INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN BIOMEDICAL ENGINEERING
Int. J. Numer. Meth. Biomed. Engng. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cnm.2486

Neuromechanic: A computational platform for simulation and analysis of the neural control of movement

Nathan E. Bunderson [1,*,†], Jeffrey T. Bingham [2,3], M. Hongchul Sohn [3], Lena H. Ting [1,2,3,4] and Thomas J. Burkholder [1,2]

[1] *School of Applied Physiology, Georgia Institute of Technology, Atlanta, GA, U.S.A.*
[2] *Interdisciplinary Bioengineering Program, Georgia Institute of Technology, Atlanta, GA, U.S.A.*
[3] *School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, U.S.A.*
[4] *Department of Biomedical Engineering, Emory University and Georgia Institute of Technology Atlanta, GA, U.S.A.*

SUMMARY

Neuromusculoskeletal models solve the basic problem of determining how the body moves under the influence of external and internal forces. Existing biomechanical modeling programs often emphasize dynamics with the goal of finding a feed-forward neural program to replicate experimental data or of estimating force contributions or individual muscles. The computation of rigid-body dynamics, muscle forces, and activation of the muscles are often performed separately. We have developed an intrinsically forward computational platform (Neuromechanic, www.neuromechanic.com) that explicitly represents the interdependencies among rigid body dynamics, frictional contact, muscle mechanics, and neural control modules. This formulation has significant advantages for optimization and forward simulation, particularly with application to neural controllers with feedback or regulatory features. Explicit inclusion of all state dependencies allows calculation of system derivatives with respect to kinematic states and muscle and neural control states, thus affording a wealth of analytical tools, including linearization, stability analyses and calculation of initial conditions for forward simulations. In this review, we describe our algorithm for generating state equations and explain how they may be used in integration, linearization, and stability analysis tools to provide structural insights into the neural control of movement.

Received 15 December 2011; Revised 8 March 2012; Accepted 31 March 2012

KEY WORDS: forward simulation; linearization; stability; biomechanics; opensim; dynamics engine

1. INTRODUCTION

1.1. Movements result from complex neuromuscular interactions

The relative contributions of multiple neural and biomechanical mechanisms to smooth, graceful, and efficient biological movements are difficult to dissociate intuitively. For example, the response to a postural perturbation involves interaction among several systems. Imagine being shoved from behind: at the physical level, this will cause some flexion along the spine and at the hip, extension of the knees and flexion of the ankles, and motion of the arms. In the absence of any neuromuscular response, the distribution of joint motions depends on the inertia and anatomy of the body. In the absence of any change in neural drive, muscular forces still vary with changing muscle lengths because of joint motion and the forces will affect that motion. The nervous system makes its own multifaceted contribution, drawing information from muscle proprioceptors, vestibular, and visual sensors. Weighing the relative importance of these systems requires careful accounting for each

*Correspondence to: Nathan E. Bunderson, School of Applied Physiology, Georgia Institute of Technology, 55514th Street, Atlanta, GA 30332-0356 U.S .A.
†E-mail: nbunderson@gatech.edu

of their unique properties and their integration through the body. Without integrative and quantitative models to test these theories, controversies persist in the literature. For example, intrinsic muscle mechanics — passive viscoelasticity and cross bridge mechanics — provide an instantaneous force change that counters a perturbation and contributes to postural stability [1], but these intrinsic properties do not appear to be sufficient to maintain upright posture by themselves [2, 3]. Likewise, spinal reflexes reinforce and amplify muscular stability and could mediate balance [3, 4], but the global behavior of humans during stance is poorly described by these reactive mechanisms alone [5–7]. Finally, active attention may or may not be required for robust balance control [8]. It is experimentally difficult to separate the passive physics of the body, the intrinsic mechanical properties of the muscles, and the control processes of the nervous system because a change in any one component ripples through the entire system.

Understanding the principles that drive complex and redundant neuromuscular interactions and strategies requires computational modeling techniques that allow the contributions of different components to be varied and evaluated, particularly for motor behaviors that are inherently unstable, such as walking and standing balance control. Inclusion of postural configuration and appropriate muscle properties can significantly alter results of musculoskeletal simulations [9], and effectively stabilize them [10]. In the absence of the appropriate intrinsic stability provided by muscular contraction and feedback control, small changes in the pattern of muscle activation can completely destabilize a simulation [11]. Such sensitivity to small perturbation is not physiological, because our body dynamics are resistant to small perturbations because of viscoelastic properties not modeled in rigid body dynamics. Nonphysiological instabilities make it difficult to generate *de novo* forward simulations based on optimizing task goals [12] or an assumed neural structure, because of these inherent instabilities. We still lack an appropriate computational platform for understanding the neural strategies for muscle activation that can be modulated to generate a range of different movements.

1.2. Solution strategies

All musculoskeletal models solve the same basic problem of determining how the body moves under the influence of external and internal forces according to laws of physics. Simplifying and quantifying the complex skeletal, muscular, and neural components that affect these forces is the first challenge of neuromusculoskeletal modeling.

The dominant strategy for musculoskeletal simulation and analysis is to separate the components with an inverse dynamics approach. In this strategy it is axiomatic to divide the body into rigid (i.e., nondeformable fixed inertial parameters) segments linked by mechanical constraints representing the action of connective tissue at the physiological joints of the body. This is partly motivated by well-developed formalism for modeling systems of rigid links and by easy access to many excellent rigid body dynamics engines. These engines reduce the description of motion of the rigid body system to

$$\ddot{\vec{q}} = f\left(\vec{q}, \dot{\vec{q}}, \vec{T}_{\mathrm{M}}\right), \tag{1}$$

where $\vec{q}$ is a vector defining the posture of the system of rigid bodies and $\vec{T}_{\mathrm{M}}$ is a vector of generalized internal forces arising primarily from muscle forces.

The strategy of separating components of the neuromusculoskeletal system is extremely well suited to the analysis or reproduction of experimentally-measured kinetic and kinematic data. Given a set of experimentally measured kinematics $\left(\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}}\right)$ and external forces, Equation (1) can be inverted to yield internal generalized forces $\vec{T}_{\mathrm{M}}$ [13–16].

Muscle forces $\left(\vec{F}_{\mathrm{M}}\right)$ dominate the internal generalized forces and can be derived, along with muscle excitation $\left(\vec{\varepsilon}\right)$, from the internal generalized forces based on various optimization criteria or measured EMG [17–19]. It is nearly axiomatic to model each muscle as a completely independent

DOI: 10.1002/cnm

force generator attached to the rigid segments at finite, discrete points. The widespread acceptance of Hill-type muscle models yields a relationship between kinematics and muscle force,

$$\vec{F}_{\mathrm{M}} = f\left(\vec{q}, \dot{\vec{q}}, \vec{\varepsilon}\right), \tag{2}$$

which allows inference of the neural drive required to produce the predicted pattern of muscle forces. The neural model is often embodied in the cost function (e.g., minimum effort) used to distribute the generalized forces among the various muscles.

Although modeling the rigid body dynamics separate from muscle mechanics has been an invaluable tool for analyzing biomechanical experiments, such a strategy masks the complex interdependencies of the integrated neuromechanical system. Reconstruction of an experimentally observed motion yields a feedforward pattern of activation, but does not elucidate how the nervous system arrives at this pattern, nor does it reveal the continuous sensory cues and feedback transformations that are used to maintain it in the face of continual system and external noise. Because the practice of separating skeletal, muscle, and neural dynamics grew from inverse dynamics, it carries an implicitly inverse paradigm and is limited in its ability to predict motor behavior based on hypotheses about the neural control system.

We have developed an intrinsically forward computational platform (Neuromechanic) that explicitly represents the interdependencies among rigid body dynamics, frictional contact, muscle mechanics, and neural control modules. On this platform the dynamics have been formulated to include states describing rigid body dynamics $\left(\vec{q}, \dot{\vec{q}}\right)$, neural dynamics $\left(\vec{n}\right)$, and muscle dynamics $\left(\vec{a}, \vec{l}\right)$.

$$\begin{bmatrix} \dot{\vec{q}} \\ \ddot{\vec{q}} \\ \dot{\vec{n}} \\ \dot{\vec{a}} \\ \dot{\vec{l}} \end{bmatrix} = f\left(\vec{q}, \dot{\vec{q}}, \vec{n}, \vec{a}, \vec{l}, t\right). \tag{3}$$

Although the difference seems subtle, the strategy has significant advantages for optimization and forward simulation, particularly if one's principal interest is neural control. Explicit inclusion of all state dependencies allows calculation of system derivatives and affords a wealth of analytical tools to examine both kinematic and muscle states, including linearization, stability analyses (useful for optimization) and calculation of initial conditions for forward simulations. In the sections that follow, we will describe our algorithm for generating all state derivatives and explain how they may be used in integration, linearization, and stability analysis tools to provide insights into the neural control of movement.

2. PROGRAM DESIGN

At the center of Neuromechanic is a method for the calculation of integrated neuromechanical state equations governing the dynamics of motion. Given the state of the system at a particular instance of time, the rate of change of that state is calculated and integrated forward in time yielding the state at the next time step (forward simulation) generating realistic descriptions of movement, muscle mechanics, and neural dynamics. Forward simulation may reveal many details about the neuromechanical coupling across kinematic, muscle, and neural states that are difficult or impossible to obtain through experimental observation or inverse dynamics. Figure 1 illustrates the order in which the equations of state are generated by Neuromechanic at each time step and the interdependencies among the elements. The benefit of the Neuromechanic formalism is that these state equations are

DOI: 10.1002/cnm

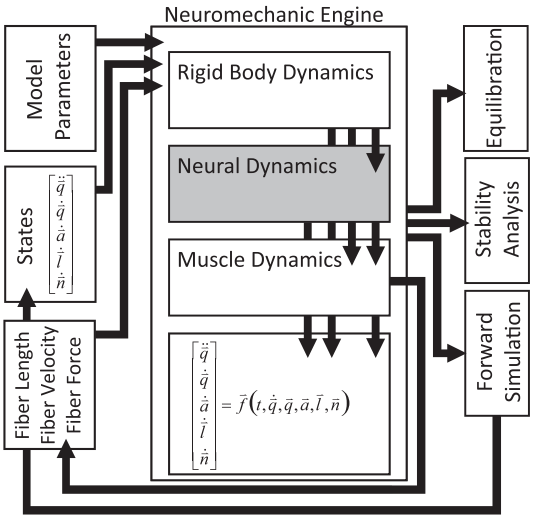

Figure 1. The schematic demonstrates the order in which state equations are generated by Neuromechanic at each time step and the interdependencies among the elements. The inputs at any time step are the constant model parameters, the kinematic, muscle, and neural states, and muscle related state dependent variables (i.e., muscle fiber force, muscle fiber length, muscle fiber velocity) from the previous iteration. Neuromechanic first calculates all variables (e.g., system inertia, gravitational forces, moment arm matrices) dependent only on rigid body states. These variables are then used along with the muscle related state dependent variables from the previous iteration to calculate muscle excitation values and neural state derivatives. Muscle forces, muscle state derivatives, and other muscle related state-dependent variables from rigid body and neural variables. Finally, the frictional contact forces and rigid body state derivatives are computed.

available to tools that enhance analysis and simplify the forward simulation process. An equilibration tool is used to set the many initial conditions to viable and realistic values. A stability analysis tool allows a prediction of the behavior of the model, even before the simulation begins, given these initial conditions. A linearization tool can be used to design optimal neural controllers. In addition, the program is extensible with neural control modules for modeling and testing a wide variety of neural control hypotheses. The methods used to construct the state equations and their applications are discussed in this section.

2.1. Constructing state equations

2.1.1. Skeletal dynamics. The inertial and skeletal dynamics are modeled using well-established rigid body dynamics methods. Although a remarkable diversity exists in the development of algorithms for numerically simulating rigid multibody dynamics, all methods ultimately result in identical system dynamics. The different algorithms balance the ease of conceptually representing a dynamic system with the speed, accuracy, and stability of the resulting differential equations that must be solved [20–22]. The dynamics engine behind Neuromechanic is based on the common simplification that biological systems can be represented as open-chain, rigid linkages. This slight compromise in generality is balanced by a simpler representation of the dynamics, increased speed, and more stable numerical simulation. Each rigid body is assigned translational and/or rotational degrees of freedom relative to a parent body resulting in a minimum number of dependent coordinates. This coordinate scheme also lends itself well to an anatomical description of a biomechanical model where the joint angles are defined relative to their adjacent body parts as opposed to some abstract reference. It also allows implicit embedding of kinematic constraints into the formulation of the equations of motion and removes the need for additional constraint equations. The concatenated position and velocity of these coordinates comprise the rigid body state vectors $\left(\vec{q},\dot{\vec{q}}\right)$ and the equations of motion for these states are developed from a Lagrangian formulation of D'Alembert's principle similar to the algorithm proposed by Kane and Levinson [23]. This particular algorithm is

DOI: 10.1002/cnm

well suited to open-chain linkages with a tree structure and allows rapid recursive computation of kinematic and dynamic variables from local coordinates. It is important to note that the equation of motion in Neuromechanic is fundamentally the same as in the separate physics solution

$$\mathbf{M}\ddot{\vec{q}}_r = \vec{V} + \vec{T}_{\mathrm{E}} + \vec{T}_{\mathrm{M}}. \tag{4}$$

The system inertia matrix $\mathbf{M}$ is dependent on the systems current configuration $\left(\vec{q}\right)$. The vector of generalized Coriolis and centripetal forces $\vec{V}$ and vector of externally imposed generalized forces $\vec{T}_{\mathrm{E}}$ are dependent on both the configuration $\left(\vec{q}\right)$ and joint velocities $\left(\dot{\vec{q}}\right)$. The vector of internal generalized forces $\vec{T}_{\mathrm{M}}$ is entirely due to the force generated in muscles, which will be described later. Here, it is sufficient to say that they are dependent on rigid body, muscle, and neural states.

The vector of externally imposed generalized forces is induced by a combination of externally applied forces and moments (wrenches) that depend on the environment. In Neuromechanic these are separated into those induced by gravity $\left(\vec{G}\right)$, frictional contact $\left(\vec{P}\right)$, other imposed external wrenches $\left(\mathbf{J}_{\mathrm{E}}^T\,\vec{W}_{\mathrm{E}}\right)$, and imposed, external kinematic constraints $\left(\mathbf{J}_{\mathrm{C}}^T\,\vec{W}_{\mathrm{C}}\right)$. The external wrenches influence the rigid body dynamics through the product of the external wrench $\left(\vec{W}\right)$ and endpoint Jacobian at the point of application ($\mathbf{J}$).

$$\vec{T}_{\mathrm{E}} = \vec{G} + \vec{P} + \mathbf{J}_{\mathrm{E}}^T\,\vec{W}_{\mathrm{E}} + \mathbf{J}_{\mathrm{C}}^T\,\vec{W}_{\mathrm{C}}. \tag{5}$$

The tree structure and recursive formulation employed by Neuromechanic has the benefit that the system inertia and generalized Coriolis and centripetal forces are efficiently calculated. Moreover, the formulation of these components results in extremely efficient calculation of arbitrary endpoint Jacobians and external forces can be defined either in terms of relative or inertial frame coordinates.

In the case of kinematic constraints it is the kinematics, rather than the wrenches that arise from them, which are prescribed. Holonomic acceleration constraints ($\ddot{x}_c$) are prescribed and the induced wrenches augment system of linear equations of motion as Lagrange multipliers [22].

$$\begin{aligned} \mathbf{M}\ddot{\vec{q}}_r &= \vec{V} + \vec{G} + \mathbf{J}_{\mathrm{E}}^T\,\vec{W}_{\mathrm{E}} + \mathbf{J}_{\mathrm{C}}^T\,\vec{W}_{\mathrm{C}} + \vec{T}_{\mathrm{M}} \\ \ddot{x}_c &= \mathbf{J}_{\mathrm{C}}\ddot{\vec{q}}_r + \dot{\mathbf{J}}_{\mathrm{C}}\dot{\vec{q}}_r \end{aligned} \tag{6}$$

$$\begin{bmatrix} \mathbf{M} & -\mathbf{J}_{\mathrm{C}}^T \\ -\mathbf{J}_{\mathrm{C}} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\vec{q}}_r \\ \vec{W}_{\mathrm{C}} \end{bmatrix} = \begin{bmatrix} \vec{V} + \vec{G} + \mathbf{J}_{\mathrm{E}}^T\,\vec{W}_{\mathrm{E}} + \vec{T}_{\mathrm{M}} \\ \dot{\mathbf{J}}_{\mathrm{C}}\dot{\vec{q}}_r - \ddot{\vec{x}}_{\mathrm{C}} \end{bmatrix} \tag{7}$$

Inelastic impact and frictional contact forces are obtained from a velocity-based frictional contact model [24], which has been adapted to work with acceleration-based dynamics. In the current implementation of Neuromechanic, frictional contact constraints cannot be combined with holonomic acceleration constraints. The frictional contact forces are calculated each time step after all other contributions, including internally imposed generalized forces, are computed.

2.1.2. Neural control. The internal generalized forces $\left(\vec{T}_{\mathrm{M}}\right)$ depend on the rigid body states and a neural controller and muscle dynamics. In the Neuromechanic workflow, the neural control (including muscle excitation functions) is calculated after the skeletal and inertial dynamics (consisting of variables that depend only on rigid-body states), but before muscle dynamics at each time step.

DOI: 10.1002/cnm

Neuromechanic supports the implementation of neural dynamics with an arbitrary number of neural states. The simplest, zero-state feedforward and/or instantaneous feedback control are implemented in the form of a neural network. Each 'neuron' transforms inputs (sx_i) derived from kinematic $\left(\vec{q}, \dot{\vec{q}}\right)$ and muscle (a, l) states to neural outputs (o),

$$o = f_N\left(\sum_{i=1}^{nsyn} sp_{i1}(sx_i - sp_{i2})\right), \tag{8}$$

Specifically, the inputs may be generalized coordinate positions and velocities, muscle force, fiber length and velocity, musculotendon length and velocity, the scalar output of another neuron, or time-dependent cubic-spline functions. The inputs sum linearly with a gain (sp_{i1}) and offset (sp_{i2}) applied and may be passed through sigmoidal, hyperbolic tangent, or Heaviside shaping functions

$$\begin{aligned} f_N(x) &= (1 + e^{-x})^{-1} \\ f_N(x) &= \tanh(x) \\ f_N(x) &= lb + (ub - lb)H(x) \end{aligned}, \tag{9}$$

where lb and ub are lower and upper bounds and $H(x)$ is the Heaviside step function. Alternatively, the output may be saturated at ub for values greater than ub and at lb for values less than lb. Because the output of one neuron may be an input for other neurons, these can be strung together in layers after the fashion of a neural network.

More advanced and specific neural controllers are implemented in external control modules. The control modules can implement their own neural equations of state,

$$\dot{\vec{n}} = f(\ldots), \tag{10}$$

which are incorporated into the state vector (Equation (3)) for simulation and analysis. For example, a center of mass controller, included in the software distribution, adds six states representing the integral of linear and angular momentum of the animal,

$$\dot{\vec{n}}_{\mathrm{M}} = L\dot{\vec{q}}. \tag{11}$$

The matrix L relates the generalized coordinate velocities with the vector of linear and angular momentum $\left(\dot{\vec{n}}_{\mathrm{M}}\right)$ and depends on the generalized coordinate positions. It is calculated by the Neuromechanic master program and passed into the neural control module. An API is available for creation of controllers based on any of Neuromechanic's states and many state-derived variables.

As previously stated the output of any neuron may serve as an input for other neurons. In addition the neuron output may be the excitation (ε) to a muscle. These 'motoneurons' provide the direct connection from the neural controller to the muscle dynamics. One of the most confounding interdependencies of neuromusculoskeletal modeling is that the muscle excitation functions may depend nonlinearly on muscle variables and the muscle variables may depend nonlinearly on the muscle excitation function. These functions may combine to form a transcendental function, which must typically be solved iteratively. In the current implementation of Neuromechanic this applies to three muscle variables (muscle fiber length, muscle fiber velocity, and muscle force) and neural excitation. To circumvent this computationally-expensive step, Neuromechanic assumes that changes in these muscle variables are small and uses the stored values from the previous time-step to compute the excitation function. As long as time steps are appropriately small, errors introduced from this method should be negligible.

2.1.3. Muscle dynamics. Neuromechanic uses Hill muscle models [25, 26] to estimate muscle forces from rigid body states, neural activation, and internal muscle states that capture series elasticity and activation dynamics. In addition to providing the musculotendon force (F_{M}), the

DOI: 10.1002/cnm

muscle models provide state equations for activation level (a_i) and contractile element length (l_i) for muscle i,

$$\begin{bmatrix} \dot{a}_i \\ \dot{l}_i \end{bmatrix} = f\left(l_i, a_i, \vec{q}, \varepsilon_i\right), \tag{12}$$

where ε_i is the excitation from a neuron. In addition the muscle force may be directly proportional to the muscle excitation

$$F_{M,i} = F_{MAXi}\varepsilon_i, \tag{13}$$

where F_{MAXi} is a fixed model parameter. By using this proportional muscle model in combination with a neural control module, additional user-defined muscle models may be included with an arbitrary number of additional muscle states.

The internally induced generalized force vector $\left(\vec{T}_{\mathrm{M}}\right)$ is calculated from the muscle forces $\left(\vec{F}_{\mathrm{M}}\right)$ through the moment arm matrix (**R**), which is dependent only on fixed parameters (muscle attachment points and wrapping surfaces) and the configuration $\left(\vec{q}\right)$,

$$\vec{T}_{\mathrm{M}} = \mathbf{R}\vec{F}_{\mathrm{M}}. \tag{14}$$

2.2. Application of state equations

2.2.1. Forward simulation. As described previously, given the current configuration of a neuromusculoskeletal model (including fixed parameters such as mass and the current kinematic, muscle, and neural states) a single vector (Equation (3)) describing the rate of change of all dynamic states can be obtained at each time step. During forward simulation, the vector of state time derivatives is numerically integrated to obtain the state values at the next time step. Those states are, in turn, used to calculate the updated state derivatives. This process is repeated generating the trajectory for each state and many additional state-derived variables useful for understanding animal motion. Neuromechanic implements three numerical integrators: a fixed step fourth-order Runge–Kutta, a variable step fourth-order/fifth-order Runge–Kutta and a Runge–Kutta method of order 8 [27].

2.2.2. Equilibration. To obtain reasonable forward simulations, each state (kinematic, muscle and neural) must be set to an appropriate initial value, and finding these values can often require substantial offline calculations. Neuromechanic implements an equilibration tool that can be used to automatically set the muscle and neural states to equilibrium initial conditions based on a solution that minimizes a user-modifiable cost function. First, a set of muscle forces achieving kinematic equilibrium is obtained by solving

$$\mathbf{R}\vec{F}_{\mathrm{M}} = -\mathbf{J}_{\mathrm{E}}^{T}\,\vec{W}_{\mathrm{E}} - \vec{G} - \mathbf{J}_{\mathrm{C}}^{T}\,\vec{W}_{\mathrm{C}} - \vec{P} \tag{15}$$

subject to

$$\vec{F}_{\mathrm{MIN}} \leqslant \vec{F}_{\mathrm{M}} \leqslant \vec{F}_{\mathrm{MAX}}, \tag{16}$$

where $\left(\vec{F}_{\mathrm{MIN}}\right.$ and $\left.\vec{F}_{\mathrm{MAX}}\right)$ are the vectors of minimum and maximum allowable muscle force, respectively. Because there are often multiple solutions to this problem the tool chooses a unique solution by solving Equation (15) subject to the constraint in Equation (16) while minimizing a quadratic cost,

$$c = \frac{1}{2}\,\vec{F}_{\mathrm{M}}^{\,T}\mathbf{Q}\vec{F}_{\mathrm{M}} + \vec{C}^{\,T}\,\vec{F}_{\mathrm{M}}. \tag{17}$$

The quadratic (**Q**), and linear $\left(\vec{C}\right)$ terms can be set by the user based on the requirements of the task. Set to the identity matrix and zero vector respectively results in a cost function, c, which

DOI: 10.1002/cnm

describes the sum square of total muscle force. Setting the diagonal terms of $\mathbf{Q}$ to the square inverse of the maximum isometric force for each muscle and setting all other terms of $\mathbf{Q}$ and $\vec{C}$ to zero approximates the popular muscle stress [19] cost (see Illustration 2 below).

After solving for the equilibrium muscle forces, the neural excitations $\left(\vec{\varepsilon}\right)$ and compatible muscle states $\left(\vec{a}, \vec{l}\right)$ are found to satisfy the equilibrated muscle and neural dynamic equations,

$$\dot{n}_i = 0 = f(\ldots). \tag{18}$$

$$\dot{a}_i = 0 = f(a_i, \varepsilon_i), \tag{19}$$

$$\dot{l}_i = 0 = f\left(l_i, a_i, \vec{q}\right). \tag{20}$$

The result is a complete set of mechanical, neural, and muscle states that produce the desired interaction forces in a desired equilibrium posture, providing convenient initial conditions for many forward simulations.

2.2.3. Linearization and stability analysis. The integrated approach also makes it easy to generate a linearized approximation of the system dynamics for rapid approximation of neuromechanical responses, optimization, and Lyapunov stability analysis. The linearized system state matrix is a truncated Taylor series expansion that describes how a small change in the states will affect the system,

$$\dot{\vec{z}} = \begin{bmatrix} \dot{\vec{q}} \\ \ddot{\vec{q}} \\ \dot{a} \\ \dot{l} \\ \dot{\vec{n}} \end{bmatrix} = f\left(\vec{q}, \dot{\vec{q}}, \vec{a}, \vec{l}, \vec{n}, t\right) \tag{21}$$

$$\Delta\dot{\vec{z}} = \dot{\vec{z}} - \dot{\vec{z}}_0 = \sum_{m=1}^{\infty} \frac{\Delta\vec{z}^{\,m}}{m!} \left(\frac{\partial^m f}{\partial \vec{z}^{\,m}}\right)_{\vec{z}=\vec{z}_0} \cong \mathbf{A}\Delta\vec{z}$$
$$\text{where } \mathbf{A} = \left(\frac{\partial f}{\partial \vec{z}}\right)_{\vec{z}=\vec{z}_0} \tag{22}$$

Here, the full value of an integrated approach to generating system equations across kinematic, muscle and neural states can be seen. Neuromechanic generates the columns of the linearized state matrix $\mathbf{A}$ by slightly perturbing each element of the state vector to determine the sensitivity of the state derivatives to that state. Because all state dependencies are collected into a single routine, the perturbation is propagated through all state equations. This differentiation of the system state matrix includes all native states and additional states prescribed in the neural control modules. This facilitates linearization of the state equations even with the arbitrary user-defined neural controllers. The linearized state matrix is very useful in various optimization methods and useful in revealing the local stability of the system. If the state vector $\vec{z}_0$ is an equilibrium value, the eigenvalues of $\mathbf{A}$ reveal the first-order response to perturbations along the system modes (eigenvectors of $\mathbf{A}$). Eigenvalues with negative real parts indicate that the linearized system will be asymptotically stable, while positive eigenvalues indicate divergence toward infinity. Equally important, the linearized system matrix allows application of the whole wealth of linear control theory, subject to the caveat of small perturbations, including analytical solutions to optimization problems.

DOI: 10.1002/cnm

2.2.4. Extensibility. Neuromechanic emphasizes control of biomechanical models with the goal of evaluating interactions between neural control strategies and constraints imposed by the musculoskeletal system in the production of stable and robust movement. User-defined neural control modules may model sensory feedback, synaptic and transmission delays, and internal models of skeletal and muscle dynamics. Nearly all the information used to simulate musculoskeletal dynamics is available to these neural control modules. This includes additional and sometimes abstract variables such as center of mass dynamics and momentum variables, which are key to some hypothesized models of neural control. Moreover, the user-defined states of the neural control modules are directly incorporated into the global state vector making them available for the previously described integration, equilibration, and linearization tools.

3. ILLUSTRATIONS

3.1. Illustration 1: stability as a criteria for muscle activation pattern selection

We used Neuromechanic to investigate the role of intrinsic muscle properties and musculoskeletal anatomy on the stability of a feline hindlimb. One of the inherent difficulties of such an analysis is that production of a target endpoint force is insufficient to define a unique muscle activation pattern, but allows a large space of redundant muscle activation patterns to produce the required ground reaction forces. The mechanical stability varies among these patterns because of the intrinsic stability of individual muscles and co-contraction of antagonist pairs. We hypothesized that muscular redundancy could be resolved and postural performance improved by using musculoskeletal stability as a criterion to select muscle activation patterns to maximize stability and satisfy endpoint force production [10]. To test this hypothesis, we evaluated the Lyapunov stability of a limb under the action of many different muscle activation patterns, each producing the identical endpoint force. The model was implemented in Neuromechanic and has seven kinematic degrees of freedom and 31 muscles. The posture of the limb and ground reaction force was chosen based on experimental measurements. We then used the equilibration tool of Neuromechanic to generate random muscle activation patterns, which produced the desired ground reaction force in the desired posture. The linearization and stability analysis tool of Neuromechanic was then used to determine the stability of the limb for each of the muscle activation patterns. We found that only 35% of the random muscle activation patterns produce mechanically stable limb behavior (Figure 2(a)). The implication is that a large fraction of the activation patterns that satisfy the force constraint of the postural task require active, attentive stabilization. Furthermore, a cost function based on the moment arms and intrinsic stiffness of muscle could be used to obtain activation patterns with high stability and little co-contraction [10], suggesting that the nervous system can use mechanical stability to restrict the large available control space to a smaller domain, or even a single activation pattern, thus resolving the apparent redundancy as effectively as a metabolic cost constraint.

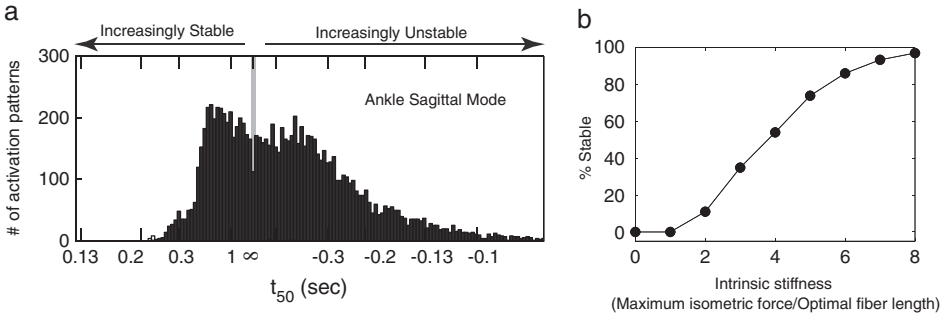

Figure 2. (a) From 10,000 tested activation patterns 35% were stable at an intrinsic stiffness level of 3 (units are maximum isometric force per optimal fiber length). Even the most unstable had a doubling time of greater than 200 ms. (b) The percentage of stable solutions increases with increasing intrinsic stiffness.

DOI: 10.1002/cnm

A limitation of this analysis is that, because of the complexity of the model, there are many parameters (including posture, ground reaction force, and magnitude of intrinsic muscle stiffness) that influence the results. For example, as the magnitude of intrinsic muscle stiffness is increased the percentage of stable patterns increases (Figure 2(b)). This demonstrates the need for sensitivity analysis of complex biomechanical models.

3.2. Illustration 2: fast computation of Lyapunov stability

We used Neuromechanic to explore the stability of the extremely large subspace of activation patterns generating a desired endpoint force using the feline hindlimb model. This subspace is described by the 24-dimensional null space of the moment arm matrix, resulting from the disjunction of the seven static joint constraints and the 31 muscles included in the model. A search for a solution with constraints or costs associated with the Lyapunov stability can be formulated by using the state derivatives from the linearization step of Neuromechanic. However, Lyapunov stability depends on the state and must be recalculated at every optimization step, requiring substantial computational overhead, which increases $\left[\mathrm{O}(n^2)\right]$ with the size of the state space. We developed a computational shortcut by approximating the state matrix as a linear function of the muscle activation vector. Although the state equations depend on muscle activation resulting in a unique state matrix for each unique activation pattern, the dependence is linear provided that muscle models contain no series elasticity [25]. Because of this linearity, the state matrix (**A** in Equation (22) above) can be written as a function of muscle activation $\left(\vec{\varepsilon}\right)$, activation-dependent ($\mathbf{A}_m$) coefficients, and activation-independent ($\mathbf{A}_0$) coefficients.

$$\mathbf{A} = \mathbf{A}_0 + \sum_{m=1}^{NumberofMuscles} \varepsilon_m \mathbf{A}_m$$
$$\text{where } \mathbf{A}_m = \left(\frac{\partial^2 f}{\partial \vec{z}\, \partial \varepsilon_m} \right)_{\vec{z} = \vec{z}_0} \tag{23}$$

These coefficients can be obtained by regression of activation patterns and their corresponding state matrices. Thereafter, additional state matrices can then be generated from arbitrary muscle activation patterns very efficiently [O(n)]. To test the validity of the method, we directly compared linearized state matrices estimated from precalculated regression coefficients in MATLAB (Mathworks, Inc., Natick, MA, USA) with those calculated directly from Neuromechanic based on a sample of 200 muscle activation patterns. Root mean square (RMS) errors between estimated and computed state matrices were $2.2 \pm 0.7 \times 10^{-6}\%$ and the shortcut was four orders of magnitude faster. Although this shortcut still requires the precalculation step, it is still computationally advantageous especially when the search space is large requiring the analysis of many activation patterns.

Use of the method requires that a relatively simple Hill-type muscle model with inelastic tendon be used. The static effect of elastic tendon in series, in essence, is that the shape of the force-length curve is distorted; range of lengths where actuator operates on the ascending region of the force-length curve is stretched. Therefore, the method may not be appropriate if it is desired to investigate the role of compliant tendons in more dynamic conditions such as when subjected to quick stretches. However, provided that the motor task in interest can be assumed quasi-static, such as in generating an isometric limb endpoint force in the human finger [28] or leg [29], such linear mapping between the muscle activation vector and the state equation at static equilibrium has been found to be valid. Moreover, this computational shortcut approximates the true state matrix even for models with series elasticity provided that the series elasticity is much larger than the parallel elasticity. As the difference in magnitude of these elasticities decreases, the approximation becomes invalid and the full state matrix should be calculated with the perturbation method. Similar separations are possible for other classes of state variables, and it is the unified formulation of the system that allows both linearization and determination of the state-dependence of the linearized state matrix.

DOI: 10.1002/cnm

3.3. Illustration 3: analysis of frontal plane standing using delayed neural controllers and intermittent ground contact

We illustrated Neuromechanic's ability to utilize delayed feedback control and intermittent ground contact by comparing a simulation to the results of a delayed feedback four-bar linkage model of frontal plane standing [30]. Delayed feedback is an important feature of neuromuscular systems, because neural conduction velocity can be slow relative to kinematics, resulting in long and potentially destabilizing reflex latencies. These delays result in highly nonlinear behavior and are difficult to model. The previous model consisted of four segments corresponding to the ground, two legs, and the torso connected by pin joints in a closed chain. Inertial and geometric properties were scaled based on average anthropometric data [31]. The leg segments consisted of a lumped representation of the shank and thigh with a locked knee and pin joints for the ankle and hip. The torso segment included a head, arms (folded across the chest), trunk, and pelvis and attached to the leg segments by pin joints at the hips. The ground segment was considered immobile and its length specified the stance width of the model. Muscular force was modeled as a lumped term and applied with constant moment arms as torque about each hip joint. Hip torque was generated as delayed feedback with fixed gains on position (255 N-m/rad) and velocity (85 N-m/rad/s). The delay was a single lumped value of 120 ms to account for neural transmission from sensation to actuation (100 ms) and mechanical actuation (20 ms) as observed from the automatic postural response [32]. Perturbations were applied to the four-bar linkage as an inertial acceleration pulse with magnitude 7.75 m/s^2 to match platform translations from experimental ramp-and-hold protocols. The equations of motion for the previous model were derived using a symbolic dynamics package (AutoLev 4.1, OnLine Dynamics, Inc). This model predicted that ground reaction force at one foot would reach zero, representing foot lift-off. The pin constraint did not permit lift-off, and further simulation would require implementation of a second, open kinetic chain model to account for periods of single limb contact.

The Neuromechanic model was assembled with the same parameters and geometry as above except for the feet, which were unpinned and free to move. This resulted in a 5 DOF rigid-body

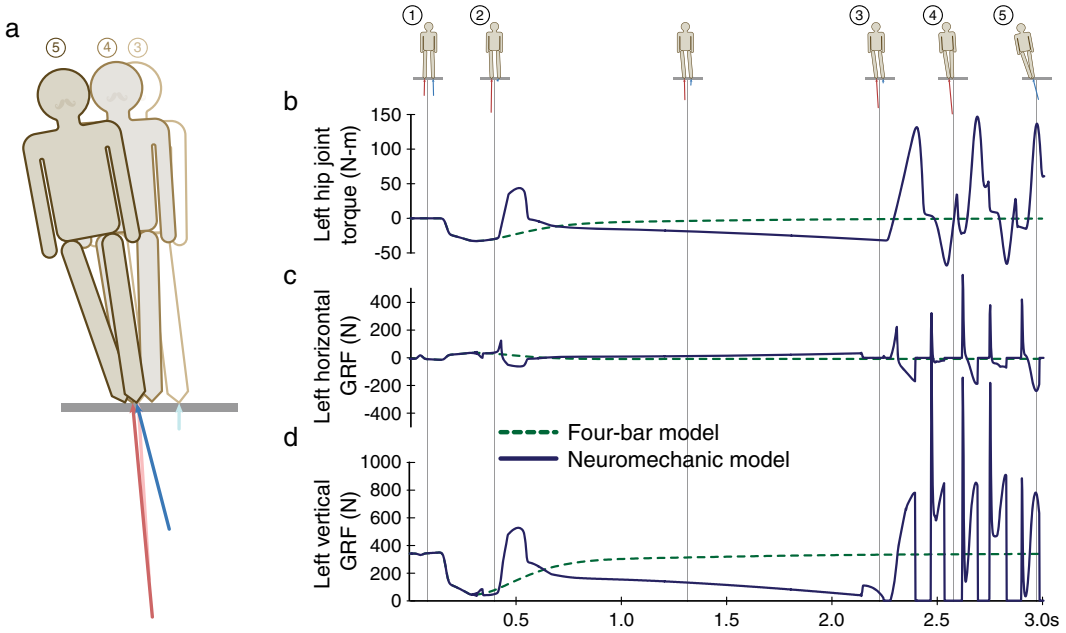

Figure 3. Neuromechanic reproduced analytical results during conditions of nonstepping (1,2). The two models diverged when the left foot slipped in the Neuromechanic simulation (2). Stepping behavior emerged in the Neuromechanic model because of frictional contact (3 – 5). (a) Overlayed frames show the sequence of stepping (3 – 5). The model results were compared for (b) hip torque and (c – d) ground reaction forces (GRF) for the left leg.

DOI: 10.1002/cnm

system of three links with the root body defined as the torso and the legs as child bodies resting on an impenetrable rigid surface with a dry frictional coefficient of 0.8. Ground reaction forces and hip torque were compared for both models for 3 s of simulation (Figure 3). During the first 400 ms of simulation the results were consistent between the two models when ground contact was maintained. At approximately 400 ms slip occurred at the left foot in the Neuromechanic model, which led to different and richer results than the pinned model. Furthermore, near 2.3 s of simulation the Neuromechanic model allowed for the modeled feet to come off of the ground leading to emergent stepping. This stepping behavior was not explicitly controlled, but rather emergent from the dynamics and was also observed in previous experiments with a physical robot [33]. The emergent stepping was not possible in the simpler, four-bar linkage model.

4. CONCLUSION

We have developed a platform that emphasizes an integrated, control-oriented approach to neuromusculoskeletal modeling. Rather than extracting the individual skeletal, muscular, and neural components from experimental data, Neuromechanic focuses on providing the tools necessary to test hypotheses of neural control within the context of a musculoskeletal model. Neuromechanic provides a way to evaluate multiple solutions for a motor task, and to search the large space of possible solutions, rather than relying on matching experimental data, or finding a single solution based on optimal control. The approach used for generating equations of motion allows the effects of rigid body, muscle, and neural states to be evaluated within the same set of equations. This allows for linearized equations of motion to be generated for stability analysis. Therefore, the generation of a trajectory and the system's ability to reject perturbations along that trajectory can be evaluated. Because these equations are constructed at each timepoint, changes in degrees of freedom associated with ground contact can be easily simulated. The tools presented should facilitate an understanding of the sensitivity of musculoskeletal dynamics to neural, muscular, and skeletal dynamics, facilitating the development of robust and stable forward dynamics simulations of movement.

ACKNOWLEDGEMENTS

NIH R01 HD046922 to Lena H. Ting and P01 HD032571 to T. Richard Nichols. Neuromechanic is available at http://www.neuromechanic.com.

REFERENCES

1. Winter DA, Patla AE, Prince F, Ishac M, Gielo-Perczak K. Stiffness control of balance in quiet standing. *Journal of Neurophysiology* 1998; **80**:1211–1221.
2. Morasso PG, Baratto L, Capra R, Spada G. Internal models in the control of posture. *Neural Networks* 1999; **12**:1173–1180.
3. van Soest A, Rozendaal L. The inverted pendulum model of bipedal standing cannot be stabilized through direct feedback of force and contractile element length and velocity at realistic series elastic element stiffness. *Biological Cybernetics* 2008; **99**:29–41.
4. Nichols TR, Houk JC. Improvement in linearity and regulation of stiffness that results from actions of stretch reflex. *Journal of Neurophysiology* 1976; **39**:119–142.
5. Masani K, Popovic MR, Nakazawa K, Kouzaki M, Nozaki D. Importance of body sway velocity information in controlling ankle extensor activities during quiet stance. *Journal of Neurophysiology* 2003; **90**:3774–3782.
6. Loram ID, Kelly SM, Lakie M. Human balancing of an inverted pendulum: Is sway size controlled by ankle impedance? *Journal of Physiology* 2001; **532**:879–891.
7. Carpenter MG, Allum JHJ, Honegger F. Directional sensitivity of stretch reflexes and balance corrections for normal subjects in the roll and pitch planes. *Experimental Brain Research* 1999; **129**:93–113.
8. Shumway-Cook A, Woollacott M. Attentional demands and postural control: The effect of sensory context. *Journals of Gerontology. Series A, Biological Sciences and Medical Sciences* 2000; **55**:10–16.
9. Higginson JS, Zajac FE, Neptune RR, Kautz SA, Burgar CG, Delp SL. Effect of equinus foot placement and intrinsic muscle response on knee extension during stance. *Gait Posture* 2006; **23**:32–36.
10. Bunderson NE, Burkholder TJ, Ting LH. Reduction of neuromuscular redundancy for postural force generation using an intrinsic stability criterion. *Journal of Biomechanics* 2008; **41**(7):1537–1544.

DOI: 10.1002/cnm

11. Risher DW, Schutte LM, Runge CF. The use of inverse dynamics solutions in direct dynamics simulations. *Journal of Biomechanical Engineering* 1997; **119**:417–422.
12. Pandy MG. Computer modeling and simulation of human movement. *Annual Review of Biomedical Engineering* 2001; **3**:245–273.
13. Anderson FC, Pandy MG. Dynamic optimization of human walking. *Journal of Biomechanical Engineering* 2001; **123**:381–390.
14. Pandy MG, Anderson FC, Hull DG. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Journal of Biomechanical Engineering* 1992; **114**:450–460.
15. Pandy MG, Anderson FC. Dynamic simulation of human movement using large-scale models of the body. *Phonetica* 2000; **57**:219–228.
16. Neptune RR. Computer modeling and simulation of human movement. Applications in sport and rehabilitation. *Physical Medicine and Rehabilitation Clinics of North America* 2000; **11**:417–434. viii.
17. Thelen DG, Anderson FC, Delp SL. Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics* 2003; **36**:321–328.
18. Thelen DG, Anderson FC. Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. *Journal of Biomechanics* 2006; **39**:1107–1115.
19. Crowninshield RD, Brand RA. A physiologically based criterion of muscle force prediction in locomotion. *Journal of Biomechanics* 1981; **14**:793–801.
20. García de Jalón J, Bayo E. *Kinematic and Dynamic Simulation of Multibody Systems : The Real-Time Challenge*, Mechanical Engineering Series. Springer-Verlag: New York, 1994. xvi, 440 p.
21. Featherstone R. *Rigid Body Dynamics Algorithms*. New York: Springer, 2008. ix, 272 p.
22. Shabana AA. *Vibration of Discrete and Continuous Systems*, (2nd edn), Mechanical Engineering Series. Springer: New York, 1997. xv, 393 p.
23. Kane TR, Levinson DA. *Dynamics, Theory and Applications*, Mcgraw-Hill Series in Mechanical Engineering. McGraw-Hill: New York, 1985. xv, 379 p.
24. Kaufman DM, Sueda S, James DL, Pai DK. Staggered projections for frictional contact in multibody systems. *ACM Transactions on Graphics* 2008; **27**:1.
25. Zajac FE. Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control. *Critical Reviews in Biomedical Engineering* 1989; **17**:359–411.
26. Schutte LM, Rodgers MM, Zajac FE, Glaser RM. Improving the efficacy of electrical stimulation-induced leg cycle ergometry: An analysis based on a dynamic musculoskeletal model. *IEEE Transactions on Rehabilitation Engineering* 1993; **1**:109–125.
27. Hairer E, Nørsett SP, Wanner G. *Solving Ordinary Differential Equations*, (2nd rev. edn), Springer Series in Computational Mathematics. Springer-Verlag: Berlin; New York, 1993.
28. Valero-Cuevas FJ, Towles JD, Hentz VR. Quantification of fingertip force reduction in the forefinger following simulated paralysis of extensor and intrinsic muscles. *Journal of Biomechanics* 2000; **33**:1601–1609.
29. Kuo AD, Zajac FE. Human standing posture: Multi-joint movement strategies based on biomechanical constraints. *Progress in Brain Research* 1993; **97**:349–358.
30. Bingham JT, Choi JT, Ting LH. Stability in a frontal plane model of balance requires coupled changes to postural configuration and neural feedback control. *Journal of Neurophysiology* 2011; **106**:437–448.
31. Winter DA. Motor mechanisms of balance during quiet standing. *Journal of Electromyography and Kinesiology* 2003; **13**:49–56.
32. Horak FB, Macpherson JM. Postural Orientation and Equilibrium. In *Handbook of physiology*. American Physiological Society: New York, 1996; 255–292.
33. Scrivens JE, Deweerth SP, Ting LH. A robotic device for understanding neuromechanical interactions during standing balance control. *Bioinspiration and Biomimetics* 2008; **3**. 026002.

DOI: 10.1002/cnm