TECHNICAL NOTE

JOURNAL OF APPLIED BIOMECHANICS, 2002, **18**, 357-365

A Software Tool for Faster Development of Complex Models of Musculoskeletal Systems and Sensorimotor Controllers in Simulink™

Rahman Davoodi and Gerald E. Loeb
University of Southern California

Computer models of the neuromusculoskeletal systems can be used to study different aspects of movement and its control in humans and animals. SIMM with Dynamics Pipeline (Musculographics Inc., Chicago) and SD-Fast (Symbolic Dynamics Inc., Mountain View, CA) are software packages commonly used for graphic and dynamic simulation of movement in musculoskeletal systems. Building dynamic models with SIMM requires substantial C programming, however, which limits its use. We have developed Musculoskeletal Modeling in Simulink (MMS) software to convert the SIMM musculoskeletal and kinetics models to Simulink (Mathworks Inc., Natick, MA) blocks. In addition, MMS removes SIMM's run-time constraints so that the resulting blocks can be used in simulations of closed-loop sensorimotor control systems.

Key Words: modeling, movement control, SIMM, SD-Fast

Introduction

Movement in humans and other animals is the result of complicated interactions involving voluntary command signals, sensory receptors, reflex circuits, muscle actuators, skeleton, and environment. In order to understand the integrated system or to replace damaged portions with neural prosthetic components, movement must be studied in its entirety. Most of our current knowledge about sensorimotor control is the result of direct measurements from the neurons, muscles, and limbs of naturally moving persons. Such studies are difficult and limited when the person is neurologically impaired, creating the need for clinical analytical systems that employ models to make inferences about the workings of internal components which are not directly observable. Computer models of the neuromusculoskeletal system can extend and complement experimental studies. Their parameters are accessible for inspection and/or modification. The performance of the system can be simu-

The authors are with the A.E. Mann Institute for Biomedical Engineering, University of Southern California, 1042 West 36th Place, Rm DRB-B12, Los Angeles, CA 90089-1112.

lated with different versions of the components in order to help us understand pathological conditions and develop strategies for their treatment.

The validity of model-based predictions depends on the accuracy and completeness of the models, however. Developing musculoskeletal models is often as challenging as recording data from behaving participants (subjects) and requires a high level of biomechanical, mathematical, and software engineering skills. Because each model is usually developed for a specific purpose, it tends to be designed and written in a programming environment that is convenient for the developer but not conducive to the 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., Ann Arbor, MI) (Adamczyk & Crago, 2000; Lemay & Crago, 1996), SD-FAST (Symbolic Dynamics Inc., Mountain View, CA) (Davoodi & Andrews, 1998, 1999), DADS (LMS International, Leuven, Belgium) (Gerritsen, van den Bogert, Hulliger, & Zernicke, 1998), and Working Model (MSC Software Corp., Santa Ana, CA) (Loeb, Brown, & Cheng, 1999). However, these software packages lack the 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 at all possible, with the specific mechanical simulation software. These packages also lack the capability to generate realistic animations of the motion in musculoskeletal systems.

SIMM was developed as a special software environment for creating anatomically realistic musculoskeletal models (Delp & Loan, 1995, 2000). The user generates, or otherwise obtains, a set of input files describing bone surfaces, articulations, and muscle-tendon parameters, and uses SIMM to assemble these graphically into an anatomically realistic model (Figure 1). With the help of SD-Fast, SIMM generates a set of files in the programming language “C” containing the equations of motion for the musculoskeletal model that can be compiled and used for simulations.

The model generated in this way has substantial limitations on its ability to incorporate run-time changes of muscle excitation, external forces, prescribed motion, and initial conditions, which handicap its use in studying control algorithms. For example, the muscles can only be excited in an open-loop manner while many applications involve closed-loop control of muscles. Furthermore, the models of muscle force generation in SIMM are relatively primitive and do not represent more realistic properties or pathological states. Although additional programming can eliminate these limitations, it requires C-programming skills and familiarity with the structure of the SIMM-generated C-programs. Further, any other component required by the system under study (e.g., sensors, command signals, controllers) must be programmed in C.

Simulink is an attractive and popular software package for building models of complex systems because it provides a graphical interface to help the user assemble and navigate multicomponent systems. The user can invoke the powerful Matlab language and toolboxes to build components, organize simulation sequences, and render the results graphically. The block-oriented structure of Simulink facilitates reuse and sharing of code among researchers. One particularly relevant example is the Virtual Muscle™ package (Cheng, Brown, & Loeb, 2000) that is freely available over the Internet <http:ami.usc.edu>. This Matlab program generates

Figure 1 — Example arm model in SIMM window. SIMM uses such graphical user interfaces to accept user construction and modification of the modeled musculoskeletal components and to animate motion of the model system.

Simulink blocks representing the force-generating properties of realistic muscle models with user-specifiable fiber types, recruitment schemes, and architectures.

Methods

MMS automatically converts the output of SIMM into Simulink blocks without the user having to write any C-code at all. MMS consists of a set of Matlab scripts and C-files that are added to the normal model building process. 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

Nomenclature —

GH = glenohumeral
HU = humeroulnar
RU = radioulnar
RC = radiocarpal
ABAD = abduction/ adduction
IER = internal/external rotation
FE = flexion/extension
PS = pronation/supination
URD = ulnar/radial deviation
BIC_ln = biceps long
BIC_sh = biceps short
BRA = brachialis
BRD = brachioradialis
TRI_ln = triceps long
TRI_lt = triceps lateralis
TRI_m = triceps medialis

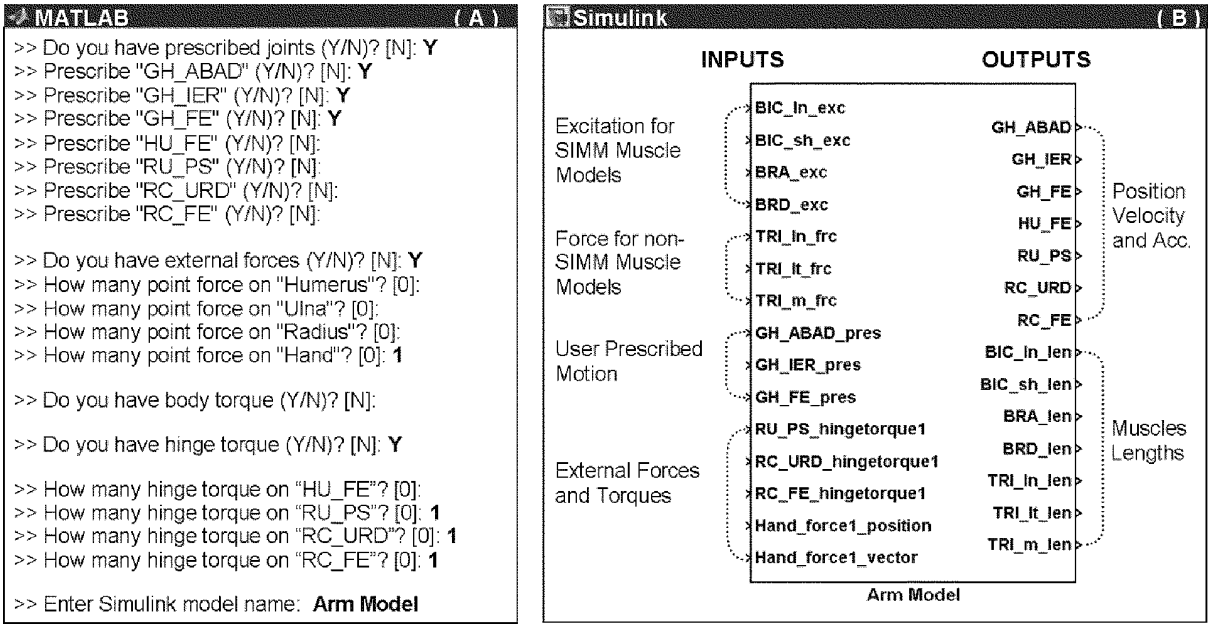

Figure 2 — (A) Interactions between the MMS script and the user to configure and build the Simulink model of the example arm model. User's inputs are in bold. This process can be repeated to configure the model differently. The user has decided to prescribe the motion of all DOF of the GH joint, apply a point force to the Hand, apply hinge torques to the RU and RC joints, and leave the HU joint free to be controlled by the muscles. (B) Final MMS-generated Simulink block with appropriate I/O ports. The user connects the I/O ports to run the simulation.

output appears graphically as a large block with intuitively labeled input and output connectors for all its state variables (Figure 2B). Once inside Simulink, these connectors can be used to interface the musculoskeletal model to other modeled components and graphical display windows.

SIMM itself operates on the model input files to produce a model-specific parameter file and a skeletal definition file for SD-Fast, as shown in Figure 3. At this point, MMS takes over and automates the rest of the process for building the final Simulink model, as depicted by the interactions between the user and MMS in Figure 2A.

1. 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 handle the muscle geometry for any set of joint angles; MMS can bypass its default muscle models if requested by the user, permitting the resulting Simulink block to be coupled to other non-SIMM 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. During the simulation, MMS uses the Dynamic Pipeline's functions to calculate the muscle length (based on the tendon path and insertion defined in SIMM), which is made available to the non-SIMM muscle model. The non-SIMM muscle model uses the length information and its activation level to calculate total muscle force as a scalar. MMS applies the muscle force to muscle attachment points along the line of action as determined by the Dynamic Pipeline's functions. This implementation preserves the existing functionality and data structure.

2. 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 such as the number and names of the muscles and degrees of freedom (DOF) to a Matlab M-file. This single run also reveals any potential errors in the SIMM model definition files and verifies their integrity before proceeding to the next stage where new MMS features can be added to the SIMM model.

3. The MMS code generator uses the inputs from the user to generate a C-file implementing the user-specified external forces and motion. Because external forces in the SIMM's input file are expressed in segmental reference frame, most common external forces such as those measured by force plates (usually expressed in a stationary ground reference frame) must be continuously recalculated for the moving segment's changing orientation. While maintaining this option, MMS allows the user to express the external forces in ground reference frame. The MMS implementation also eliminates an SIMM limitation that prevents the user from modifying the external forces and motion in run-time. Automatically generated code makes use of the SD-Fast functionality to apply the user-requested forces and motion.

4. The newly generated files are used in a new round of compilation and model building (Figure 3, dashed lines) to generate the final Simulink model. At run-time, the model generates an SIMM-compatible motion file that can be used later by SIMM to animate the resulting motion.

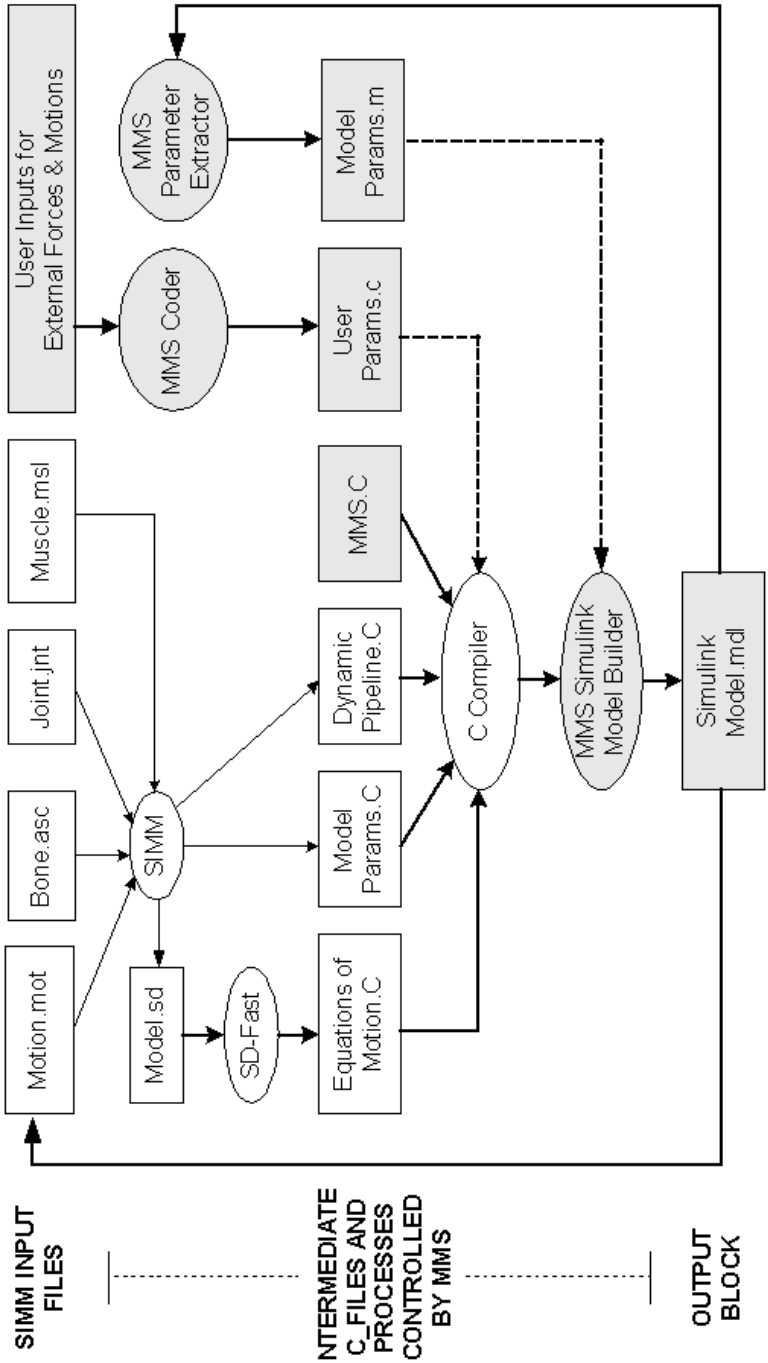

Figure 3 — Model development process and file structure; shaded boxes and thick lines represent MMS processes, dashed lines represent iterative processes. See text for description of process flow.

The complicated model-building process is due primarily to our desire to develop MMS without modifying the SIMM and SD-Fast software. This enables the current SIMM users to build Simulink models of their existing SIMM models without any further modification. To the MMS user, however, reconfiguration of models and attachment of new features appears to be simple. In order to construct and run a typical model, the user needs only to connect the inputs (e.g., muscle excitations or forces generated by other blocks) of the Simulink block to appropriate sources and run the simulation. The resulting motion can be animated within SIMM.

More advanced applications can take advantage of the utilities, toolboxes, and easy programming environment in Matlab and Simulink. For example, various Matlab toolboxes can be used to design controllers for functional electrical stimulation (FES), the Matlab optimization toolbox can be used to adjust them, and Matlab scripts can be used to sequence iterations of the model to study the effects of adaptive control algorithms or the sensitivity of the model to systematic changes of its parameters.

To demonstrate the model-building process, the anatomical model of a simplified arm with 4 rigid segments, 7 DOF, and 7 muscles is presented below. Figure 1 shows the model as it appears in SIMM. The corresponding dynamic model is shown in Figure 2B as it appears in Simulink after MMS. First the model specifications must be entered into the SIMM model definition files. This is normally done within SIMM using a combination of imported files such as 3-D renderings of bones and graphically constructed models of muscle attachments and tendon paths. These files can be read by SIMM to represent the model anatomy (Figure 1) and automatically generate the three sets of C-files for basic dynamic simulation (Figure 3). At this point, MMS script is run to build the final Simulink block (Figure 2A).

In responses to the MMS queries, the user indicates that he/she wishes to prescribe all three DOF of the GH joint, apply a point force to the hand to simulate a load, apply hinge torques to RU and RC joints to simulate torque motors, and leave the HU joint free to be controlled by the muscles. The user has also indicated (in SIMM's muscle file) that the first four muscles will use SIMM's default muscle models and the next three muscles will use non-SIMM muscle models (e.g., Virtual Muscle). Accordingly, MMS incorporates the appropriate input ports to receive these inputs from other Simulink blocks. For example, SIMM default muscle models must be fed with the muscle excitation levels while the non-SIMM muscle models receive the muscle force as computed by those external muscle model blocks. The outputs include the muscle lengths required as inputs to those muscle models, which can also be used to drive models of sensors such as muscle spindles.

Discussion

MMS is the product of our experience with SIMM. We expect MMS to considerably reduce the time required to develop and test models of complete systems by eliminating the need to write C-code and by using the Simulink environment to configure and run simulations using a library of model components. In addition, MMS provides three major enhancements to the capabilities of SIMM:

1. MMS allows the use of any non-SIMM model of muscle force production in place of or in combination with the default SIMM muscle models. This enables the users to develop their own muscle models within Simulink. Virtual Muscle is one of these non-SIMM muscle force production models that has been developed in Simulink to incorporate more realistic properties than the existing SIMM muscle models. The SIMM-based Simulink block generates musculotendon path-length information as one input to each Virtual Muscle block; it receives the force output from each Virtual Muscle block and applies it to the skeleton. The excitation input to the muscles is unconstrained at run-time, permitting simulations of closed-loop feedback and adaptive control. The Virtual Muscle blocks also provide a richer set of output signals such as muscle fascicle length and tendon strain, which are useful for construction of proprioception models for sensorimotor feedback in such closed-loop models.

2. MMS allows the user to change the initial configuration of the musculoskeletal model in each simulation run without recompilation. This includes restricting DOF in the model, as long as the basic topology is preserved.

3. MMS allows the interactive configuration and run-time modification of the external forces, external torques, and prescribed motions. The implementation allows the user to modify the magnitude and application point of the external forces and torques in run-time. The prescribed motions can also be modified in run-time, including locking or unlocking the joints.

Because the C-code generated by SIMM and SD-Fast is accessible, the user has great flexibility in modifying it to suit the specific needs of the problem at hand. However, this requires a great deal of C-programming skills, familiarity with the C-code generated by SIMM, Dynamic Pipeline and SD-Fast, knowledge of mechanical dynamics, and time. By eliminating these requirements, MMS facilitates further the anatomical model development process offered by SIMM while increasing its flexibility for simulations.

To develop dynamic models of the musculoskeletal systems with SIMM, one must obtain four separate licenses: SIMM, Dynamic Pipeline, SD-Fast, and a C-compiler. To use MMS, one needs to also license Matlab with Simulink because MMS-generated models run in Simulink.

Simulink and Matlab are used widely for teaching and research, which should facilitate sharing and reusing modeled components. Their tools make it easy to add blocks simulating sensors, reflexes, pattern generators, muscles, and optimization/learning algorithms. For example, we use Matlab's Neural Network Toolbox to develop FES controllers and study their effects. MMS provides the flexibility required to use Virtual Muscle blocks that model electrically stimulated muscles responding in run-time to candidate FES controllers for reach and grasp.

Researchers, clinicians, and educators in the fields of motor control and biomechanics are among the potential users who may benefit from using MMS. A free copy of MMS can be obtained from the website of *JAB*. A free copy of Virtual Muscle and the latest updates to MMS can be obtained from <http:ami.usc.edu>

References

Adamczyk, M.M., & Crago, P.E. (2000). Simulated feedforward neural network coordination of hand grasp and wrist angle in a neuroprosthesis. *IEEE Transactions on Rehabilitation Engineering*, **8**, 297-304.

Cheng, E.J., Brown, I.E., & Loeb, G.E. (2000). Virtual muscle: A computational approach to understanding the effects of muscle properties on motor control. *Journal of Neuroscience Methods*, **101**, 117-130.

Davoodi, R., & Andrews, B.J. (1998). Computer simulation of FES standing up in paraplegia: A self-adaptive fuzzy controller with reinforcement learning. *IEEE Transactions on Rehabilitation Engineering*, **6**, 151-161.

Davoodi, R., & Andrews, B.J. (1999). Optimal control of FES-assisted standing up in paraplegia using genetic algorithms. *Medical Engineering & Physics*, **21**, 609-617.

Delp, S.L., & Loan, J.P. (1995). A graphics-based software system to develop and analyze models of musculoskeletal structures. *Computers in Biology and Medicine*, **25**, 21-34.

Delp, S.L., & Loan, J.P. (2000). A computational framework for simulating and analyzing human and animal movement. *Computing in Science & Engineering*, **2**, 46-55.

Gerritsen, K.G., van den Bogert, A.J., Hulliger, M., & Zernicke, R.F. (1998). Intrinsic muscle properties facilitate locomotor control—A computer simulation study. *Motor Control*, **2**, 206-220.

Lemay, M.A., & Crago, P.E. (1996). A dynamic model for simulating movements of the elbow, forearm, and wrist. *Journal of Biomechanics*, **29**, 1319-1330.

Loeb, G.E., Brown, I.E., & Cheng, E.J. (1999). A hierarchical foundation for models of sensorimotor control. *Experimental Brain Research*, **126**, 1-18.

Acknowledgments

Development of MMS was funded by the A.E. Mann Institute for Biomedical Engineering. We would like to thank Drs. Ning Lan and Ian Brown for helpful discussions and suggestions.