holmquest HJ, Scot D, Dow M. Functional electrotherapy: stimulation of the peroneal nerve synchronized with the swing phase of the gait of hemiplegic patients. Arch Phys Med Rehabil 1961;42:101–5.

[2] Long C, Masciarelli V. An electrophysiological splint for the hand. Arch Phys Med Rehabil 1963;44:499–503.
[3] Bajd T, Kralj A, Sega J, Turk R, Benko H, Strojnik P. Use of a two-channel functional electrical stimulator to stand paraplegic patients. Phys Ther 1981;61:526–7.
[4] Peckham PH, Marsolais EB, Mortimer JT. Restoration of key grip and release in the C6 tetraplegic patient through functional electrical stimulation. J Hand Surg [Am] 1980;5:462–9.
[5] Petrofsky JS, Phillips CA, Stafford DE. Closed loop control for restoration of movement in paralyzed muscle. Orthopedics 1984;7:1289–302.
[6] Hoshimiya N, Naito A, Yajima M, Handa Y. A multichannel FES system for the restoration of motor functions in high spinal cord injury patients: a respiration-controlled system for multijoint upper extremity. IEEE Trans Biomed Eng 1989;36:754–60.
[7] Prochazka A, Gauthier M, Wieler M, Kenwell Z. The bionic glove: an electrical stimulator garment that provides controlled grasp and hand opening in quadriplegia. Arch Phys Med Rehabil 1997;78:608–14.
[8] Hunt KJ, Munih M, Donaldson N. Feedback control of unsupported standing in paraplegia—part I: optimal control approach. IEEE Trans Rehabil Eng 1997;5:331–40.
[9] Munih M, Donaldson N, Hunt KJ, Fiona MD. Feedback control of unsupported standing in paraplegia—part II: experimental results. IEEE Trans Rehabil Eng 1997;5:341–52.
[10] Adamczyk MM, Crago PE. Simulated feedforward neural network coordination of hand grasp and wrist angle in a neuroprosthesis. IEEE Trans Rehabil Eng 2000;8:297–304.
[11] Davoodi R, Andrews BJ. Computer simulation of FES standing up in paraplegia: a self-adaptive fuzzy controller with reinforcement learning. IEEE Trans Rehabil Eng 1998;6:151–61.
[12] Davy DT, Audu ML. A dynamic optimization technique for predicting muscle forces in the swing phase of gait. J Biomech 1987;20:187–201.
[13] Khang G, Zajac FE. Paraplegic standing controlled by functional neuromuscular stimulation: part II—computer simulation studies. IEEE Trans Biomed Eng 1989;36:885–94.
[14] Khang G, Zajac FE. Paraplegic standing controlled by functional neuromuscular stimulation: part I—computer model and control-system design. IEEE Trans Biomed Eng 1989;36:873–84.
[15] Lemay MA, Crago PE. A dynamic model for simulating movements of the elbow, forearm, and wrist. J Biomech 1996;29:1319–30.
[16] Popovic DB, Stein RB, Oguztoreli MN, Lebiedowska MK, Jonic S. Optimal control of walking with functional electrical stimulation: a computer simulation study. IEEE Trans Rehabil Eng 1999;7:69–79.
[17] Riener R, Fuhr T. Patient-driven control of FES-supported standing up: a simulation study. IEEE Trans Rehabil Eng 1998;6:113–24.
[18] Yamaguchi GT, Zajac FE. Restoring unassisted natural gait to paraplegics via functional neuromuscular stimulation: a computer simulation study. IEEE Trans Biomed Eng 1990;37:886–902.
[19] Davoodi R, Andrews BJ. Optimal control of FES-assisted standing up in paraplegia using genetic algorithms. Med Eng Phys 1999;21:609–17.
[20] Gerritsen KG, van den Bogert AJ, Hulliger M, Zernicke RF. Intrinsic muscle properties facilitate locomotor control—a computer simulation study. Motor Control 1998;2:206–20.
[21] Loeb GE, Brown IE, Cheng EJ. A hierarchical foundation for models of sensorimotor control. Exp Brain Res 1999;126:1–18.
[22] Delp SL, Loan JP. A graphics-based software system to develop and analyze models of musculoskeletal structures. Comput Biol Med 1995;25:21–34.
[23] Delp SL, Loan JP. A computational framework for simulating and analyzing human and animal movement. Comput Sci Eng 2000;2:46–55.
[24] Kane TR, Levinson DA. Dynamics: theory and application. New York: McGraw-Hill, 1985.
[25] Cameron T, Loeb GE, Peck RA, Schulman JH, Strojnik P, Troyk PR. Micromodular implants to provide electrical stimulation of paralyzed muscles and limbs. IEEE Trans Biomed Eng 1997;44:781–90.
[26] Richmond FJR, Loeb GE, Dupont AC, Bagg SD, Creasy JL. Clinical trials of Bions™ for therapeutic electrical stimulation. In: Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Istanbul, Turkey, 2001.
[27] Loeb GE, Peck RA, Moore WH, Hood K. BION system for distributed neural prosthetic interfaces. Med Eng Phys 2001;23:9–18.
[28] Cheng EJ, Brown IE, Loeb GE. Virtual muscle: a computational approach to understanding the effects of muscle properties on motor control. J Neurosci Methods 2000;101:117–30.
[29] Davoodi R, Loeb GE. Conversion of SIMM to SIMULINK for faster development of musculoskeletal models. In: Proceedings of the Sixth Annual Conference of the International Functional Electrical Stimulation Society, Cleveland, USA, 2001. p. 282–4.
[30] Brown IE, Cheng EJ, Loeb GE. Measured and modeled properties of mammalian skeletal muscle. II. The effects of stimulus frequency on force–length and force–velocity relationships. J Muscle Res Cell Motil 1999;20:627–43.
[31] Brown IE, Loeb GE. Measured and modeled properties of mammalian skeletal muscle: IV. Dynamics of activation and deactivation. J Muscle Res Cell Motil 2000;21:33–47.
[32] Huxley AF. Muscle structure and theories of contraction. Prog Biophys Mol Biol 1957;7:255–318.
[33] Singh K, Richmond FJ, Loeb GE. Recruitment properties of intramuscular and nerve-trunk stimulating electrodes. IEEE Trans Rehabil Eng 2000;8:276–85.
[34] Brown IE, Loeb GE. Toward the ultimate skeletal muscle model. In: Proceedings of the 25th Annual Conference of the American Society of Biomechanics, San Diego, USA, 2001. p. 323–4.