Model-Based Development of Neural Prostheses for Movement

Rahman Davoodi*, Chet Urata, Markus Hauschild, Mehdi Khachani, and Gerald E. Loeb, *Member, IEEE*

***Abstract*—Neural prostheses for restoration of limb movement in paralyzed and amputee patients tend to be complex systems. Subjective intuition and trial-and-error approaches have been applied to the design and clinical fitting of simple systems with limited functionality. These approaches are time consuming, difficult to apply in larger scale, and not applicable to limbs under development with more anthropomorphic motion and actuation. The field of neural prosthetics is in need of more systematic methods, including tools that will allow users to develop accurate models of neural prostheses and simulate their behavior under various conditions before actual manufacturing or clinical application. Such virtual prototyping would provide an efficient and safe test-bed for narrowing the design choices and tuning the control parameters before actual clinical application. We describe a software environment that we have developed to facilitate the construction and modification of accurate mathematical models of paralyzed and prosthetic limbs and simulate their movement under various neural control strategies. These simulations can be run in real time with a stereoscopic display to enable design engineers and prospective users to evaluate a candidate neural prosthetic system and learn to operate it before actually receiving it.**

***Index Terms*—Functional electrical stimulation (FES), musculoskeletal modeling, prosthetic arms, simulation, virtual reality.**

I. Introduction

CONTROLLERS in functional electrical stimulation (FES) systems for paralyzed limbs and in motorized prosthetic limbs for amputees are complex because they must solve problems of sensorimotor coordination similar to those normally handled by the central nervous system. These controllers must continuously coordinate the actions of nonlinear, redundant, and nonstationary actuators such as muscles to move the limb with kinematically redundant joints and mechanically interacting segments to desired targets in 3-D space. Further, these movements must be performed in the face of external perturbations and in coordination with residual voluntary movements so that man and machine operate in harmony. Design of controllers with such sophistication is obviously nontrivial. Even the relatively simple systems that have been available clinically have relied on and often taxed the subjective judgment of experienced prosthetists [1]–[4]. Such *ad hoc* design methods are time consuming, they produce inconsistent results among therapists, and they become impractical when extended to more complex prosthetic systems and to a wider range of activities of daily living.

Systematic design methods in other fields (e.g., aeronautics, chemical processing, etc.) usually employ a mathematical model of the plant as a test-bed in which to design and test controllers even before the plant is manufactured. Similarly, developers of neural prostheses have been interested in using mathematical models and computer simulations of prosthetic and paralyzed limbs to design and evaluate prosthetic control systems systematically before actual clinical implementation [5]–[12].

In these model-based analyses, computerized mathematical models serve as test-beds with precisely controllable experimental conditions to analyze the performance of the neural prosthesis under various operating conditions, analyze the stability and sensitivity to model and control parameters, and examine internal variables that are not accessible in experimental trials. Despite its potential, however, applications of model-based analysis in design and fitting of neural prostheses have been very limited. One of the main reasons is the inability to anticipate and model the voluntary commands from the patient that are an integral part of the neural prosthetic system. Indeed, the ability of a subject to learn to operate the prosthetic system without undue visual and mental attention is critical for clinical acceptance. To avoid this difficult modeling problem, most researchers have abandoned model-based analysis in favor of trial-and-error experimentation with actual prosthetic systems. For systematic model-based analysis, some researchers have developed simplified models of the voluntary control system [6], [11], [13], some have treated it as a disturbance that must be rejected by the neural prosthetic controller, and some have tried to eliminate its effects by constraining the patient or instructing them not to use their voluntary movements [14]–[16]. As a result, model-based analyses have been limited to simple applications such as controlling the force of a single muscle [17] or controlling the movement of a single joint [18], [19], where voluntary contributions are completely absent. But real-world neural prostheses require significant voluntary contributions from the patient that are purposeful and coordinated. These actions control the movement of the residual limb while at the same time a prosthetic controller operates the prosthetic or paralyzed part of the limb. Further,

Manuscript received July 25, 2006. This work was supported in part by the Alfred Mann Institute for Biomedical Engineering, University of Southern California, by the National Science Foundation Engineering Research Center for Biomimetic Microelectronic Systems (EEC-0310723), and by Defense Advanced Research Projects Agency's Revolutionizing Prosthetics contract to the Applied Physics Laboratory, Johns Hopkins University. *Asterisk indicates corresponding author.*

*R. Davoodi is with the Alfred Mann Institute for Biomedical Engineering and the Department of Biomedical Engineering, University of Southern California, 1042 Downey Way, Los Angeles, CA 90089 USA (e-mail: davoodi@usc.edu).

C. Urata is with the Urata Corporation, Waianae, HI 96792 USA (e-mail: chet@uratacorp.com).

M. Hauschild is with the Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: hauschil@usc.edu).

M. Khachani is with the Alfred Mann Institute for Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: khachani@usc.edu).

G. E. Loeb is with the Alfred Mann Institute for Biomedical Engineering and the Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: gloeb@usc.edu).

Digital Object Identifier 10.1109/TBME.2007.902252

the patient may provide continuous voluntary commands to the prosthetic system in the form of residual limb movements or electrical activity of the residual muscles, peripheral nerves, or neurons in the cerebral cortex. The voluntary contributions to movement of the prosthetic system are significant and must be included in any systematic analysis method. Furthermore, they can be expected to change over time, similarly to the acquisition of any new motor skill. The rate of learning is itself a major determinant of clinical acceptability.

As a framework for systematic model-based analysis of prosthetic systems, we have developed a software tool known as musculoskeletal modeling software (MSMS) that can be used to model and simulate paralyzed and prosthetic limbs. Real-time patient-in-the-loop simulations allow the users to investigate systematically the dynamics of interactions between adaptation and learning in the patient's central nervous system and the neural prosthetic control system without having to model the voluntary contributions of the subject. A similar approach is used in flight simulators where a human subject can learn to operate a complex machine in a safe environment. Here, we will discuss the applications of model-based analysis to design and fitting of neural prostheses and review the currently available software tools for such analyses. Then we will describe the motivation for and the status of the MSMS software development project in our laboratory and its application to design and fit neural prosthetic systems.

II. Applications of Models of Neural Prostheses

We envision three types of simulation analyses that can be used to facilitate the design and fitting of neural prosthetic systems: 1) dynamic simulations for model-based design and evaluation of neural prosthetic controllers; 2) real-time dynamic simulations with the patient-in-the-loop for controller design and patient training; and 3) dynamic simulations for clinical fitting of neural prostheses to individual patient with specific disability.

Dynamic Simulations of Neural Prostheses: The purpose of these simulations is to predict the movement of the limb in response to a given control strategy. Researchers and developers can use these simulations to test and compare different combinations of actuators and control strategies to narrow the choices and come up with the best strategy for a given control problem. These simulations may be performed repeatedly to optimize the parameters of the controller or to perform sensitivity analysis. Because these simulations do not have to run in real-time, they can be made complex by inclusion of model details and components.

Dynamic Simulations of Neural Prostheses With the Patient in the Loop: In these simulations, instead of modeling the patient's voluntary actions, the patient is placed in the simulation loop. These patient-in-the-loop simulations are usually performed in a virtual reality environment (VRE) where the patient can generate voluntary commands to operate a simulated limb. The resulting motion of the simulated limb can be displayed to the patient from his/her point of view as stereoscopic visual feedback. This enables the patient to see the consequence of his/her voluntary actions and, therefore, learn to operate the candidate neural prosthesis before actually receiving it. The main requirements for these dynamic simulations are the real-time execution of simulation and availability of interfaces between the simulation code and the patient via interface hardware. This type of simulation will also be useful in experiments with behaving nonhuman primates to determine the adequacy of command information that can be derived from novel interfaces such as cortical microelectrode arrays [20], [21].

Dynamic Simulations for Clinical Fitting of Neural Prostheses: The two simulation analyses previously described are general and can be applied to any neural prosthetic system. They are also aimed at more sophisticated users such as researchers, engineers, and developers who can use them to investigate different solutions to a specific fitting problem to come up with the best solution. Once an acceptable solution is found, the simulation models and the fitting methods must be packaged in an easy to use environment for clinical application. The graphic user interface (GUI) of such specialized applications must expose only the relevant functionality for the clinical fitting of a specific neural prosthesis to a specific clinical problem. It must also allow the model of the physical plant to be modified to accurately describe the physiognomy and clinical condition of each individual patient.

III. Current Methods and Tools for Neural Prosthetic Modeling

Researchers and developers of neural prostheses have performed model-based analyses of their systems by writing their own computer programs, using commercial software tools, or a combination of commercial and in-house software.

Software programs for modeling and simulation of neural prostheses are complex and difficult to develop. In-house development of such software (especially manual derivation of the equations of motion) is difficult and prone to errors. Further, each model is usually developed for a very specific purpose and tends to be designed and written in a programming environment that is convenient for the developer but not conducive to easy maintenance, sharing or reuse of its component parts. There are many such programs that have taken a long time to develop but have been discarded after the completion of the specific project or student thesis because they could not be shared or used by people other than the original developers.

Some clinical researchers have used commercial modeling software developed for other applications. For example, software packages for simulation of mechanical systems have been used to model musculoskeletal and neural prosthetic systems: ADAMS (Mechanical Dynamics Inc.) [5], [22], SD-Fast (Symbolic Dynamics Inc.) [6], [13], DADS (LMS International) [23], and Working Model (MSC Software Corp.) [24]. These software packages relieve the user from the error prone and painstaking process of deriving and programming the equations of motion. However, they lack the specialized components specific to physiological systems, such as models of muscle force production and complex 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, real-time animations of the motion in musculoskeletal systems or model man-machine interactions in

patient-in-the-loop simulations that are essential for the design and evaluation of clinical systems.

The first specialized software for development of anatomically realistic musculoskeletal models was SIMM [25], [26]. The user generates (or otherwise obtains) a set of input files describing bone surfaces, articulations, and muscle-tendon parameters, and uses SIMM to graphically assemble these components into an anatomically realistic model. With the help of SD-Fast, SIMM generates a set of files in the C programming language containing the equations of motion for the musculoskeletal model that can be compiled and used for dynamic simulations. As specialized musculoskeletal modeling software, SIMM provides important functionality that simplifies the creation of musculoskeletal models. A model generated by SIMM has limitations on its ability to incorporate run-time changes of muscle excitation, external forces, and prescribed motion, thus handicapping its use to study neural prosthetic control systems. For example, the muscles can only be excited in an open-loop manner, while many applications involve closed-loop control of muscles. Overcoming these limitations 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 by the user.

As a result of these limitations in the existing software, many researchers involved in modeling neural prostheses have been forced to develop and maintain their own modeling software, either to replace or augment the limited commercial software now available. For example, our firsthand experience with SIMM led us to develop an add-on software package known as MMS [27], [28] to enhance SIMM's capabilities and overcome some of its limitations. MMS allows modelers to add interactive external forces, prescribed motion, and sensors to their SIMM model without writing any code. It then automatically exports the resulting model to the dynamic simulation environment of Simulink in MATLAB (Mathworks Inc.) that is familiar to most academic users and provides access to additional toolboxes and utilities for model-based analysis.

Although the SIMM musculoskeletal modeling software was enhanced greatly by the features provided by MMS, it was still not suited for real-time patient-in-the-loop simulations. Furthermore, the required set of software tools was expensive, it was difficult to learn and use, it was difficult to maintain, and more importantly, it was slow to adapt to the ever growing needs of the community. These shortcomings motivated efforts in various laboratories to develop a new generation of software for musculoskeletal modeling and model-based analysis of neural prostheses. Our efforts started in 2000 with a series of meetings with stakeholders in the field and led to the launch of the MSMS project in late 2002. MSMS, whose features and applications are described in the following, is being developed as an open-source software tool for model-based analysis and fitting of neural prosthetic systems.

In 2004, the National Institutes of Health established a Center for Biomedical Computation at Stanford University that is tasked with the development and free distribution of open-source software for physics-based simulation of biological structures at multiple scales—from molecules to organisms. One of the main groups in the center is the neuromuscular biomechanics group that is led by Scott Delp, the original developer of SIMM. This group is developing open-source software for physics-based simulation of biomechanical systems. The development and free distribution of such open-source software is expected to greatly facilitate research and provide the framework to allow individual researchers to share their models and software with the community (for information on this effort visit www.simbios.stanford.edu). The software developed by this group and MSMS have a common objective which is the development and sharing of realistic musculoskeletal models. But MSMS focuses in addition on the modeling of prosthetic limbs and the use of prosthetic and human limb models to design and fit prosthetic control systems to the patients. In order to facilitate sharing of models between these and other software environments, we are developing a standard format for representation of musculoskeletal models as described in the following.

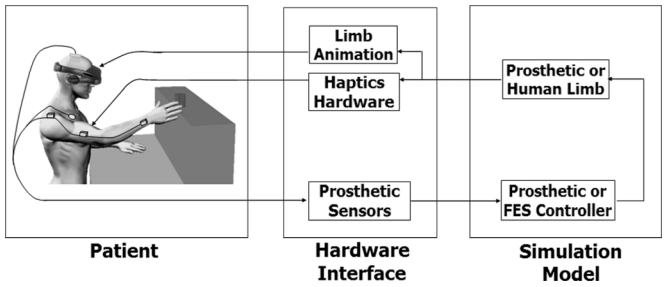


Fig. 1. Main components in neural prostheses for restoration of limb movement to paralyzed and amputee patients. In both cases, the patient generates some form of voluntary command signals that are measured by the prosthetic sensors. The sensory information is fed to the prosthetic controller to calculate the control signals. The simulation model predicts the movement of the actual limb in response to the control actions and other external forces and perturbations. The movement and interactions of the limb with the environment are sent to the patient via visual and haptic displays so that he/she can evaluate the outcome of his/her control actions and adjust them, if necessary, to perform the movement task successfully.

IV. MSMS Software for Modeling and Simulation of Neural Prostheses

A. Goals and Objectives

The main purpose of MSMS is to provide a user-friendly environment for researchers and clinicians alike to model and simulate the behavior of complex neural prosthetic systems for paralyzed and amputee patients. As shown in Fig. 1, these two types of prosthetic systems have many commonalities and, therefore, can be modeled in the same software environment. Models of both prosthetic systems consist of three main blocks: the patient, the simulated model of the prosthetic limb and controller, and the hardware/software interface between the two. The patient generates command signals such as movement of the residual limb still under voluntary control, electromyographic (EMG) activity of intact or reinnervated muscles, or activity of neurons recorded from the peripheral or central nervous system. These voluntary signals indicate the intentions of the patient and enable him/her to control the operation of the neural prosthesis. In the simplest form, these command signals may be used to turn

the neural prosthesis or one of its functions on/off but they more typically provide multiple, continuously varying control signals to enable continuous control of the limb movement. The patient also receives feedback in the form of visual and/or haptic signals that help him/her evaluate the control outcomes. The simulation model block contains the mathematical model of the human or prosthetic limb and the model of the FES or prosthetic control system. The hardware interface block connects the patient in the real world to the simulated limb in the virtual world.

B. Analysis of Requirements

Modeling and simulation software for academic research are usually developed by researchers who have no training in computers and, therefore, are very unlikely to follow professional software development practices. The result is usually software tools that are quickly developed to serve the needs of a specific project but are difficult to maintain or use by others. Given the complexity of the MSMS project, and to ensure that it satisfies all of its requirements and is easy to maintain in the future, we decided to employ professional software development methods. These included systematic analysis of requirements and development and testing of MSMS to satisfy the requirements. Therefore, the first step in development of MSMS was to analyze its requirements before writing a single line of code. In a series of meetings between the software developers and researchers with domain expertise, the requirements were captured and documented in different levels of detail.

The analysis of key requirements justified the decision to develop MSMS, identified the main users such as the researchers and clinicians, identified the key requirements such as the need for accurate models of human and prosthetic limbs that can be easily interfaced to models of feedback control systems, and identified the need for fast simulations for controller optimization and real-time patient-in-the-loop simulations. We concluded that MSMS must have the flexibility to evolve and grow in the future as new models and data become available and new applications are identified. We also identified the key components of MSMS such as the user interface capabilities, classes of model components, and types of model-based analyses. The next step in analysis of requirements was the analysis of different use case scenarios in which step-by-step sequence of actions required to complete different model-based analyses such as "building a limb model" or "performing forward dynamic simulation" were captured from the end users and documented. Use case analysis is an effective method for capturing the requirements from the users who know the problem they need to solve but do not have the programming expertise to translate it into software requirements. Finally, from the key requirements and the use case scenarios, a more detailed set of requirements were derived for MSMS software.

To satisfy the previous requirements, we designed the software architecture for MSMS that is shown in Fig. 2. The architectural block diagram of MSMS has three top-level blocks: GUI, modeling and simulation, and database that are connected via standard application program interfaces (APIs). The architecture allows us to change the GUIs to serve users with different levels of expertise and interface to third party applications and devices. APIs are interleaved allowing any block to be replaced without requiring the rewriting of the other blocks.

C. Software Development and Testing

We have used an iterative software development process to develop and test MSMS software. The process consists of a sequence of short iterations, each implementing, integrating, and testing a subset of required features. Therefore, each iteration period (three weeks in our case) results in fully functioning and tested software with few additional features. The iterative process is particularly suited to a complex software project such as MSMS whose full set of requirements is difficult to anticipate at the beginning of the project.

The main parts of MSMS such as the GUI, the modeling unit and the database are developed in the Java programming language, which supports sophisticated tools such as OpenGL for graphical rendering. The computationally intensive simulation unit is implemented in Simulink using Simulink library blocks and C programming language. The design and interface with control systems is also performed in the Simulink simulation environment, which provides many useful tools for the development of control systems and a graphical schematic representation of the system architecture.

D. Features and Capabilities

The development effort to date has resulted in an integrated environment for modeling and simulation of FES and prosthetic systems that are described as follows.

Development of Standard Model Representations: Presently, musculoskeletal models are associated with the text file that specifies their structure and parameters. Every existing package defines its own proprietary file format independently. This proliferation of file formats makes it difficult to share, reuse, and extend models among different groups. Our solution was to develop a standard file format that would be acceptable to all users and could be expanded indefinitely to deal with unanticipated components, reducing the need for individual model formats. We have developed a standard file format using extensible markup language (XML), which now includes the definitions of the components found in various neural prosthetic systems such as the bones, joints, muscles, wrapping objects, and electric motors. MSMS can also translate models of human or prosthetic limbs developed in other popular applications. For example, models of human limbs defined in SIMM and models of prosthetic limbs defined in SolidWorks (SolidWorks Corp.), can be imported into MSMS. Once loaded, the model can be modified in the 3-D graphical environment of MSMS and exported in XML file format for sharing with other users and applications. The conversion from our standard file format to other representations is also straightforward. Such conversions involve writing translators that are easier to develop for XML representations whose rules, structure, and semantics are defined in a schema. Furthermore, there are many software tools for working with XML files. The availability of our standard representation and its schema will allow the other applications to develop translators to covert to/from our standard XML file format.

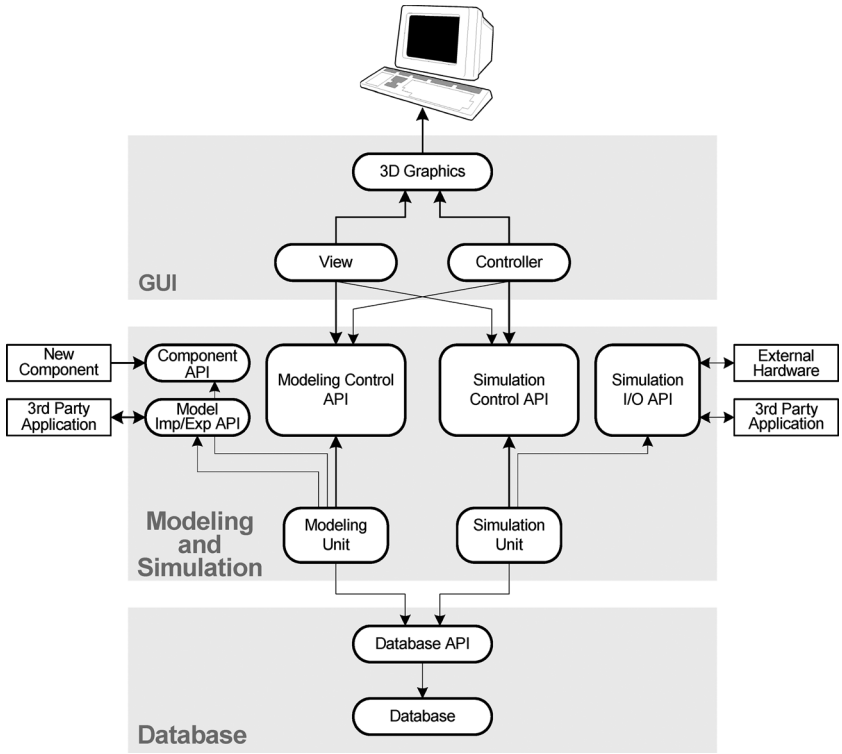


Fig. 2. Architectural diagram of MSMS showing its main units: GUI, modeling and simulation, and database. The GUI is the conduit for the user to view the model and control all MSMS functions. Modeling and simulation provides tools for model building and simulation. It has standard APIs that separates it from the GUI allowing the developers to change the design of the GUI to meet the needs of different users such as researchers and clinicians without modifying the underlying code. To ensure continued expansion of capabilities in the future, this unit has utilities to add new model components and import/export models from/to third party applications. The database unit stores model data in standard XML file formats.

GUI Tools for Model Construction and Manipulation: A properly designed GUI simplifies the construction, modification, and validation of a particular model system. Even when complete data are available to construct a model, it is often difficult to incorporate them in a computer model using currently available software such as SIMM. For example, it is a major job to iteratively change the position of muscle attachment points and wrapping objects to achieve consistency with measurements of muscle moment arms. The user has to change one parameter at a time, plot the moment arm, and visually compare it to the experimentally measured moment arms for validation. More intuitive GUI tools and analytical methods must be developed to allow easy construction and validation of the models. The GUI of MSMS can load, visualize, animate, and graphically edit 3-D models of human and prosthetic limbs (see Figs. 3 and 4). The model can be viewed from a camera that can be positioned by the mouse or key strokes. The model can be animated by the motion data from a saved file or streamed in real-time from a motion capture system or a dynamic simulation of the same model. The muscle attachment points can be edited graphically in anatomical coordinates. The graphical representation of the muscle is composed of the muscle fascicles and the in-series tendon whose properties can be edited separately and in varying degrees of detail. The user can access the parameters and more detailed views of the model components by clicking on the graphical objects. For example, the high-level properties window for a muscle shows its mass, length and fiber composition while lower-level views reveal the force production properties of each fiber type. Among the features to be added is a context-sensitive mouse also known as "smart mouse" that will allow the user to use the mouse in different intuitive ways depending on the object being edited. For example, after selecting the origin of a muscle, the motion

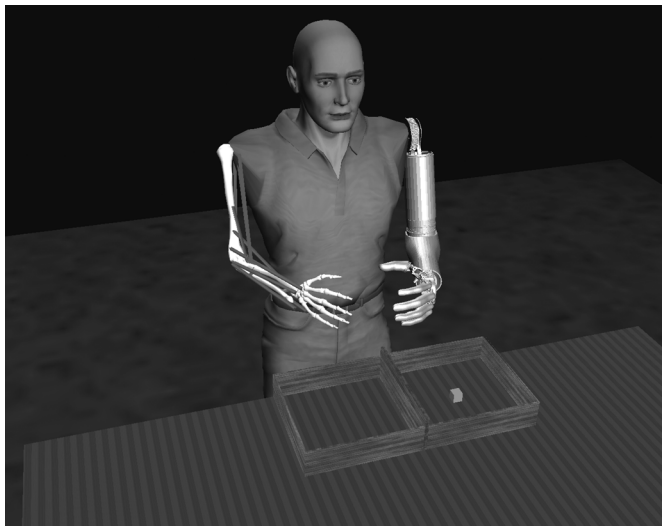


Fig. 3. MSMS integrated environment for modeling of human limbs, prosthetic limbs, and the rehabilitation tasks. A model subject is shown with human right arm appropriate for FES investigations and a prosthetic left arm appropriate for prosthetic applications. The box and blocks rehabilitation task is also modeled in which the patient must move the block from one box to the next.

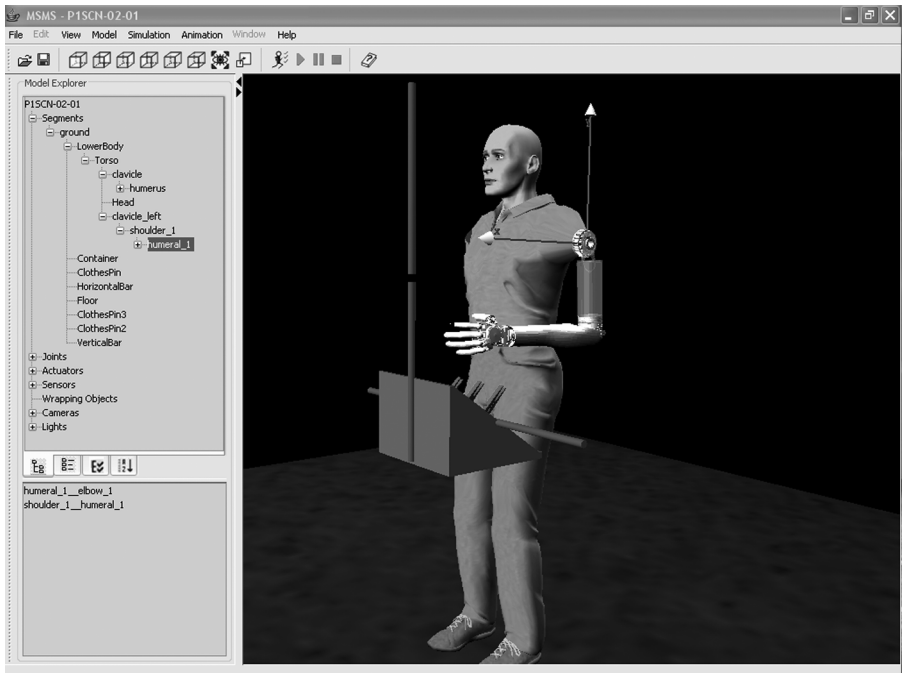


Fig. 4. MSMS screenshot showing the model of an amputee patient with prosthetic left arm performing virtual clothes pin task where the patient has to move the clothes pins from horizontal bar to vertical bar.

of the mouse moves the attachment point along the surface of the bone rather than in the x–y plane of the screen.

In addition to the GUI tools that help the modeler to edit model parameters manually, analytical methods can be used to automate the process of automatically finding "best-fit" sets of parameters to match the model to available data from the system. For example, the moment arm of a muscle at a given joint depends on its origin and insertion on the skeleton plus the shape and size of intermediate objects around which the tendon is deformed. Currently, MSMS does not have such tools but it has been designed to facilitate incorporation of optimization techniques such as those used by Garner and Pandy [29] or other automated parameter fitting algorithms in the future.

Muscle Wrapping Around Bony Surfaces: In MSMS, the wrapping of muscles around bony surfaces is modeled by wrapping algorithms. Different geometric shapes modeling the bony surfaces are placed on the path of the muscle. Then the wrapping algorithms are used to calculate the path of the muscle from its origin to its insertion as constrained by these objects [30]. MSMS currently supports cylindrical and spherical wrapping objects. Each muscle can have as many wrapping objects as necessary to model its contacts with different bony surfaces.

Dynamic Simulations: To perform the dynamic simulations of the human or prosthetic limb models, first, the differential equations governing their motion must be derived. These equations are then integrated numerically to predict the motion of the limb over time. There are several dynamic engine software packages that can be used by MSMS to automatically derive the equations of motion for its limb models. The choice of the dynamic engine, however, is an important decision with great implications. For example, the choice of SD-Fast as the only dynamic engine for SIMM significantly limited its user base because it increased both the cost and difficulty of use. To avoid such pitfalls, MSMS architecture is designed to accommodate any dynamic engine package that might be necessary for different applications and users. To choose the appropriate dynamic engines for initial implementation in MSMS, we performed a side-by-side comparison of several dynamic engines such as SD-Fast, Autolev (Dynamics Online), and SimMechanics (Mathworks Inc.). Sample multibody models were used to benchmark the speed and accuracy of these dynamic engine packages. We also considered other features such as support for closed-loop topologies that are important in musculoskeletal systems and implementation concerns such as ease of integration with MSMS. From this comparison, we chose to implement SimMechanics as the first dynamic engine for MSMS because it uses fast order (N) formulation to derive equations of motion, it has a large collection of joints, actuators and sensors, it can handle open- and closed-loop structures, it can enforce different types of position and velocity constraints, and it is readily available to academic users who are one of the main groups targeted by MSMS. Autolev uses Kan's formulation that produces very efficient code but because of its design it could not be integrated with MSMS. SD-Fast has both Kane's and order (N) formulations and can be easily integrated with MSMS. SD-Fast is more expensive than SimMechanics but it produces equations that execute faster. For example, we applied these dynamic engines to derive the equations of motion for the skeletal model of the human arm (right arm in Fig. 3) with 9 degrees-of-freedom (DOF). The equations where then numerically integrated in Simulink using fourth-order Runge–Kutta algorithm with fixed time step of 1 ms. The average execution time for each second of simulation in a personal computer (PC) with 3.6 GHz Pentium processor was 0.425 s for SimMechanics and 0.124 s for SD-Fast equations. Therefore, the code gener-

ated by SD-Fast for this model executed 3.4 times faster than SimMechanics. This speed advantage that is mainly due to the customization of the generated code to the model parameters might be essential for complex models especially if they have to run in real-time. Currently, MSMS uses SimMechanics as its primary dynamic engine but implementation of SD-Fast is planned for the future.

Once the neural prosthetic model is built, MSMS can automatically build and run dynamic simulations of the model in Simulink. SimMechanics is used as the dynamic engine while other Simulink library blocks and C programs (wrapped in Simulink S-function blocks) are used to simulate other components of the system such as preprocessing of command signals and the operation of feedback control loops. The advantage of exporting MSMS models to Simulink is that the Simulink environment is familiar to most users, it is affordable, and it provides many additional toolboxes for control, optimization, math, and signal processing that can facilitate investigations.

Real-Time Dynamic Simulations With the Subject in the Loop: For simulations where the patient or able-bodied subject is in the loop, the dynamic simulations must run in real-time. The Windows PC operating system is generally poorly suited to achieving reliable real-time performance with complex models and display systems. For MSMS, we have developed a real-time simulation environment that uses MATLAB's xPC Target toolbox [31]. The simulation model of the neural prosthetic systems is first exported to Simulink by MSMS. Simulink itself is not real-time capable, but its models can be compiled and executed in a separate real-time PC (the xPC) with minor modifications. Real-time simulations in our lab are performed in a VRE consisting of 3 PCs that are connected to each other, the subject, and the measurement and display devices via input/output (I/O) hardware (see Fig. 5). The main reason for distribution of VRE elements in multiple PCs is that some algorithms need to be executed in real-time but do not require sophisticated video output. The visualization of the VRE on the other hand requires high quality 3-D video output but does not need to be executed in real-time as long as minimum delay and high frame update rate can be ensured. These fundamentally incompatible specifications require the separation of these two classes of algorithms to different PCs, one with a real-time kernel for fast real-time code execution and the other a Windows machine with good video performance but poor real-time capabilities. We use Flock of Birds (FOB) magnetic motion tracking system (Ascension Technology Corp.) to measure the arm movements, but the VRE system can accept properly formatted data from any motion capture technology. A gyro-based three-axis sensor (3DM-GX1, Microstrain Inc.) measures the head movements that are used to display the movement of the virtual limb to the subject from his/her point of view. Analog and digital signals from the subject, such as EMG, control switches, and external triggers are digitized by a general purpose data acquisition board (PCI-6040E, National Instruments Corp.). The real-time xPC samples all these inputs to the prosthetic controller, calculates the control outputs, and runs the physics-based simulation of the virtual arm to predict its movement as the result of the control inputs and external forces in real-time. The predicted movement is sent to visualization

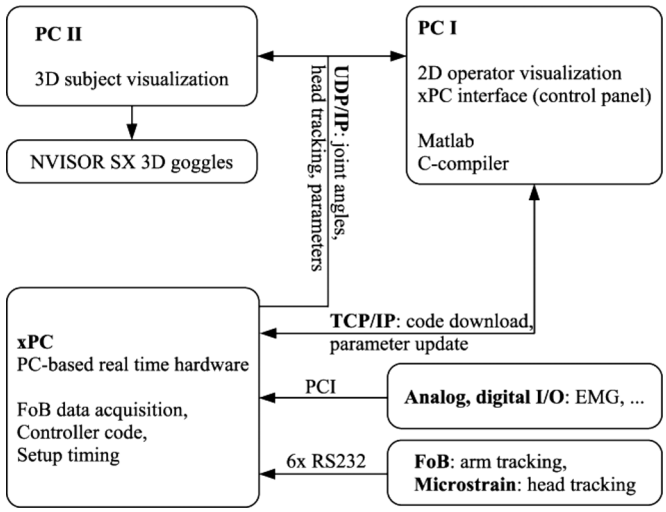


Fig. 5. Architecture of the VRE for real-time dynamic simulations with the patient in the loop. The VR models are simulated in xPC that is a regular PC with real-time operating system. xPC measures the command sources from the subject, calculates the resulting motion of the virtual limb, and sends it to PC II that displays a 3-D stereoscopic rendering of the limb motion to the subject from his/her point of view. PC I is used by the operator to develop VR models and control VR simulations.

PCs for animation of the model limb. Multiple PCs can be used to display the limb's movement to the subject, to the operator, and other users. For example, our configuration employs one PC for 3-D stereoscopic display to the subject from his/her perspective via a head-mounted display (nVisor SX, NVIS Inc.), and a second PC for 2-D display to the operator who can change his own view of the virtual world independently. The operator PC provides a user interface for experiment control, online parameter tuning, and development and downloading of code to the real-time PC.

Real-time simulations are generally difficult, especially for detailed models of complex systems. The complexity of the model, the efficiency of the simulation software, and the available processing power will determine whether a model can be simulated in real-time or not. The simulation of the multibody dynamics is generally fast and is not expected to be a problem for neural prostheses with realistic DOF. It is, however, more challenging when the system includes detailed models of biological components such as muscles and proprioceptive sensors or prosthetic components such as electric motors. These new components add to the model complexity and the computational load and because they generally have faster dynamics than the limb itself, the combined system can be characterized as stiff for numerical integration. To numerically integrate such stiff dynamic simulation problems, one can use explicit integration methods such as Runge–Kutta. But depending on how stiff the problem is, the required integration time step for accurate simulations might be too small to be achievable in a single real-time PC. As a remedy to this problem, MSMS provides multiple simulation algorithms for each model component such as a muscle or electric motor. This enables the users to tradeoff simulation accuracy for simulation speed whenever the computational power is limited and simpler models are acceptable for a given application. Another solution for stiff problems is the use

of implicit integration methods. These integration methods require more computation for each integration time step but allow the use of larger integration time steps. For example, we have simulated a 10 DOF prosthetic arm with ten electric motors (see Fig. 4) in a 3.6 GHz single processor PC. When we used accurate models of electric motors, the system was too stiff and the required integration time step for explicit integrators was too small and, therefore, impractical. But by the use of an implicit integrator (ode14x in Simulink) it was possible to simulate this model at a rate of about 100 Hz. We could also use an explicit integrator (fourth-order Runge–Kutta) that simulated the model at a rate of about 1000 Hz but we had to use simpler electric motor models.

E. Current Application Areas

The iterative approach for software development has enabled us to frequently release working versions of MSMS before its development is completed. These internal releases are applied to two applications in FES control of reach and grasp and neural control of prosthetic arm that provided us with valuable feedback to improve the design of MSMS before its final release. These two applications fit within the general framework of Fig. 1 and are implemented using the hardware setup shown in Fig. 5. They, however, differ in the MSMS models and the hardware interface as described in the following.

FES Control of Reach and Grasp: In our laboratory, we are investigating the control of reach and grasp in quadriplegic patients. These patients usually retain voluntary control of their upper arm but are paralyzed in their lower arm. To restore normal reach and grasp movements to these patients, we are using the residual movement of the upper arm to voluntarily drive the FES control of the paralyzed lower arm movement. One important question is whether the remaining residual movements are adequate for the control of paralyzed joints and whether the patient can learn to use them to effectively operate the FES system. To investigate these questions, we have used MSMS to build a model of the human arm and the task environment that are simulated in VRE (similar to the model in Fig. 3 minus the prosthetic left arm). In VRE, the patient voluntarily moves his upper arm. These movements are measured by the FOB motion capture system and sent to real-time PC via serial bus. The real-time PC then executes the control algorithm to produce the control signals for the paralyzed lower arm. In this example, controllers use the shoulder joint movement to predict the lower arm movement that would have been produced if the arm was intact. These calculations are done by a neural network model that has been trained on normal reaching movements [32]. The movement of the arm is then sent to MSMS in visualization PC to animate and display it to the subject from his/her point of view. The subject, therefore, can see the effect of his voluntary control actions on the arm movement and can learn by practice to produce the appropriate commands to perform the task successfully. In our preliminary studies, subjects learned to perform the VRE tasks faster by practice showing that VRE is an effective training environment for this application [33].

Neural Control of Prosthetic Arm: MSMS and the VRE described here are currently used by several teams around the country as part of a large project sponsored by Defense Advanced Research Projects Agency (DARPA) to develop new prosthetic arms for amputees. These teams are using VRE to investigate innovative prosthetic control strategies. Voluntary neural signals such as EMG from residual muscles, neural signals from the residual peripheral nerves or cerebral cortex are measured by different types of electrodes. The measured neural signals are then streamed into the real-time PC where they are processed to extract control signals to operate the prosthetic arm. These control signals are then fed to a physics-based simulation of the prosthetic arm to predict the movement of the prosthetic arm. The simulated movement of the prosthetic arm is then sent to the MSMS in visualization PC to animate and display it to the patient from his/her point of view (see Fig. 4). The visual and haptic feedback helps the patient learn to produce appropriate voluntary commands to successfully perform the reaching and grasp tasks in VRE. Currently, the MSMS models and VRE setup for this application have been developed and tested with able-bodied subjects producing EMG signals and prerecorded neural signals. The system is currently being installed in two laboratories where amputees (generating EMG signals) and monkeys (generating cortical neural signals) will control the movement of prosthetic limbs in VRE.

V. Conclusion

Neural prosthetic limbs plus the human patients operating them form very complex systems whose behaviors are difficult to intuit and, therefore, not amenable to *ad hoc* design and clinical fitting. We have developed new software known as MSMS that can simulate the behavior of such complex systems in their entirety. MSMS allows users to build accurate models of human and prosthetic limbs or a combination of them in the same environment. The models then can be exported to Simulink where physics-based simulations can be performed to predict the behavior of the system under various movement control strategies and external conditions. Accurate simulations including all of the known system parameters and elements can be performed to narrow the choices for the control strategy, optimize the control parameters, and analyze the sensitivity to control and system parameters. The same models or their simpler versions can be simulated in real-time with the patient-in-the-loop within the VRE where the patients can evaluate and learn to operate candidate neural prostheses before actually receiving them. As a general framework, MSMS can be used to simulate any neural prosthetic system. Currently, we are using MSMS for design and evaluation of FES control systems for reach and grasp and prosthetic control systems for amputees.

Acknowledgment

The authors would like to thank the MSMS project advisors Behzad Dariush, Brian Garner, Emanuel Todorov, Francisco Valero-Cuevas, Gary Yamaguchi, Ian Brown, Marcus Pandy, Robert Kirsch, Scott Delp, Stefan Schaal, and Victor Ng-Thow-Hing for their enthusiastic support and guidance.

References

[1] T. Bajd, A. Kralj, J. Sega, R. Turk, H. Benko, and P. Strojnik, "Use of a two-channel functional electrical stimulator to stand paraplegic patients," *Phys. Ther.*, vol. 61, pp. 526–527, 1981.

[2] W. T. Liberson, H. J. Holmquest, D. Scot, and M. Dow, "Functional electrotherapy: Stimulation of the peroneal nerve synchronized with the swing phase of the gait of hemiplegic patients," *Arch. Phys. Med. Rehab.*, vol. 42, pp. 101–105, 1961.
[3] P. H. Peckham, E. B. Marsolais, and J. T. Mortimer, "Restoration of key grip and release in the C6 tetraplegic patient through functional electrical stimulation," *J. Hand Surg. [Am.]*, vol. 5, no. 5, pp. 462–469, Sep. 1980.
[4] J. S. Petrofsky, C. A. Phillips, and D. E. Stafford, "Closed loop control for restoration of movement in paralyzed muscle," *Orthopedics*, vol. 7, pp. 1289–1302, 1984.
[5] M. M. Adamczyk and P. E. Crago, "Simulated feedforward neural network coordination of hand grasp and wrist angle in a neuroprosthesis," *IEEE Trans. Rehab. Eng.*, vol. 8, no. 3, pp. 297–304, Sep. 2000.
[6] R. Davoodi and B. J. Andrews, "Computer simulation of FES standing up in paraplegia: A self-adaptive fuzzy controller with reinforcement learning," *IEEE Trans. Rehab. Eng.*, vol. 6, no. 2, pp. 151–161, Jun. 1998.
[7] G. Khang and F. E. Zajac, "Paraplegic standing controlled by functional neuromuscular stimulation: Part II—Computer simulation studies," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 9, pp. 885–894, Sep. 1989.
[8] G. Khang and F. E. Zajac, "Paraplegic standing controlled by functional neuromuscular stimulation: Part I—Computer model and control-system design," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 9, pp. 873–884, Sep. 1989.
[9] R. F. Kirsch, A. M. Acosta, F. C. T. van der Helm, R. J. J. Rotteveel, and L. A. Cash, "Model-based development of neuroprostheses for restoring proximal arm function," *J. Rehab. Res. Develop.*, vol. 38, no. 6, pp. 619–626, Nov. 2001.
[10] D. B. Popovic, R. B. Stein, M. N. Oguztoreli, M. K. Lebiedowska, and S. Jonic, "Optimal control of walking with functional electrical stimulation: A computer simulation study," *IEEE Trans. Rehab. Eng.*, vol. 7, no. 1, pp. 69–79, Jan. 1999.
[11] R. Riener and T. Fuhr, "Patient-driven control of FES-supported standing up: A simulation study," *IEEE Trans. Rehab. Eng.*, vol. 6, no. 2, pp. 113–124, Feb. 1998.
[12] A. Soares, A. Andrade, E. Lamounier, and R. Carrijo, "The development of a virtual myoelectric prosthesis controlled by an EMG pattern recognition system based on neural networks," *J. Intell. Inf. Syst.*, vol. 21, no. 2, pp. 127–141, 2003.
[13] R. Davoodi and B. J. Andrews, "Optimal control of FES-assisted standing up in paraplegia using genetic algorithms," *Med. Eng. Phys.*, vol. 21, pp. 609–617, 1999.
[14] K. J. Hunt, M. Munih, and N. Donaldson, "Feedback control of unsupported standing in paraplegia-Part I: Optimal control approach," *IEEE Trans. Rehab. Eng.*, vol. 5, no. 4, pp. 331–340, Dec. 1997.
[15] M. Munih, N. Donaldson, K. J. Hunt, and M. D. Fiona, "Feedback control of unsupported standing in paraplegia-part II: Experimental results," *IEEE Trans. Rehab. Eng.*, vol. 5, no. 4, pp. 341–352, Dec. 1997.
[16] J. J. Abbas and R. J. Triolo, "Experimental evaluation of an adaptive feedforward controller for use in functional neuromuscular stimulation systems," *IEEE Trans. Rehab. Eng.*, vol. 5, no. 1, pp. 12–22, Mar. 1997.
[17] P. E. Crago, J. T. Mortimer, and P. H. Peckham, "Closed-loop control of force during electrical stimulation of muscle," *IEEE Trans. Biomed. Eng.*, vol. 27, no. 6, pp. 306–312, Jun. 1980.
[18] B. N. Ezenwa, R. M. Glaser, W. Couch, S. F. Figoni, and M. M. Rodgers, "Adaptive control of functional neuromuscular stimulation-induced knee extension exercise," *J. Rehab. Res. Develop.*, vol. 28, no. 4, pp. 1–8, 1991.
[19] M. S. Hatwell, B. J. Oderkerk, C. A. Sacher, and G. F. Inbar, "The development of a model reference adaptive controller to control the knee joint of paraplegics," *IEEE Trans. Autom. Control*, vol. 36, no. 6, pp. 683–691, Jun. 1991.
[20] J. P. Donoghue, "Connecting cortex to machines: Recent advances in brain interfaces," *Nature Neurosci.*, vol. 5, pp. 1085–1088, Nov. 2002.
[21] A. B. Schwartz, "Cortical neural prosthetics," *Annu. Rev. Neurosci.*, vol. 27, pp. 487–507, 2004.
[22] M. A. Lemay and P. E. Crago, "A dynamic model for simulating movements of the elbow, forearm, an wrist," *J. Biomech.*, vol. 29, no. 10, pp. 1319–1330, Oct. 1996.
[23] K. G. Gerritsen, A. J. van den Bogert, M. Hulliger, and R. F. Zernicke, "Intrinsic muscle properties facilitate locomotor control—A computer simulation study," *Motor Control*, vol. 2, no. 3, pp. 206–220, Jul. 1998.
[24] G. E. Loeb, I. E. Brown, and E. J. Cheng, "A hierarchical foundation for models of sensorimotor control," *Exp. Brain Res.*, vol. 126, no. 1, pp. 1–18, May 1999.
[25] S. L. Delp and J. P. Loan, "A graphics-based software system to develop and analyze models of musculoskeletal structures," *Comput. Biol. Med.*, vol. 25, no. 1, pp. 21–34, Jan. 1995.
[26] S. L. Delp and J. P. Loan, "A computational framework for simulating and analyzing human and animal movement," *Comput. Sci. Eng.*, vol. 2, no. 5, pp. 46–55, 2000.
[27] R. Davoodi and G. E. Loeb, "A software tool for faster development of complex models of musculoskeletal systems and sensorimotor controllers in simulink," *J. Appl. Biomech.*, vol. 18, pp. 357–365, 2002.
[28] R. Davoodi, I. E. Brown, and G. E. Loeb, "Advanced modeling environment for developing and testing FES control systems," *Med. Eng. Phys.*, vol. 25, no. 1, pp. 3–9, Jan. 2003.
[29] B. A. Garner and M. G. Pandy, "Estimation of musculotendon properties in the human upper limb," *Annals Biomed. Eng.*, vol. 31, no. 2, pp. 207–220, Feb. 2003.
[30] B. A. Garner and M. G. Pandy, "The obstacle-set method for representing muscle paths in musculoskeletal models," *Comput. Methods Biomech. Biomed. Eng.*, vol. 3, no. 1, pp. 1–30, 2000.
[31] M. Hauschild, R. Davoodi, and G. E. Loeb, "A virtual reality environment for designing and fitting neural prosthetic limbs," *IEEE Trans. Neural Syst. Rehab. Eng.*, vol. 15, no. 1, pp. 9–15, Mar. 2007.
[32] R. Kaliki, R. Davoodi, and G. E. Loeb, "The effect of training set on prediction of elbow trajectory from shoulder trajectory during reaching to targets," in *Proc. 28th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2006, pp. 5483–5486.
[33] R. Davoodi, M. Hauschild, J. Lee, P. T. Montazemi, and G. E. Loeb, "Biomimetic control of FES reaching," presented at the 1st Int. Conf. Neural Interface Control, Wuhan, China, 2005.

Rahman Davoodi received the B.S. degree in mechanical engineering and the M.Sc. degree in biomechanical engineering from Sharif University of Technology, Tehran, Iran, and the Ph.D. degree in biomedical engineering from University of Alberta, Edmonton, AB, Canada.

He is currently a Research Assistant Professor with the Department of Biomedical Engineering, University of Southern California, Los Angeles. His current research is focused on the use of functional electrical stimulation (FES) to restore activities of normal daily living to the paralyzed and amputee patients such as standing, reaching, grasping, and exercise. He has used machine learning techniques to coordinate man-machine interactions in these systems. He has also developed several software tools for modeling, simulation, and virtual prototyping of complex neural prostheses for paralyzed and amputee patients.

Chet Urata received the B.A. degree in computer science from the University of California, Berkeley, in 1984.

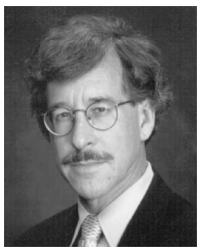
He currently runs the Urata Corporation, a software development company based in Waianae, HI. He has almost 25 years of programming experience in the fields of biomedical, defense, general business, and the internet.

Markus Hauschild received the Diploma in mechatronics from Munich University of Applied Sciences, Munich, Germany, in 2002. Currently, he is pursuing the Ph.D. degree in biomedical engineering from the University of Southern California, Los Angeles.

In 2001, he joined the DLR Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany, where he developed control strategies for harmonic drive gears. In 2003, he was a Visiting Researcher with the National Yunlin University of Science and Technology, Taiwan, R.O.C., where he developed iterative learning control for compensation of periodic disturbances. His research interests include human-machine-interfaces, motor control, neural interfaces, and prosthetics.

Mehdi Khachani received the B.Eng. degree in electrical engineering from Ecole Polytechnique de Monteal, Montreal, QC, Canada, and the M.Eng. degree in biomedical engineering from McGill University, Montreal, QC, Canada.

He is a Biomedical Engineer with the Modeling and Control Group, the Alfred Mann Institute, University of Southern California, Los Angeles. His main work is directed towards developing the musculoskeletal modeling software and the virtual reality environment.



Gerald E. Loeb (M'98) received the B.A. and M.D. degrees from Johns Hopkins University, Baltimore, MD, in 1969 and 1972.

Currently, he is a Professor with the Department of Biomedical Engineering and Neurology and a Director of the Medical Device Development Facility of the A. E. Mann Institute for Biomedical Engineering, the University of Southern California, Los Angeles. He completed one year of surgical residency with the University of Arizona, Tucson, before joining the Laboratory of Neural Control at the National Institutes of Health from 1973 to 1988. From 1988 to 1999, he was a Professor with the Department of Physiology and Biomedical Engineering, Queen's University, Kingston, ON, Canada. He was one of the original developers of the cochlear implant to restore hearing to the deaf and from 1994 to 1999, he was a Chief Scientist for Advanced Bionics Corporation, Sylmar, CA, manufacturers of the Clarion cochlear implant. He holds 43 issued U.S. patents and is the author of over 200 scientific papers. His current research is directed toward neural prosthetics to reanimate paralyzed muscles and limbs using a new technology that he and his collaborators developed called BIONs. This work is supported by an NIH Bioengineering Research Partnership and is one of the testbeds in the National Science Foundation Engineering Research Center on Biomimetic MicroElectronic Systems, for which he is deputy director. These clinical applications build on his long-standing basic research into the properties and natural activities of muscles, motoneurons, proprioceptors, and spinal reflexes.

Dr. Loeb is a Fellow of the American Institute of Medical and Biological Engineers.