Production of Simulated EMG Using the Physiological Model of the `simtext` Program

A. Hamilton-Wright, Daniel W. Stashuk

June 17, 2008

Abstract

The `simtext` program is explained in some depth, including the purpose and use of various controls, as well as the output file format.

A discussion of use cases involving the analysis of the same muscle morphology from different needle locations is provided, allowing the investigator to explore various interesting scenarios of muscle structure not possible in real data.

1 Simulation of EMG Data

The `simtext` program provides a text-based interface to the simulation routines described in Hamilton-Wright and Stashuk (2005), and were presented at the AAEM (Hamilton-Wright, Stashuk and Doherty, 2002, 2003a,b).

The output files of the `simtext` program contain synthetic EMG signals consistent with those acquired through needle electrodes from the medium to large limb muscles of human subjects.

These synthetic signals can be used to exercise decomposition routines based on signal analysis, and are therefore compatible with the format of the DQEMG program (Stashuk, 1999, 2001; Stashuk and Brown, 2002). The default signal generated emulates a signal acquired through a concentric needle electrode at 31250 samples/second and bandpass filtered from 10-10,000Hz.

This document describes the function of this program, the format of the data files produced, and the control parameters that may be used to generate simulations of different types of muscle tissue.

Section 2 provides an overview of the function of the program, including command-line flags and overall output data structure. Section 3 describes the model used for the development of disease, including the control values required for the model and their general effects. Section 4 describes all the control values for the simulator, including those for both normal and diseased data simulation, and Section 5 contains a description of the output data formats.

Section 6 details some operations that should help make clear the use and potential of the `simtext` program.

2 Program Overview

The simulator program is called `simtext`. Running this program will produce data consistent with a single contraction acquired from a single muscle.

The program has been written to use a command-line interface and text-based menus. Under a Microsoft Windows™ environment, it is written to function if "double-clicked" upon, but it is primarily designed to be run from within a "terminal window" (also known as a "DOS window").

Running in a "terminal window" will provide the ability to run the program with any of a number of flags, as described below. Flags are indicated as an argument to the program beginning with a '-' character. These will change the programs behaviour, including the output data location and the mechanism used to determine the input options.

The functionality of the program is the same, regardless of what operating system it is run under. The code is known to compile and run correctly (as of July, 2007) under the following operating systems:

- 32-bit Microsoft Windows™ (NT,2000,XP *etc*.)
- PowerPC™ and Intel™ based Macintosh™ OSX
- Linux©
- Free/Net/OpenBSD©

Note that no testing has been performed under any 64-bit operating systems at this time.

2.1 Running the Program

The program is set up to allow you to produce *sets* of contractions. For this reason, there is a known location for all the output files (called the "output hierarchy"), and a configuration file that is read and re-written each run, to preserve settings from the last run. If either of these are missing, they will be created on the first run.

There are therefore two basic strategies to run the program:

1. the program may be run interactively, and a user may examine/change options through a text-based menu system
2. the program may be directed to read all options from the text-based configuration file

In either case, if the configuration file is edited outside of the `simtext` program, the next run will take these changes into account. This feature can be used to run the `simtext` program inside a script by manipulating the attribute value pairs in the configuration file.

2.1.1 Running `simtext` Interactively

When run, a screen of options is presented to the user; these options can be changed to reflect different types and sizes of muscle, as well as different disease types. The full list of these settings is outlined in Section 4. The settings last used when the simulator was run are placed in the file `simulator.cfg` in the base directory of the output directory tree.

Once a run has begun, the simulator will print output on the screen as the simulation progresses. Portions of the simulation that may take a long time are provided with an estimate of completion, in order that it does not appear the simulation has entered an infinite loop. The simulation of a single contraction may take quite a while; it can be expected that 2-3 minutes may pass on even the fastest machines.

2.1.2 simtext Command Line Flags

All command line flags may be found with a brief accompanying help text by running the simulator using the option "simtext -help." For reference, these flags are:

-version : print version info; this is most useful to identify a version of the simulator code

-skip-confirm : skip user confirmation and run in batch mode. All configuration data will be loaded from the most recent simulator.cfg file if present. Implies -nowait-for-key and -runOnce.

-wait-for-key : wait for ENTER to be pressed before exiting This option is intended to allow use if double-clicked in a Windows environment, so that the user can see the screen output printed before the program exits. This is the default on Windows.

-nowait-for-key : do not wait for ENTER to be pressed before exiting. Useful in a batch program context. This is the default on all operating systems *except* Windows.

-runLoop : Loop around for multiple studies — intended to provide useful functionality from a double-click in Windows.

-runOnce : Only run one study — best for batch programs

-useLastMuscle : run again using information from last muscle in order to generate another contraction, presumably with a different needle position

-useOldFiringTimes : if generating another "contraction" from the last muscle, use old firing times as well. This allows exploration of the effect of needle position, as everything else will be exactly the same (*i.e.*, a MUP observed in one contraction will be present (although possibly with a different shape) at the same time offset in another contraction generated in this way

-useNewFiringTimes : generate new firing times to simulate a new contraction.

-use-drive<driveLetter> : (Windows only) use a drive other than c: to build the output hierarchy (see next section).

2.2 Output Hierarchy

The "output hierarchy" refers to the directory in which output will be organized. In the root of the "output hierarchy" will be the simulator.cfg file containing configuration information used for the next run. Also in this directory will be sub-directories containing the data produced by the simulator in each run.

2.2.1 Windows Output Hierarchy

The output tree of the simulator will be built in the directory c:\simulator unless the option -use-drive<driveL is supplied, in which case all output will be placed based in a tree rooted in the drive indicated by <driveLetter>.

2.2.2 Macintosh & Unix Output Hierarchy

On non-Windows based operating systems, the top of the output hierarchy just mentioned will be placed in a directory called "data" created in the directory from which the simtext executable was run (*i.e.*, the "current working directory").

2.2.3 Output Hierarchy Organization

On all machines, the output hierarchy will be as follows:

file `simulator.cfg` – this file contains all of the "current" settings for the simulator. When run, the simulator reads this file to get initial values; updating this file if any values are changed by the user. A "batch" mode of operation may be obtained by placing the appropriate settings in this file and running the simulator with the option `-skip-confirm`.

dir `run`*ID* – data produced

- **dir** `patient`
  - **dir** `emg` – directory containing contraction
    - **file** `contraction`*ID*`.dat` – DQEMG-readable DQEmgData contraction file
    - **file** `micro`*ID*`.gst` – DQEMG-readable gold-standard annotation file
    - **file** `micro`*ID*`.dat` – DQEMG-readable Comperio needle contraction file
    - **file** `macro`*ID*`.dat` – DQEMG-readable Comperio surface contraction file
    - **file** `simulator`*ID*`.cfg` – simulator settings for contraction *ID*
  - **dir** `MFP-Data` – datafile for each MUP containing individual fibre templates
  - **dir** `Firing-Data` – firing time data files
- **dir** `plotdata` – plot output files

In order to load the resulting data into the DQEMG program, position the data directory at the root of the simulator directory hierarchy and use:

Operator = `run`*ID*
Patient = `patient`
Muscle = `emg`

2.2.4 Output Files

In the muscle directory will be all the output files for the given contraction; in the patient directory will be files used to store the muscle structure. This files are used if the simulator is re-run to generate more than one contraction with the "same" muscle; presumably with the needle at a different position in order to get a different view of the same muscle. If this is desired, note that the user has the option of using the same, or different firing times as discussed in Section 6.

For users examining the `.gst` files, it is worth noting that the control `jitterAccThresh` ("MFP threshold for jitter or MU GST inclusion threshold (kV/s^2)" is used to filter which MUPs are recorded in the `.gst` file. See the discussion below under this control value.

3 Disease Model

A generalized neuropathy and a generalized myopathy may be modelled by the `simtext` program. Both disease models process from a normal muscle state, meaning that the development of a complete normal muscle is simulated before the effects of disease are calculated. This allows muscles to be created with various degrees of progression of a given disease.

The major controls of each disease are presented here; in the discussion of the full set of available controls for the simulator in Section 4, all disease controls will be given.

3.1 Neuropathy

The neuropathic model functions by assuming that an initial population of healthy neurons exists, each of which controls a single motor unit. The model is given a value controlling the fraction of α-motor neurons that will be lost to a simulated disease. This is the major control of the model, and is termed the "Neuropathic MU Loss Fraction."

The disease "progresses" by simulating the death of each fibre in turn, followed by the adoption of the muscle fibres originally controlled by this neuron into nearby motor units. Both the fraction that a motor unit will grow when adopting new fibres, as well as the maximum distance that a neuronal twig is expected to grow may be controlled.

By modelling the disease progression fibre by fibre, the events that occur over time may be simulated. This implies that muscle fibres orphaned "early" in the simulated disease will likely be re-adopted, while fibres whose neuron dies later in the disease are more likely not to have (yet) been adopted by another neuron. Equally, neurons that adopt fibres early in the disease progression are themselves candidates for neuronal loss, so if a large degree of disease involvement is modelled, an individual muscle fibre has a chance of being lost and re-adopted multiple times, as its adoptive neurons succumb to the disease progression.

If fibres are not adopted by any remaining motor unit, they are effectively lost, and will not participate in any contraction as they will receive no control stimulus from a motor neuron. Note also that insertional activity is *not* included in the simulator output.

3.2 Myopathy

As with the neuropathic model, this model includes a progression calculated within a model of the passage of time. Iterations of disease progression are performed until the disease reaches the desired state, controlled by the number of fibres that have become involved in the disease and die. This major control, termed the "Myopathic Fibre Affected Fraction," is most likely the only control required for simple disease simulation.

The model myopathic disease progresses incrementally through a series of "epochs" until the end-condition is met. In each epoch, a very few fibre are infected with the disease. For each new fibre that becomes involved, the fibre is determined to be hypertrophic or hypotrophic based on a defined probability, which may be set by users desiring finer control of the disease simulation.

For all fibres that have become involved throughout the muscle, each epoch sees an appropriate update to the fibre diameter, based on hyper- or hypotrophy. Hypertrophic fibres may split, again based on a controllable probability. Hypotrophic fibres will decrease in radius until they have shrunk below a critical diameter. A "gradually dying" control allows this diameter to be probabilistically different across the population of involved fibres.

The models for these diseases are based on discussion found in the following works:

- Harper (2002)
- Basmajian and de Luca (1985)
- Shefner (2001)
- Stashuk and Brown (2002)
- Nandedkar, Sanders and Stålberg (1983)
- Nandedkar and Sanders (1989)
- Stålberg and Karlsson (2001b)
- Stålberg and Karlsson (2001a)

- Nandedkar and Stålberg (1983a)

An interested reader is further referred to the bibliography at the end of this report for other papers providing a background on muscle simulation and simulated disease involvement.

4 Simulator Control Settings

This section provides the full list of the simulator control values. Each value may be controlled within the simulator interactive interface, or may be controlled by setting a parameter value in the `simulator.cfg` file; either will have the same effect.

For each control, therefore, the config file variable name is provided, followed by the text of the interactive menu option. Config file values will be highlighted in `typewriter text`.

`contractionLevelAsPercentMVC`

contraction level contraction level as a percent of MVC (maximum voluntary contraction) — useful values typically range from 5 to 20% MVC

`nmu_in_mscl`

number of motor units in muscle : 200 is typical for a medium limb muscle, and is the value used for evaluation in Hamilton-Wright and Stashuk (2005). The overall size of the modelled muscle cross section is calculated using this value, therefore increasing or decreasing this number will affect the total muscle size appropriately.

`electrode_type`

electrode type : concentric needle is reported in Hamilton-Wright and Stashuk (2005) though the other needle types are supported:

1. single-fibre
2. concentric (default)
3. monopolar
4. bipolar

Testing and evaluation of the simulator has been primarily performed using the concentric needle setting.

`generateMFPsWithoutInitiation`

generateMFPsWithoutInitiation : flag to control whether to use an older MFP current function generation scheme that does not take into account initiation/extinction events. It is not anticipated that the user will need this option.

Neuropathic Modelling Controls:

`pathology_neuropathy_MU_loss_fraction`

Neuropathic MU Loss Fraction : if set to a value above zero, this will simulate the loss of fibres through nerve death. Fibres lost will be potentially readopted by nearby surviving α-motor neurons, as would be physiologically realistic

`pathology_neuropathy_dist`

Max Adoption Distance In μm : controls the distance (in μm) to which a "nearby" neuron is expected to generate twigs to re-adopt a fibre orphaned by neuronal damage

`pathology_neuropathy_enlargement_fraction`

Neuropathic MU Enlargement Fraction : the fraction by which an adoptive motor unit will grow by adopting new fibres. This is used to balance the adoption model, and operates under the assumption that a surviving neuron is unlikely to adopt a disproportionate number of the remaining fibres, instead sharing the adoption among other candidate neurons based on the size of the motor unit originally associated with each neuron (that is, larger MUs will take on more adoptive fibres than small ones, however the relative proportion of adoption is assumed to be more or less the same).

Myopathic modelling controls

`pathology_myopathy_fibre_affected_fraction`

Myopathic Fibre Affected Fraction : controls the degree of involvement of a myopathy as described above. This is the fraction of muscle fibres that will have become affected by the disease before the time of the modelled contraction.

`pathology_myopathy_percentage_new_involvement`

Myopathic New Involvement Percentage In Each Cycle : describes the rate at which the myopathy affects the muscle; that is, the percentage of fibres add to the disease in each "epoch."

`pathology_myopathy_percentage_affected_dying`

Percentage Of Affected Fibres Dying : the % of fibres *dying* in a given epoch calculated from the total number of diseased fibres in the muscle at a given time.

`pathology_myopathic_fibre_gradually_dying`

Myopathic fibre gradually dying? : this determines whether fibre death is modelled as a sudden, discrete event or whether a diseased fibre will remain active for some time before finally succumbing to the disease.

`pathology_myopathy_death_threshold`

Myopathic Threshold of Fibre Death : threshold of fibre diameter at which fibre death occurs; this is used when fibres are modelled as "gradually dying" and controls the point at which the fibre is deemed too small (*i.e.*, diseased) to be viable.

`pathology_myopathy_hypertrophy_fraction`

Myopathic Fraction of Fibres Becoming Hypertrophic : regulates the hypertrophic tendencies of myopathic degradation; fibres not becoming hypertrophic will become hypotrophic

`pathology_myopathy_hypertrophy_allowed_fraction`

Factor of original area at which hypertrophic fibres split : controls when hypertrophy becomes multiple fibres (that may or may not be hypertrophic themselves; this is determined independently)

`pathology_myopathy_percentage_hypertrophy_split`

Percentage Of Hypertrophic Fibres Splitting : regulates the fraction of fibres for which splitting is possible. Fibres that cannot split cease to hypertrophy at the split threshold.

`pathology_myopathicAtrophyRate`

Myopathic Rate of Atrophy : rate at which hypotrophic fibres decrease in size

`pathology_myopathicHypertrophyRate`

Myopathic Rate of Hypertrophy : rate at which hypertrophic fibres increase in size

`pathology_myopathic_dependent_procedure`

Dying and Splitting Depending on Affection Procedure? : choice between internal algorithms to model fibre loss; users are not expected to wish to use this control.

Needle controls:

`tipUptakeDistance`

tip uptake distance : distance to calculate waveforms from the tip. Shortening this distance will decrease both the time taken to do the calculations as well as the accuracy of the model.

`canUptakeDistance`

cannula uptake distance : as above, but for the cannula

`canPhysicalRadius`

radius of cannula shaft : physical cannula radius, default value of 250μm is appropriate for a typical concentric needle shaft

`cannula_length`

Cannula Length (in mm) : physical cannula dimension modelling depth of insertion, default 10mm

`needle_x_position, needle_y_position`

Needle X,Y Position (in mm) : position within muscle — used with "use last muscle" to model needle reinsertion into the same muscle. This can be used to change the needle location within the same muscle structure; see Section 6.

`needle_z_position`

Needle Z Position from NMJ in mm : This will have an effect on the modelling of endplate potentials, affecting their size, *etc*. Note that the model will not work properly for positions within the NMJ

`needleReferenceSetup`

tip/cannula reference setup : controls which is positive/negative – tip or cannula. Switching this value effectively reverses the polarity of the signal output.

`doJitter`

Enable Jitter? : controls whether or not jitter is calculated to simulate diseases of the NMJ. See Ertaş, Baslo *et al.* (2000) and Stålberg (1995); Stålberg and Trontelj (1997) for details on jitter measurements and utility in detecting disease state.

`jitter`

Jitter (variance) in μs : the variance in the firing times of the NMJ for a given muscle fibre

`jitterAccThresh`

MFP threshold for jitter or MU `.gst` inclusion threshold (kV/s^2) : slope threshold at which jitter need not be calculated, because the fibre is deemed "too far away" to have measurable single-fibre features. Note that if a MUP consists only of fibres that are "too far away", it will itself be deemed "unimportant" and will not be recorded in the `.gst` file produced.

While it may at first seem that all MUPs should be placed in the `.gst` file, consider that a large fraction of the MUPs in a medium to large muscle will be completely undetectable from a given

needle location. If a MUP at the far side of a muscle fires, the voltage at the needle tip may easily be so low as to vanish into noise. If no filter was provided, all MUPs would be labelled in the GST file, even if of zero volts in amplitude.

Our assumption is therefore that if there is insufficient slope in any fibre potential to be interesting, then the associated MUP is therefore also ignorable. A side effect of this choice is that large distant MUPs may produce unlabelled potentials in the output; a user wishing to have these labelled is encouraged to lower this control value as appropriate.

`minimumMuscleMetricThreshold`

Minimum metric to seek needle to : in a clinical environment, an experienced user does not simply collect data at random locations in the muscle; instead, "good" locations are found by moving the needle until at least one or two high-acceleration fibres are found. Similarly, the simulator will attempt to "seek out" a good location for the needle – this value provides a good balance between finding a good location and discarding too many acquisition sites

`filter_raw_signal`

Bandpass filter raw signal? : controls whether bandpass filtering is performed; defaults to true as would be performed by a clinical signal processing device

`use_noise`

Generate noise? : controls whether to include synthetic noise to provide a realistic baseline as would be present in a signal acquired in a lab

`signalToNoiseRatio`

S/N ratio : the signal-to-noise ratio for noise to be generated if selected above. The default of 25dB generates reasonable noise

`operator_name`

Recorded Operator Name : name of clinician to place in DQEmgData output file format

`patient_name`

Recorded Patient Name : name of patient to place in DQEmgData output file format

`muscle_name`

Recorded Muscle Name : name of muscle to place in DQEmgData output file format

`patient_id`

Patient ID : DQEmgData ID value

`muscle_side`

Laterality : muscle side

`firing_maximumFiringThreshold`

maximum recruitment threshold : used to determine slope of recruitment curve; at this percentage of MVC, all motor units will be firing and subsequent increase in force will occur only through firing rate changes.

`emg_elapsed_time`

total time for EMG generation : total length of time contraction will be simulated to take, default 30s

`maxShortVoltage`

max (scaled) value in 16-bit output : used to calculate the $<$compression$/>$ in the output data file

`mscl_fib_dens`

muscle fibre density : muscle fibres per square mm: default 10

`mscl_area_per_fib`

area of 1 muscle fibre : in mm: default 2.5×10^{-3}mm

`min_mu_diam`

min motor unit diameter : minimum diameter used for MU simulation: default 2mm

`max_mu_diam`

max motor unit diameter : diameter of the largest MU: default 8mm

`firing_recruitmentSlope`

IPI firing slope : slope of firing recuitment curve: default 0.8, see Fuglevand (1989); Henneman and Olson (1965); Henneman, Somjen and Carpenter (1965a,b)

`firing_minimumFiringRate`

min firing rate : firing rate for a just-recruited MU: default 8pps (Fuglevand, 1989)

`firing_maximumFiringRate`

max firing rate : max rate at which a MU will fire: default 42pps (Fuglevand, 1989)

`coefficientOfVarianceInFiringTimes`

coeff of variance : in firing times modelled as a Poisson point process: default 0.25

`mu_layout_type`

MU Layout Type : layout algorithm:

1. random
2. grid based

The default is "grid based", as described in Hamilton-Wright and Stashuk (2005)

`generate_second_channel`

Generate Second Channel? : generate surface as undersampled copy of needle, or don't include second channel at all?

In the current implementation, there is no useful surface data created. This option exists in order to create some type of correlated data in the macro channel for some experimental tools.

Dump MFP Peak-to-Peak Values? : this option creates debugging output in the patient directory.

5 Data Files Contents

The `micro.dat` and `macro.dat` files standards are based on the data widths and formats of 32-bit Intel x86 architecture. The following sizes are therefore used:

short 16 bit "little endian" integer

long 32 bit "little endian" integer

5.1 EMG {micro,macro}.dat files:

The file format is as follows:

<**EmgFile**>

<**EmgHeader**> : global values for EMG interpretation

<**channel: short**/ > : unused

<**HP cutoff: short**/ > : unused, currently 5000

<**LP cutoff: short**/ > : unused, currently 500

<**scale: short**/ > : used to calculate data values, in conjunction with <compression/ >

<**sampling rate: long**/ > : rate in samples/second at which data was digitized

<**number of samples: long**/ > : total number of samples in data file

<**elapsed time: long**/ > : time elapsed, in sampling units. This is the same as time.

<**compression: short**/ > : used in conjunction with scale. When read in all data values should be multiplied by:

$$\frac{< \text{scale}/ >}{< \text{compression}/ >}$$

< /**EmgHeader**> : This header is therefore 22 bytes long

<**EmgData**> : actual data "samples"

<**data point : N * short**/ > : This is the EMG data. There will be N shorts in this section, where N is equal to <number of samples/ > in the above header.

< /**EmgData**> :

< /**EmgFile**> : end of file

5.2 MUP dco/gst files

Both the .dco and .gst files have the same format. The only difference is the confidence the user has in the quality of the data in the file: .dco files are generated from a decomposition algorithm, while .gst files are "Gold Standard" .dco files produced by the simulator (or some other authority) to reflect "complete" knowledge about the signal.

In the case of simulated data files, the .gst files record the generating MUP for every firing recorded in the file. It is not expected that a decomposition algorithm will be able to achieve this standard as the simulator has perfect information about superpositions, however an attempt is made by the simulator to avoid recording data for MUPs that are "too distant" (again, see Hamilton-Wright and Stashuk (2005)).

<**DcoFile**> :

<**DcoHeader**> :

<**Name : 60 * char**/ > : name tag from generator of file.

<**Num Trains : short**/ > : Number of MUP trains in this file

<**Num MUPs : short**/ > : Number of MUP structures in total in this file

< /**DcoHeader**> :

<**MUP :** <**N**>> : There will be <N> MUP structures in the remainder of the file, where <N> is equal to Num MUP in the header

<**firing time : float**/ > : time in samples when MUP fired

<**buffer offset : long**/ > : offset in samples when MUP fired (same as time)

<**motor unit : short**/ > : the id of the motor unit (train) which this MUP belongs to

<**MUP number : short**/ > : monotonically increasing id of the MUP in the file. The first ID is 1.

<**uncertainty : float**/ > : the certainty with which we believe the result. Normalized $[0.0 \ldots 1.0]$

< /**MUP**> :

< /**DcoFile**> :

"Train 0" is used to record all MUPs that cannot be associated with any other train. When dealing with real data, this "train" will contain all MUPs that are unrecognizable by the decomposition algorithm or the gold-standard creating authority. As the simulator has complete knowledge of the origin of each and every MUP, it is therefore possible that there would be no MUPs in this train.

Historically, this has created problems with other software packages, so in order that train zero always has at least one MUP in it, the simulator inserts a placeholder MUP into the `.gst` file in train 0 at offset 0. This placeholder does not mark a simulated spike, and should therefore be ignored.

5.3 MUP files:

MUP files will appear in the directory:

c:\simulator\data\sim*\<MuscleName/ >\tmp-mmuaps

and they have the following form:

<**MUPFile**> :

<**MUPHeader**> :

<**NumMUPs : long**/ > : Number of MUPs in this file. There will be more than one MUP in cases of needle movement. Later there will be many, because of Jitter.

<**Length of MUP : long**/ > : Number of data elements in a MUP.

< /**MUPHeader**> :

<**MUP :** <**N**/ >> : There will be N MUPs in the remainder of the file, where N is equal to Num MUPs in the header

<**data point :** <**N**/ > *** float**> : This is the MUP data. There will be M floats for each MUP, where M is the Length of MUP, above.

< /**MUP**> :

< /**MUPFile**> :

6 Data Generation

The simtext program can be used to generate various types of related contractions in order to answer questions about muscle structure. To explore the simulator capabilities, an example session is provided here.

Much of the simulator's behaviour may be controlled through its configuration file simulator.cfg; to get a copy of the default configuration file, simply run the simulator with all default settings. This will produce a simulator.cfg file in the platform-specific location described in Section 2.2. This file can then be used as a template to change any settings desired and the simulator may then be run with the option -skip-confirm, which bypasses the user interface as described in Section 2.1.2. Alternatively, the values changed in the example here may be changed one at a time in the simulator user interface.

6.1 Begin a new study

To begin a new study, simply omit the flag -useLastMuscle; all controls provided will then be used to generate a fresh muscle. Note that this is the only time at which the level of disease involvement may be changed – to model two different levels of involvement, two different studies must be produced.

Once the simulator has run to completion, all the structural information defining the muscle will be placed in the operator/patient directory, and in the operator/patient/muscle directory for the run, micro1.dat and micro1.gst files will be produced. These characterize the signal acquired from this first contraction with this muscle structure.

6.2 Looking Around in the Muscle

After the initial contraction is performed, the user may wish to know what the signal would look like with a different needle location (deeper in the muscle, shallower, *etc*.). To perform this analysis, simply change the needle location (either in the config file or through the user interface) and re-run.

6.2.1 Using the Same Firing Times

If the option -useOldFiringTimes is used, then *the same contraction* will be performed with exactly the same neuronal-level firing time information as in the previous contraction.[1] All fibre contributions will be recalculated in order to determine the contributions of the muscle fibres relative to the new needle location automatically; no user control for this is required other than the indication of a new needle location (in x, y and/or z).

This is expected to be useful to an investigator who wishes to see the differing contributions from various motor units as the needle is moved through the muscle. By preserving the firing times from contraction to contraction, the shape of individual MUPs may be examined, as they will appear at exactly the same time offset in the different contraction files, even though their shape may change dramatically.

6.2.2 Using Different Firing Times

If the flag -useOldFiringTimes is omitted, a new contraction with the same muscle (but different firing times) is produced, as would be the case if an investigator asked a real patient to contract their muscle a second time.

[1] Note that jitter information is not saved from contraction to contraction and will be regenerated.

6.2.3 Variability of Jitter

By leaving the needle location the same and adjusting the `jitter` control, the effects of jitter may be explored. This is particularly powerful in conjunction with the use of the same firing information as again individual MUPs (and MUP trains) may be compared across contractions.

In this case, subsequent runs of the simulator will be quite fast, as if the needle location remains the same, the fibre contributions do not need to be recalculated.

7 Summary

The `simtext` muscle simulator provides users a tool for the construction and analysis of EMG data comparable to that acquired from medium to large limb muscles.

By using the facility to re-use structural and timing data from earlier contractions, the `simtext` program allows an investigator to explore the same muscle morphology from different locations.

References

AAEM '03. *General Meeting of the AAEM 2003*. AAEM '03, AAEM, San Francisco, 2003.

Andreassen, S. and A. Rosenfalck. Relationship of intracellular and extracellular action potential of skeletal muscle fibre. *Critical Reviews in Bioengineering*, 6:267–306, 1983. CRC Press Inc.

Basmajian, J. V. and C. J. de Luca. *Muscles Alive: Their Functions Revealed by Electromyography*. William & Wilkins, Baltimore, fifth edition, 1985.

Brown, W. F., C. F. Bolton and M. J. Aminoff, editors. *Neuromuscular Function and Disease*. W.B. Saunders, Philadelphia, 2002.

Buchthal, F., C. Guld and P. Rosenfalck. Motor unit territory in different human muscles. *Acta. Physiol. Scand.*, 45:72–87, 1959.

Day, S. J. and M. Hulliger. Experimental simulation of cat electromyogram: Evidence for algebraic summation of motor-unit action-potential trains. *Journal of Neurophysiology*, 86(5):2144–2158, November 2001.

de Luca, C. J., P. J. Foley and Z. Erim. Motor unit control properties in constant-force isometric contractions. *Journal of Neurophysiology*, 76(3):1503–1516, September 1996.

Duchêne, J. and J.-Y. Hogrel. A model of EMG generation. *IEEE Transactions on Biomedical Engineering*, 47(2):192–200, February 2000.

Ertaş, M., M. B. Baslo *et al.* Concentric needle electrode for neuromuscular jitter analysis. *Muscle & Nerve*, 23:715–719, May 2000.

Farina, D., R. Colombo *et al.* Evaluation of intra-muscular EMG signal decomposition algorithms. *Journal of Electromyography and Kinesiology*, 11(3):175–187, June 2001a.

Farina, D., A. Crosetti and R. Merletti. A model for the generation of synthetic intramuscular EMG signals to test decomposition algorithms. *IEEE Transactions on Biomedical Engineering*, 48(1):66–77, January 2001b.

Finsterer, J. EMG-interference pattern analysis. *Journal of Electromyography and Kinesiology*, 11:231–246, 2001.

Fuglevand, A. J. *A motor unit pool model: Relationship of Neural Control Properties to Isometric Muscle Tension and the Electromyogram*. Ph.D. thesis, University of Waterloo, 1989.

Greip, P. A. M., K. L. Boon and D. F. Stegman. A study of the motor unit action potential by means of computer simulation. *Biological Cybernetics*, 30:221–230, 1978.

Hamilton-Wright, A. and D. W. Stashuk. Physiologically based simulation of clinical EMG signals. *IEEE Transactions on Biomedical Engineering*, 52(2):171–183, February 2005.

Hamilton-Wright, A., D. W. Stashuk and T. Doherty. Using a muscle model to simulate EMG signals. In *General Meeting of the AAEM 2002*. AAEM '02, AAEM, Toronto, 2002.

——. Clinical EMG signal simulation based on physiological muscle modeling. In AAEM '03.

——. Simulation of disease effects on muscle structure, activation and acquired EMG. In AAEM '03.

Harper, C. M. Neuromuscular function and disease. In Brown, Bolton and Aminoff (2002), chapter Myopathies.

Henneman, E. and C. B. Olson. Relations between structure and function in the design of skeletal muscles. *Journal of Neurophysiology*, 28(3):581–598, May 1965.

Henneman, E., G. Somjen and D. O. Carpenter. Excitability and inhibitability of motoneurons of different sizes. *Journal of Neurophysiology*, 28(3):599–620, May 1965a.

——. Functional significance of cell size in spinal motoneurons. *Journal of Neurophysiology*, 28(3):560–580, May 1965b.

Nandedkar, S. and E. Stålberg. Simulation of macro EMG motor unit potentials. *Electroencephalography and Clinical Neurophysiology*, 56:52–62, 1983a.

Nandedkar, S. D. and D. B. Sanders. Simulation of myopathic motor unit action potentials. *Muscle & Nerve*, 12:197–202, 1989.

Nandedkar, S. D., D. B. Sanders and E. V. Stålberg. EMG of reinnervated motor units: A simulation study. *Electroencephalography and Clinical Neurophysiology*, 70:177–184, 1983.

——. Automatic analysis of the electromyographic interference pattern. *Muscle & Nerve*, 9:431–439, 1986a. Part I: Development of Quantitative Features.

——. Automatic analysis of the electromyographic interference pattern. *Muscle & Nerve*, 9:491–500, 1986b. Part I: Findings in Control Subjects and in Some Neuromuscular Diseases.

——. Simulation and analysis of electromyographic interference pattern in normal muscle. *Muscle & Nerve*, 9:423–430, 1986c. Part I: Turns and Amplitude Measurements.

——. Simulation and analysis of electromyographic interference pattern in normal muscle. *Muscle & Nerve*, 9:486–491, 1986d. Part II: Activity, Upper Centile Amplitude, and Number of Small Segments.

Nandedkar, S. D., D. B. Sanders *et al.* Simulation of concentric needle EMG motor unit action potentials. *Muscle & Nerve*, 11:151–159, February 1988.

Nandedkar, S. D. and E. V. Stålberg. Simulation of single muscle fibre action potentials. *Medical & Biological Engineering & Computing*, 21:158–165, March 1983b.

Nandedkar, S. D., E. V. Stålberg and D. B. Sanders. Simulation techiques in electromyography. *IEEE Transactions on Biomedical Engineering*, BME-32(10):775–785, October 1985.

Sanders, D. B. and E. V. Stålberg. AAEM minimicrograph #25: Single-fibre electromyography. *Muscle & Nerve*, 19:1069–1083, September 1996.

Shefner, J. M. Motor unit number estimation in human neurological diseases and animal models. *Clinical Neurophysiology*, 112(6):955–964, June 2001.

Stålberg, E., S. Andreassen *et al.* Quantitative analysis of individual motor unit potentials: a proposition for standardized terminology and criteria for measurement. *Clinical Neurophysiology*, 3(4):313–348, 1986.

Stålberg, E. and L. Antoni. The electrophysiological cross section of a motor unit. *J Neurol. Neurosurg. Psychiatry*, 43:469–474, 1980.

Stålberg, E. and B. Falck. The role of electromyography in neurology. *Electroencephalography and Clinical Neurophysiology*, 103:579–598, 1997.

Stålberg, E. and L. Karlsson. Simulation of EMG in pathological situations. *Clinical Neurophysiology*, 112(6):869–878, June 2001a.

——. Simulation of the normal concentric needle electromyogram by using a muscle model. *Clinical Neurophysiology*, 112(3):464–471, March 2001b.

Stålberg, E. V. Single fibre EMG, macro EMG, and scanning EMG. new ways of looking at the motor unit. *Critical Reviews in Neuroscience*, 2(2):125–167, 1986.

——. Single fibre EMG – an update. *Electroencephalography and Clinical Neurophysiology*, 97(4):S63, September 1995.

Stålberg, E. V. and J. V. Trontelj. The study of normal and abnormal neuromuscular transmission with single fibre electromyography. *Journal of Neuroscience Methods*, 74(2):145–154, June 1997.

Stashuk, D. W. Simulation of electromyographic signals. *Journal of Electromyography and Kinesiology*, 3(3):157–173, 1993.

——. Decomposition and quantitative analysis of clinical electromyographic signals. *Medical Engineering and Physics*, 21(6):389–404, February 1999.

——. EMG signal decomposition: How can it be accomplished and used? *Journal of Electromyography and Kinesiology*, 11(3):151–173, June 2001.

Stashuk, D. W. and W. F. Brown. Neuromuscular function and disease. In Brown *et al.* (2002), chapter Quantitative Electromyography.

Stegeman, D. F., T. H. Gootzen *et al.* Intramuscular potential changes caused by the presence of the recording EMG needle electrode. *Electroencephalography and Clinical Neurophysiology*, 93:81–90, 1994.

Suojanen, M., S. Andreassen and K. Olesen. The EMG diagnosis — an interpretation based on partial information. *Medical Engineering and Physics*, 21(6–7):517–523, Jul–Sep 1999.