# Multi-scale Hyper-time Hardware Emulation of Human Motor Nervous System using FPGA Based on Spiking Neurons

**C. Minos Niu**
Department of Biomedical Engineering
University of Southern California
Los Angeles, CA 90089
minos.niu@sangerlab.net

**Sirish K. Nandyala**
Department of Biomedical Engineering
University of Southern California
Los Angeles, CA 90089
nandyala@usc.edu

**Won Joon Sohn**
Department of Biomedical Engineering
University of Southern California
Los Angeles, CA 90089
wonjsohn@gmail.com

**Terence D. Sanger**
Department of Biomedical Engineering
Department of Neurology
Department of Biokinesiology
University of Southern California
Los Angeles, CA 90089
terry@sangerlab.net

## Abstract

Our central goal is to quantify the long-term progression of pediatric neurological diseases, such as a typical 10-15 years progression of child dystonia. To this purpose, quantitative models are convincing only if they can provide multi-scale details ranging from neuron spikes to limb biomechanics. The models also need to be evaluated in hyper-time, i.e. significantly faster than real-time, for producing useful predictions. We designed a platform with digital VLSI hardware for multi-scale hyper-time emulations of human motor nervous systems. The platform is constructed on a scalable, distributed array of Field Programmable Gate Array (FPGA) devices. All devices operate asynchronously with 1 millisecond time granularity, and the overall system is accelerated to 365x real-time. Each physiological component is implemented using models from well documented studies and can be flexibly modified. Thus the validity of emulation can be easily advised by neurophysiologists and clinicians. For maximizing the speed of emulation, all calculations are implemented in combinational logic instead of clocked iterative circuits. This paper presents the methodology of building FPGA modules in correspondence to components of a monosynaptic spinal loop. Results of emulated activities are shown. The paper also discusses the rationale of approximating neural circuitry by organizing neurons with sparse interconnections. In conclusion, our platform allows introducing various abnormalities into the neural emulation such that the emerging motor symptoms can be analyzed. It compels us to test the origins of childhood motor disorders and predict their long-term progressions.

## 1 Challenges of studying developmental motor disorders

There is currently no quantitative model of how a neurological disease process, which mainly affects the function of neurons, ends up causing the functional abnormalities identified in clinical examinations. The gap in knowledge is particularly evident for disorders of the developing human nervous system, i.e. childhood neurological diseases. In these cases, the ultimate clinical effect of cellu-

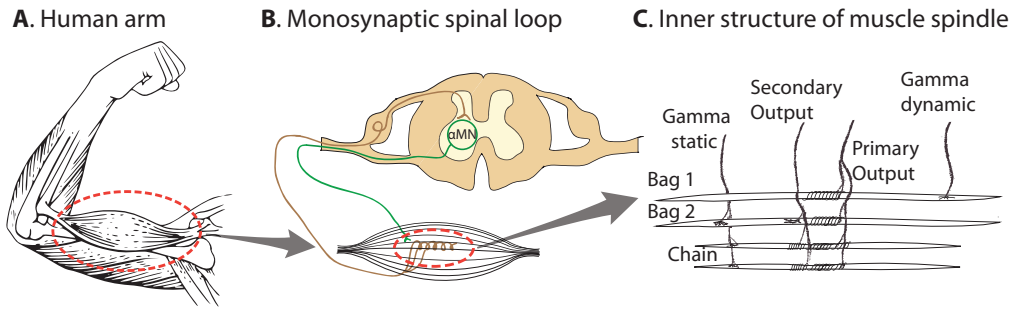


Figure 1: Illustration of the multi-scale nature of motor nervous system.

lar injury is compounded by a complex interplay among the child's injury, development, behavior, experience, plasticity, etc. Clinical experience has provided qualitative insight into the association between particular types of injury and particular types of outcome. Nevertheless, no existing measures – neither in clinic nor in cellular physiological tests – are sufficient enough to provide the linkage between cellular injuries and specific functional outcomes, especially in individual child patients. In order to understand the effect of injury and avert its functional hindrance by discovering new treatments, it is necessary to create a toolbox and set of design techniques such that all neural elements involved in the disease can be quantitatively modeled.

Perhaps more than any other organ, the brain necessarily operates on multiple spatial and temporal scales. On the one hand in terms of spatial scales, it is the individual neurons that perform fundamental computations, but the overall function and behavior emerge from the coupling between neurons and large-scale organs, e.g. ears, eyes, skeletal muscles, etc. When injured, neural functions depend on both the cellular effects of injury as well as the informational consequences of injury. Take motor nervous system as an example, quantitative modeling is only convincing if it can provide multi-scale details ranging from neuron spikes to limb biomechanics. On the other hand, neurological processes involved in disease progression usually operate on drastically different time scales, e.g. spinal reflex measured in milliseconds versus learning measured in years. This means that all models should be evaluated with time granularity as small as 1 millisecond, but the evaluation needs to continue trillions of cycles in order to cover years of progression.

It is particularly challenging to account for the multi-scale nature of human nervous system when modeling childhood movement disorders. Note that for a child received brain injury at birth, the full development of all motor symptoms may easily take more than 10 years. Therefore the large-scale, millisecond-based model needs to be evaluated significantly faster than real-time, otherwise the model will fail to produce any useful predictions in time. In this project we implemented realistic assemblies of spiking motoneurons, sensory neurons, neural circuitry, muscle fibers and proprioceptors. With VLSI and programmable logic technologies, all models were computed in Field Programmable Gate Array (FPGA) hardware in 365 times real-time. Therefore one year's disease progression can be assessed after one day of emulation. This paper first presented the methodology of building the hardware platform for multi-scale hyper-time neural emulations. The results demonstrate that is is feasible of using this platform to produce multi-scale information which are usually scarce in experiments. Successful modeling and emulation enabled by this platform will verify implications and consequences of particular hypotheses about neural injury. New treatment mechanism and drug effects can also be tested on this platform before in animals or clinical trials.

## 2 Methodology of multi-scale neural emulation

The motor part of human nervous system is responsible for maintaining body postures and generating voluntary movements. The multi-scale nature of motor nervous system is illustrated in Fig.1. When the elbow (Fig.1A) is maintaining a posture or performing a movement, the involved muscle produces force based on how much spiking excitation is delivered from its alpha-motoneurons (Fig.1B). The alpha-motoneurons are regulated by their own sensory input, which in-turn comes from the proprioceptors residing in the muscle. As the primary sensory organ found in skeletal muscles, a muscle spindle is another complex system that has its own microscopic Multiple-Input-

Multiple-Output structure (Fig.1C). Spindles continuously provide information about the length and lengthening speed of the muscle fiber. This paper uses the monosynaptic spinal loop as an example for explaining the methodology of multi-scale hyper-time neural emulation in hardware. Additional structures can be added to the backbone platform using similar methods described here.

### 2.1 Modularized architecture for multi-scale models

Decades of studies on neurophysiology provided an abundance of models characterizing different components of the human motor nervous system. The functional differentiation between physiological components allowed us to model the motor nervous system as concatenated structures that map input signals to the output. In particular, in a monosynaptic spinal loop illustrated in Fig.1B, stretching the muscle will elicit a chain of physiological activities: Muscle stretch ⇒ Spindle ⇒ Sensory Neuron ⇒ Synapse ⇒ Motoneuron ⇒ Muscle contraction. The adjacent components must have compatible interfaces, and the interfacing variables must also be physiologically realistic. In our design, each component is mathematically described in Table 1:

Table 1: Functional definition of neural models

| COMPONENT | MATHEMATICAL DEFINITION |
|---|---|
| Neuron | $S(t) = f_{\text{neuron}}(I, t)$ |
| Synapse | $I(t) = f_{\text{synapse}}(S, t)$ |
| Muscle | $T(t) = f_{\text{muscle}}(S, L, \dot{L}, t)$ |
| Spindle | $A(t) = f_{\text{spindle}}(L, \dot{L}, \Gamma_{\text{dynamic}}, \Gamma_{\text{static}}, t)$ |

As can be seen, all components are modeled as black-box functions that map the inputs to the outputs. The meanings of these mathematical definitions are explained below. This design allows existing physiological models to be easily inserted and switched. In all models the input signals are time-varying, e.g. $I = I(t),\ L = L(t)$ , etc. The argument of $t$ in input signals are omitted throughout this paper.

### 2.2 Selection of models for emulation

Models were selected in consideration of their computational cost, physiological verisimilitude, and whether it can be adapted to the mathematical form defined in Table 1.

#### Model of Neuron

Neurons take post-synaptic current $I$ as the input, and produce a binary spike train $S$ in the output. The neuron model adopted in the emulation was developed by Izhikevich [1]:

$$v' = 0.04v^2 + 5v + 140 - u + I \quad (1)$$

$$u' = a(bv - u) \quad (2)$$

$$\text{if } v = 30 \text{ mV}, \text{ then } v \leftarrow c,\ u \leftarrow u + d$$

where the output is the action potential $v$, which directly produces a binary spike train; $a, b, c, d$ are model parameters that need to be tuned based on the neuron's firing properties. Note that in Izhikevich model the action potential $v$ is in millivolts and the time is in milliseconds. Since all other models require SI units the coefficients in eq.1 need to be adjusted.

#### Model of Synapse

When a pre-synaptic neuron fires, i.e. $S(0) = 1$, an excitatory synapse subsequently produces an Excitatory Post-Synaptic Current (EPSC) that drives the post-synaptic neuron. Neural recording of hair cells in rats [2] provided evidence that the time profile of EPSC can be well characterized using the equations below:

$$I(t) = \begin{cases} V_m \times \left( e^{-\frac{t}{\tau_d V_m}} - e^{-\frac{t}{\tau_r V_m}} \right) & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The key parameters in a synapse model is the time constants for rising ($\tau_r$) and decaying ($\tau_d$). In our emulation $\tau_r = 1$ ms and $\tau_r = 3$ ms.

**Model of Muscle force and electromyograph (EMG)**

The primary effect of skeletal muscle is converting the alpha-motoneuron spikes $S$ into a force $T$, which also depends on the instantaneous length $L$ and lengthening speed $\dot{L}$ of the muscle itself. We used Hill's muscle model in the emulation. The parameter tuning is based on [3]. Another measurable output of muscle is electroencephalograph (EMG). EMG is the small skin current polarized by motor unit action potential (MUAP) when it travels along muscle fibers. Models exist to describe the typical waveform picked by surface EMG electrodes. In this project we chose to implement the one described in [4].

**Model of Proprioceptor**

Spindle is a sensory organ that provides the main source of proprioceptive information. As can be seen in Fig.1C, a spindle typically produces two afferent outputs (Group Ia and II) according to the muscle statuses ($L$ and $\dot{L}$) and gamma fusimotor drives ($\Gamma_{\text{dynamic}}$ and $\Gamma_{\text{static}}$). There is currently no closed-form models describing spindle functions. This is due to spindle's significant non-linearity, which originated from the non-linear nature of muscle fibers and their coupling with spike generating spots. On representative model of spindle dynamics was developed by Mileusnic et al. [5], which described the dynamics of cat soleus spindle using a set of differential equations. Eqs.3 to 9 show a subset of this model for spindle bag1 fiber:

$$\dot{x_0} = \left( \frac{\Gamma_{\text{dynamic}}}{\Gamma_{\text{dynamic}} + \Omega_{\text{bag1}}^2} - x_0 \right) / \tau \tag{3}$$

$$\dot{x_1} = x_2 \tag{4}$$

$$\dot{x_2} = \frac{1}{M} \left[ T_{SR} - T_B - T_{PR} - \Gamma_1 x_0 \right] \tag{5}$$

where

$$T_{SR} = K_{SR}(L - x_1 - L_{SR0}) \tag{6}$$

$$T_B = (B_0 + B_1 x_0) \cdot (x_1 - R) \cdot CSS \cdot |x_2|^{0.3} \tag{7}$$

$$T_{PR} = K_{PR} (x_1 - L_{PR0}) \tag{8}$$

$$CSS = \left( \frac{2}{1 + e^{-1000 x_2}} \right) - 1 \tag{9}$$

Eq.7 and 9 suggest that evaluating the spindle model requires multiplication, division as well as more complex arithmetics like polynomials and exponentials. The technological details of implementation are described in Section 3.

### 2.3 Neuron connectivity with sparse interconnections

Although the number of spinal neurons (~1 billion) is significantly less compared to that in the brain cortex (~100 billion), a fully connected spinal network still means approximately $2 \times 10^{12}$ synaptic endings [6]. Implementing such a huge number of synapses imposes a major challenge, if not impossible, given limited FPGA resource.

In this platform we approximated the neural connectivity by sparsely connecting sensory neurons to motoneurons as parallel pathways. We do not attempt to introduce the full connectivity. The rationale is that in a neural control system, the effect of a single neuron can be considered as mapping current state $x$ to change in state $\dot{x}$ through a band-limited channel. Therefore when a collection of neurons are firing stochastically, the probability of $\dot{x}$ depends on both $x$ and each neuron's firing behavior $s$ ($s = 1$ when spiking, otherwise $s = 0$), as such:

$$p(\dot{x}|x, s) = p(\dot{x}|s = 1)p(s = 1|x) + p(\dot{x}|s = 0)p(s = 0|x) \tag{10}$$

Eq.10 is a master equation that determines a probability flow on the state. From the Kramers-Moyal expansion we can associate this probability flow with a partial differential equation for the change

in probability density:

$$\begin{aligned}\frac{\partial}{\partial t}p(x,t) &= \frac{\partial}{\partial x}\left(D_1(x)p(x,t)\right) \\ &+ \frac{\partial^2}{\partial x^2}\left(D_2(x)p(x,t)\right) + \cdots \end{aligned} \tag{11}$$

It has been shown in [7, 8] that when higher order (>2) fluctuations in the probability density are ignored, the probability flow can be deterministically described using a linear operator $\mathcal{L}$:

$$\frac{\partial}{\partial t}p(x,t) = \mathcal{L}p(x,t) \tag{12}$$

This means that various $\mathcal{L}$s can be superimposed to achieve complex system dynamics (illustrated in Fig.2A). As a consequence, the statistical effect of two fully connected neuron populations is equivalent to ones that are only sparsely connected, as long as the probability flow can be described by the same $\mathcal{L}$. In particular, in a movement task it is the statistical effect from the neuron ensemble to skeletal muscles that determines the global behavior. Therefore we argue that it is feasible to approximate the spinal cord connectivity by sparsely interconnecting sensory and motor neurons (Fig.2B). It is worth noting that this approximation significantly reduces the number of synapses that need to be implemented in hardware.

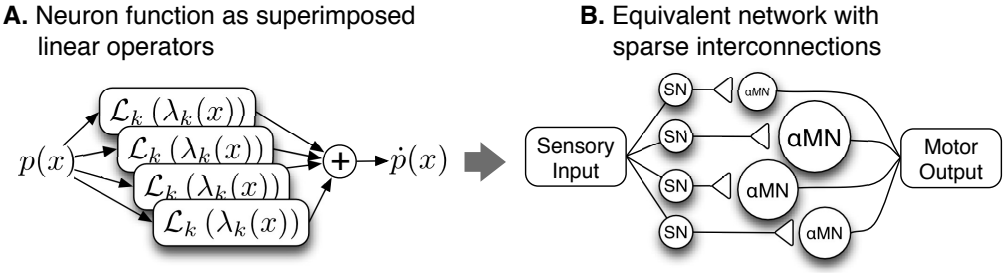


Figure 2: Functions of neuron population can be described as the combination of linear operators (A). Therefore the function can be approximated by sparsely connected parallel pathways (B).

## 3 Hardware implementation on FPGA

The platform is distributed on multiple nodes of Xilinx Spartan-6 FPGA devices. The interfacing among FPGAs and computers is created using OpalKelly development board XEM6010. Variables in the Izhikevich neuron model, synapse model and EMG model have a tight dynamic range, which guarantees that fixed-point arithmetics is accurate enough for model evaluation. The spindle model, in contrast, requires floating-point arithmetics due to its wide dynamic range and complex calculations (see eq.3-9). Hyper-time computations with floating-point numbers are resource consuming and therefore need to be implemented with special attentions.

### 3.1 Floating-point arithmetics in combinational logic

Our arithmetic implementations are compatible with IEEE-754 standard. Typical floating-point arithmetic IP cores are either pipe-lined or based on iterative algorithms like CORDIC, all of which require clocks to schedule the calculation. In our platform, no clock is provided for model evaluations thus all arithmetics need to be executed in pure combinational logic. Taking advantage of combinational logic allows all model evaluations to be 1) fast, the evaluation time entirely depends on the time duration for signal propagation and settling, which is on the order of microseconds, and 2) parallel, each model is evaluated on its own circuit without waiting for any other results.

Our implementations of adder and multiplier are inspired by the open source project "Free Floating-Point Madness", available at http://www.hmc.edu/chips/. Please contact the authors of this paper if the modified code is needed.

**Fast combinational floating-point division**

Floating-point division is usually more resource demanding than multiplications. We avoided directly implementing the dividing algorithm by approximating it with additions and multiplications. Our approach is inspired by an algorithm described in [9], which provides a good approximation of the inverse square root for any positive number $x$ within one Newton-Raphson iteration:

$$Q(x) = \frac{1}{\sqrt{x}} \approx x(1.5 - \frac{x}{2} \cdot x^2) \quad (x > 0) \tag{13}$$

$Q(x)$ can be implemented only using floating-point adders and multipliers. Thereby any division with a positive divisor can be achieved by concatenating two blocks of $Q(x)$:

$$\frac{a}{b} = \frac{a}{\sqrt{b} \cdot \sqrt{b}} = a \cdot Q(b) \cdot Q(b) \quad (b > 0) \tag{14}$$

This algorithm has been adjusted to also work with negative divisors.

**Numerical integrators for differential equations**

Evaluating the instantaneous states of differential equation models require a fixed-step numerical integrator. Backward Euler's Method was chosen to balance the numerical error and FPGA usage:

$$\dot{x} = f(x, t) \tag{15}$$

$$x_{n+1} = x_n + Tf(x_{n+1}, t_{n+1}) \tag{16}$$

where $T$ is the sampling interval. $f(x, t)$ is the derivative function for state variable $x$.

### 3.2 Serialize neuron evaluations within a homogeneous population

Different neuron populations are instantiated as standalone circuits. Within in a population of neurons, however, the homogeneous neurons mentioned in Section 2.3 are evaluated in series for optimizing FPGA usage.

Within each FPGA node all modules operate synchronously, meaning that all signal updates are triggered by a central clock of emulation. The number of neurons that can be serialized ($N_{\text{serial}}$) is determined by the following relationship:

$$F_{\text{fpga}} = C \times N_{\text{serial}} \times 365 \times F_{\text{emu}} \tag{17}$$

Here $F_{\text{fpga}}$ is the fastest kernel clock provided to FPGA; $C = 4$ is the minimal clock cycles needed for updating each state variable in the block RAM; $F_{\text{emu}} = 1kHz$ denotes the time granularity of emulation (1 millisecond), and $365 \times F_{\text{emu}}$ represents 365x real-time. Consider that Xilinx Spartan-6 FPGA devices peaks at 200MHz central clock frequency (i.e. $F_{\text{fpga}} \leqslant 200MHz$), the theoretical maximum of neurons that can be serialized is

$$N_{\text{serial}} \leqslant 200 \text{ MHz}/(4 \times 365 \times 1 \text{ kHz}) \approx 137 \tag{18}$$

In the current design we choose $N_{\text{serial}} = 128$.

### 3.3 Asynchronous spike-based communication between FPGA chips

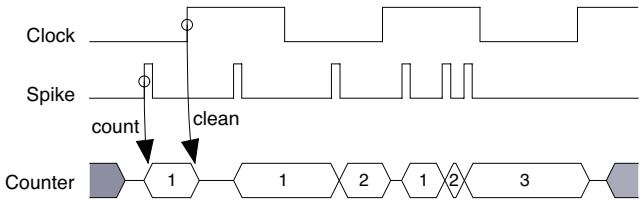


Figure 3: Timing diagram of asynchronous spike-based communication

FPGA nodes are networked by transferring spikes to each other. Our design allowed the sender and the receiver to operate on independent clocks without having to synchronize. The timing diagram

of the spike-based communication is shown in Fig.3. The sender sends off Spike on a 1-bit channel with pulse width of $1/(365 \times F_{\text{emu}})$. On the receiver each Spike triggers a counting event; each Clock reads the accumulated count of spikes and cleans the counter afterward. Note that the time difference between Spike and Clock is not predictable due to asynchronicity.

## 4 Results: emulated activities of motor nervous system

Pictures of a working FPGA node, two networked nodes and a screenshot of the software front-end are shown in Figure 4. Each FPGA node is able to emulate monosynaptic spinal loops consisting of 1,024 sensory neurons and 1,024 motor neurons. The spike-based asynchronous communication is successful between two FPGA nodes. On-line plotting of the signals requires the emulation to be significantly slowed down. When the emulation is at full speed (365x real-time) the software front-end is not able to visualize the signals due to limited data throughput.

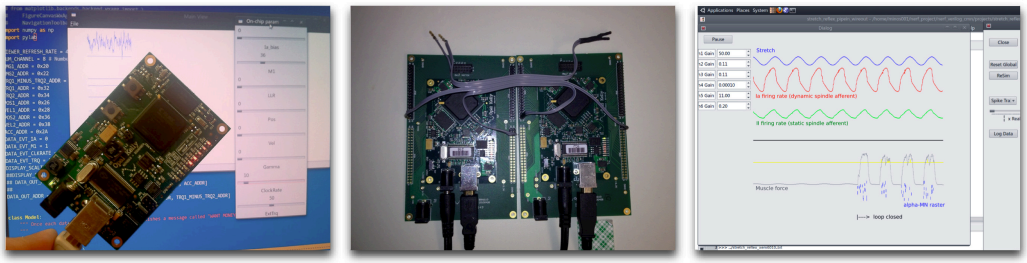


Figure 4: Pictures of the neural emulation platform in operation. Left – One working FPGA node. Center – Two FPGA nodes networked using asynchronous spiking protocol. Right – Software front-end displaying multi-scale signals.

The emulation platform successfully created multi-scale information when the muscle is externally stretched (Fig.5A). We also tested if our emulated motor system is able to produce the recruitment order and size principles that are close to real physiological data. When a voluntary motor command is sent to the alpha-motoneuron pool, the motor units are recruited in an order that small ones get recruited first and then the big ones. The results are shown in Fig.5B, where the top panel shows decoded motor unit activities from real human EMG [10], and the bottom panel shows 20 motor unit activities emulated using our platform. No qualitative difference was found.

## 5 Discussion and future work

We designed a hardware platform for emulating the multi-scale motor nervous activities in hyper-time. We managed to use one node of single Xilinx Spartan-6 FPGA to emulate monosynaptic spinal loops consisting of 2,048 neurons, associated muscles and proprioceptors. The neurons are organized as parallel pathways with sparse interconnections. The emulation is successfully accelerated to 365x real-time. The platform can be scaled by networking multiple FPGA nodes, which is enabled by an asynchronous spike-based communication protocol. The emulated monosynaptic spinal loops are capable of producing reflex-like activities in response to muscle stretch. Our results of motor unit recruitment order are compatible with the physiological data collected in real human subjects.

It has been shown [11] that replicating classic types of spinal interneurons (propriospinal, Ia-excitatory, Ia-inhibitory, Renshaw, etc.) is sufficient to produce stabilizing responses and rapid reaching movement in a wrist. Our platform will introduce those interneurons to describe the known spinal circuitry in further details. Physiological models will be refined as needed, too. Izhikevich model is a good balance between verisimilitude and computational cost, which makes it adequate for emulating the movement behavior driven by spiking motoneurons. Consider that one purpose of our platform is to test the drug effect during disease progression, the neuron model is expected to cover sufficient molecular details like how neurotransmitter affect the ion channels and eventually

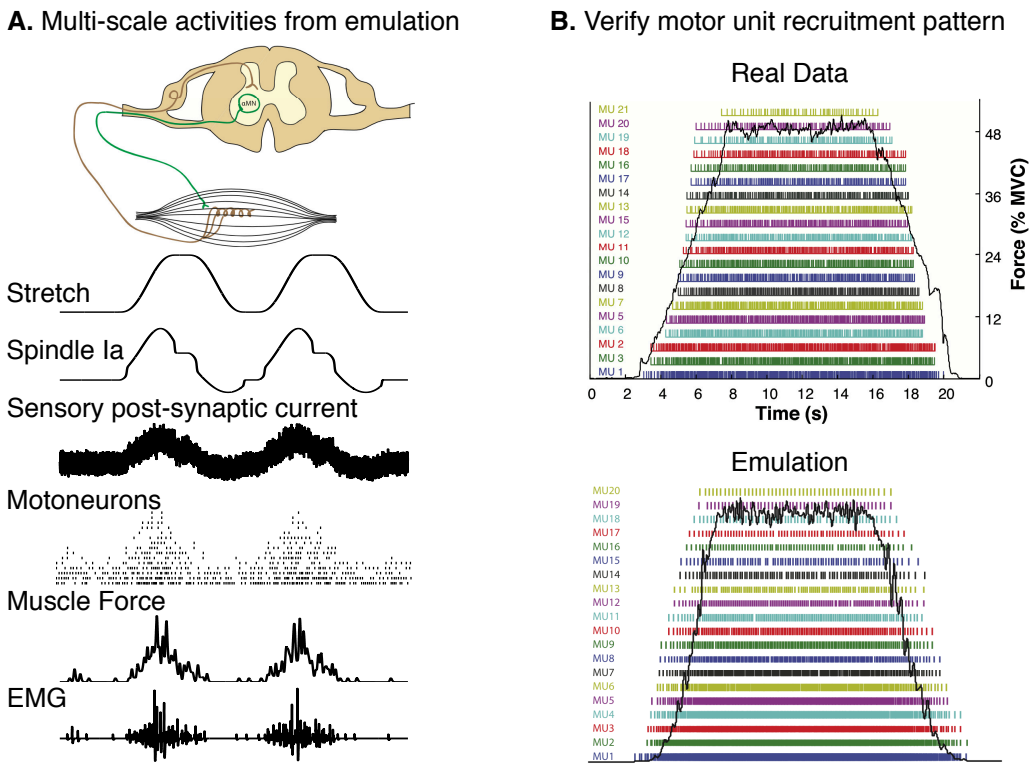


Figure 5: A) Physiological activity emulated by each model when the muscle is sinusoidally stretched. B) Comparing the emulated motor unit recruitment order with real experimental data.

the firing property of the neuron. With the advancing of programmable semiconductor technology, it is possible to upgrade to Hodgkin-Huxley neuron models. For the muscle models, Hill's type of model does not fit the muscle properties accurately enough when the muscle is being shortened. Alternative models will be tested.

Other studies showed that the functional dexterity of human limbs – especially in the hands – is critically enabled by the tendon configurations and joint geometry [12]. As a result, if our platform is used to understand whether known neurophysiology and biomechanics are sufficient to produce able and pathological movements, it will be necessary for the platform to control human-like limbs. Since the emulation speed can be flexibly adjusted from arbitrarily slow to 365x real-time, the speed can be set to exactly 1x real-time such that the platform will function as a digital controller with 1kHz refresh rate.

The main purpose of the emulation is to learn how certain motor disorders progress during childhood development. This first requires the platform to reproduce motor symptoms that are compatible with clinical observations. For example it has been suggested that muscle spasticity in rats is associated with decreased soma size of alpha-motoneurons [13], which presumably reduced the firing threshold of neurons. Thus when lower firing threshold is introduced to the emulated motoneuron pool, similar EMG patterns as in [14] should be observed. It is also necessary for the symptoms to evolve with neural plasticity. In the current version we presume that the structure of each component remains time invariant. In the future work Spike timing dependent plasticity (STDP) will be introduced such that all components are subject to structural change.

**Acknowledgments**

The authors thank Dr. Gerald Loeb for helping set up the emulation of spindle models. This project is supported by NIH NINDS grant R01NS069214-02.

## References


[1] E M Izhikevich. Simple model of spiking neurons. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 14(6):1569–1572, 2003.

[2] Elisabeth Glowatzki and Paul A Fuchs. Transmitter release at the hair cell ribbon synapse. *Nature neuroscience*, 5(2):147–154, January 2002.

[3] Reza Shadmehr and Steven P. Wise. A Mathematical Muscle Model. In *Supplementary documents for "Computational Neurobiology of Reaching and Pointing"*, pages 1–18. MIT Press, Cambridge, MA, 2005.

[4] Andrew J Fuglevand, David A Winter, and Aftab E Patla. Models of recruitment and rate coding organization in motor-unit pools. *Journal of neurophysiology*, 70(6):2470–2488, December 1993.

[5] Milana P Mileusnic, Ian E Brown, Ning Lan, and Gerald E Loeb. Mathematical models of proprioceptors. I. Control and transduction in the muscle spindle. *Journal of neurophysiology*, 96(4):1772–1788, October 2006.

[6] Samuel Gelfan, Grace Kao, and Daniel S Ruchkin. The dendritic tree of spinal neurons. *The Journal of comparative neurology*, 139(4):385–411, August 1970.

[7] Terence D Sanger. Neuro-mechanical control using differential stochastic operators. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 4494–4497, 2010.

[8] Terence D Sanger. Distributed control of uncertain systems using superpositions of linear operators. *Neural computation*, 23(8):1911–1934, August 2011.

[9] Chris Lomont. Fast inverse square root [online]. 2003. Available from: `http://www.lomont.org/Math/Papers/2003/InvSqrt.pdf`.

[10] Carlo J De Luca and Emily C Hostage. Relationship between firing rate and recruitment threshold of motoneurons in voluntary isometric contractions. *Journal of neurophysiology*, 104(2):1034–1046, August 2010.

[11] Giby Raphael, George A Tsianos, and Gerald E Loeb. Spinal-like regulator facilitates control of a two-degree-of-freedom wrist. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30(28):9431–9444, July 2010.

[12] Francisco J Valero-Cuevas, Jae-Woong Yi, Daniel Brown, Robert V McNamara, Chandana Paul, and Hood Lipson. The tendon network of the fingers performs anatomical computation at a macroscopic scale. *IEEE transactions on bio-medical engineering*, 54(6 Pt 2):1161–1166, June 2007.

[13] Allison Brashear and Elie Elovic. *Spasticity: Diagnosis and Management*. Demos Medical, 1 edition, August 2010.

[14] M F Levin and A G Feldman. The role of stretch reflex threshold regulation in normal and impaired motor control. *Brain research*, 657(1-2):23–30, September 1994.