CoSMo 2012: Neuromechanics and Spinal Cord
Muscle Redundancy Matlab Exercise
Jason Kutch, Ph.D. (kutch@usc.edu)

Contents

1 Introduction

In the morning lectures, we learned that the nervous system can select from many different muscle coordination patterns to achieve the same net joint torque. This phenomenon is referred to as *muscle redundancy*. We also learned, however, that the set of coordination pattern options available to the nervous system is not completely wide open. Constraints on coordination pattern selection could play an important role in understanding neuromuscular disorders.

In this Matlab exercise, we will learn how to analyze muscle redundancy using computational geometry, Monte-Carlo methods, as well as optimization. **Note: These exercises require the MATLAB Optimization Toolbox to run.**

2 Muscle Redundancy: The nervous system requests biomechanical options

Before understanding what the nervous system does, it is important to understand what it *has to do* and what it *could have done*. In the context of muscle redundancy, this boils down to specifying the constraints of the mechanical task, and then describing the complete space of coordination patterns that could have met the task constraints. In this section, we will work out examples of several different approaches to this problem.

Typically, the action of multiple muscles is described by a linear mapping between muscle force and joint torque. The matrix that performs this transformation is called the *moment arm matrix*, usually

denoted **R**. The muscle forces are described in a vector $f = [f_1, f_2, , \ldots, f_m]^T$ where m is the number of muscles. The overall torque in an isometric (no movement) task is then

$$\tau = \mathbf{R}f \tag{1}$$

Interestingly, there is a lot to understanding muscle function in this seemingly simple equation.

2.1 Computational geometry

Linear maps, such as Equation (1), have the property that they map convex sets to convex sets. A convex set is a set of points for which any two points in the set are connected by a straight line. The convexity property allows us to exactly describe all coordination patterns that would produce a given joint torque.

Open the Databaser, and select Load Config in the User menu. Select CoSMo2012_Example1.mat. In "Refine results, Model, and Publish", select the function "Step1_TwoMuscleVertexEnumeration", and press the "Publish" button. This will prompt you for a desired torque, and launch a result figure. You will see an xy plot of force in 1 muscle against force in a second muscle. The red box represents all permissible muscle force combinations if there were no constraints of the task. The blue points represent the vertices of the intersection of the red box with the constraints imposed by the task. The line connecting the two points has been called the "task-specific activation set", and it shows all muscle coordination patterns that would produce the same net joint torque. Notice that there are an *infinite* number of such muscle coordination patterns, but they are defined by a *finite* number of vertices.

You can use "Step1a_ChangeTwoMuscleModel" to modify the parameters of the two muscle model, which are the maximum muscle forces and the moment arms of the the muscles about the joint.

> **Exercise 1:** Make the two muscle *agonists*, meaning that they produce torque in the same direction. Vary the desired torque and explain graphically whether a muscle must be used for a task (it is necessary) or whether the task could be achieved without the muscle (it is redundant).

"Step1_TwoMuscleVertexEnumeration." has been carefully documented for your convenience. Look at it carefully and understand how it works. Everything that has been shown here works for an arbitrary number of muscles, but the number of vertices of the task-specific activation set grows exponentially in the number of muscles, making this analytical solution infeasible by about 14 or so muscles. Fortunately, there is a numerical work-around (see the next section).

2.2 Feasible coordination patterns using Monte-Carlo

Another approach that can be used when the number of muscles in your model is very large is to use Monte-Carlo sampling with "smart" rejection. This procedure is compared with the analytical method using the function "Step2_TwoMuscleMonteCarlo.m". Running this function will give you white crosses along the subspace derived analytically by vertex enumeration.

> **Exercise 2:** Play with the number of desired coordination patterns, and determine if they uniformly cover the analytically-derived subspace (i.e. does the numerical method give you an adequate representation of the underlying subspace. Bonus: how does the Monte-Carlo method do when the number of muscles, i.e. the dimension of the problem, increases

2.3 More Information

Further reading on this topic can be found in these papers, as well as the references therein:

1. Kutch JJ, and Valero-Cuevas FJ. Muscle redundancy does not imply robustness to muscle dysfunction. J Biomech 44: 1264-1270, 2011. Click for PDF

2. Kutch JJ, and Valero-Cuevas FJ. Challenges and New Approaches to Proving the Existence of Muscle Synergies of Neural Origin. PLoS Comput Biol 8: e1002434, 2012. Click for PDF

3. Bunderson NE, Burkholder TJ, and Ting LH. Reduction of neuromuscular redundancy for postural force generation using an intrinsic stability criterion. J Biomech 41: 1537-1544, 2008. Link to Article

3 Muscle Redundancy: The nervous system would like to know what's best

Now that we have shown how to derive all possible solutions that the nervous system could use, it is appropriate to ask what coordination pattern the nervous system would pick given different cost functions. Let's stay with the example of two agonists and see what would be best given different cost functions. The important thing to understand here is that cost functions that superficially seem like they would favor the same coordination pattern actually make different predictions. For a discussion of cost functions for predicting muscle coordination patterns, see (Buchanan TS, and Shreeve DA. An evaluation of optimization techniques for the prediction of muscle activation patterns during isometric tasks. J Biomech Eng-Trans ASME 118: 565-574, 1996. Click for PDF)

3.1 Minimizing the total amount of muscle force

You might first guess that the nervous system would want to exert as little force as possible. This could be formulated as the minimization of a cost function:

$$\min_f \sum_{i=1}^{m} f_i \tag{2}$$

Exercise 3: Use the function "Step3_TwoMuscleCostFunction" to explore the effect of minimizing the sum of muscle forces. Explore the effect of changing the model parameters, including the moment arms and the maximum muscle forces.

Well, you are likely to find that force will be shared between the two muscles if all things are equal (moment arms and max muscle forces), but as soon as anything is unbalanced, this cost function favors 1 muscle dominating.

3.2 Minimizing the total amount of squared muscle force

You could also guess that the nervous system would want to exert as little energy as possible. If you stretch an elastic mechanical system, like a spring, the force exerted by that spring is $f = -k\Delta x$, but the energy stored in that spring is $E = \frac{1}{2}k(\Delta x)^2$. Therefore, there will be a quadratic relation between force and energy:

$$E = \frac{1}{2}k\frac{f^2}{k^2} = \frac{1}{2k}f^2 \tag{3}$$

Therefore, you could imagine that minimizing total muscle energy consumed in a contraction would be equivalent to minimizing the following cost function:

$$\min_f \sum_{i=1}^{m} f_i^2 \tag{4}$$

Exercise 4: Use the function "Step3_TwoMuscleCostFunction" to explore the effect of minimizing the sum of muscle forces compared to minimizing the sum of muscle forces squared, especially when there are asymmetries in the system like unequal moment arms or maximum muscle forces.

4 Muscle Redundancy: When would a muscle turn off?

In this section, we will explore a schematic explanation for the results of Kouzaki and Shinohara, among others, on the switching of muscles during sustained isometric contraction. Open the Databaser and use the function "Step4_MuscleSwitching.m". You can change the parameters of the base model using "Step1a_ChangeTwoMuscleModel.m". Edit the parameters of muscle fatigue and recovery by opening the function "Step4_MuscleSwitching" (right click on it in the list and select Edit Publishing Method) and find the section "EDIT AS NEEDED".

By running the function with the default parameters, you should find that activity oscillates back and forth between muscle 1 and muscle 2 if the target torque is in a medium to low range: too low and the muscles never fatigue, and too high the muscles can not recover fast enough.

Double click. Quicktime and Adobe Reader req.

Exercise 5: (Challenging). In order to make the oscillation work, I needed to add a "cross-coupling" between the two muscles. One muscle checks to make sure that the other is not recharging before it allows itself a break. Edit the code to scan through sets of parameters to determine if this cross-coupling is necessary for sustained activity oscillation between the two muscles.

5 Motor unit redundancy: Introduction to motor unit force generation

Motor units generate force by the temporal summation of *twitches*. The twitch is the temporal force profile that occurs within a single motor unit subsequent to stimulation of the muscle fibers by a single spike in the motoneuron.

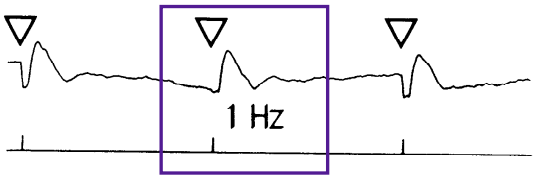

Figure 1: Single twitches, elicited by repeated stimulation of a single motor axon in human muscle (Thomas CK, Bigland-Richie B, and Johansson RS. Force-frequency relationships of human thenar motor units. J Neurophysiol 65: 1509-1516, 1991. Link to Article)

When the motor neuron firing rate increases to the point where force does not decay from the previous spike when the current spike arrives, temporal summation of twitches occurs and force begins to build in the motor unit. Forces among motor units sum to give the total muscle force.

The standard model for the recruitment of motor units within a muscle is called the Fuglevand Model. This model is described in Fuglevand AJ, Winter DA, and Patla AE. Models of recruitment and rate coding organization in motor-unit pools. J Neurophysiol 70: 2470-2488, 1993 Link to Article. This model assumes that the firing rates of various motor units are coupled by a single descending drive signal, which gives rise to the standard "Size Principle" of motor unit recruitment. We will play with this model and some reduced models that predict dynamics of motor unit recruitment during sustained contraction.

Open the Databaser, and select "Choose Class" in the "Refine Results ..." section. Choose the folder "CoSMo2012Neuromech_MotorUnit". Click the "+" under "Result(s) and/or Model(s)", and select "ResultsAndModels/Example2_MotorUnit/FuglevandParameters.mat". This Matlab datafile contains most of the parameters from the original Fuglevand paper. Select "Step1_FuglevandModel.m" and and click "Publish!". The program will run the Fuglevand muscle force model for as many different levels of excitation as possible, and produce a plot showing the firing rate of each motor unit as a function of excitation. The Fuglevand model also predicts a detailed time series of muscle force, motor unit spike times, and EMG, which we will explore later.

Exercise 6: (Best done on your own time). Explore the effect of changing various parameters of the Fuglevand model. You will have to open the "FuglevandParameters.mat" file, edit, and then save to a file of your choice and then load that file in the Databaser. Pay careful attention to the "Recruitment Threshold Excitation (RTE)" parameter, which can change a muscle from "rate coded" to "recruitment coded").

6 Motor unit redundancy: Optimizing motor unit activation and ensuing oscillations

We want to explore what happens when the motor units are potentially uncoupled, and the nervous system selects motor unit firing rates through an optimization procedure. As we discussed in class, this is a very hard computational problem called mixed integer nonlinear programming (MINLP). We can solve this problem, but not for 120 motor units. So, were going to do it for 5 motor units, which you should actually think of a 5 groups of 24 motor units, each doing roughly the same thing.

6.1 Making a motor unit pool model

First, we can make a motor unit pool model. This is accomplished using the function "Step2_MakeMotorUnitPool.m". When you run this function, it will ask how many motor units you want in the model (5), what the range of peak force is from the smallest motor unit to the strongest motor unit (10), and will ask the peak firing rate (30 Hz) of all the units (assumed to be the same in this simple model. It will then ask you to save these parameters. You should then use the "+" under "Result(s) and/or Model(s)" to add the file you just created.

6.2 Dynamic MINLP Optimization

Now, having the .mat file that you created selected, select the function "Step2_MotorUnitOrderlyRecruitment.m" and press Publish. A movie will automatically start in Figure 1. The point of this simulation is to ramp up muscle force over time, and at each time step, solve the MINLP problem to predict the best combination of motor units for the amount of muscle force. The firing rates of the motor units will be on top and the overall muscle force is on the bottom.

Exercise 7: Re-engineering the motor unit model using "Step2_MakeMotorUnitPool.m" and re-run the "Step3_MotorUnitOrderlyRecruitment" function. See how the choice of motor unit properties will effect their recruitment.

You can also use the MINLP approach to motor units to explore dynamic recruitment during sustained contractions at the same level of force. This is what "Step4_MotorUnitRotation.m" does. It also takes your motor unit model .mat file as input, and will generate a movie of the motor unit firing rates, overall force, and energetic cost.

Exercise 8: Using "Step4_MotorUnitRotation.m", start at 5% maximum, see what happens, and then increase the force gradually. You should see some wild things.

Here we have been using a very simple model of fatigue; there are actually many ways to model fatigue. For more complex models, please see (Dideriksen JL, Farina D, and Enoka RM. Influence of fatigue on the simulated relation between the amplitude of the surface electromyogram and muscle force. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 368: 2765-2781, 2010.Link to Article).

6.3 Experimental predictions

We have used this simple model to show that we would expect oscillations in motor unit activity and metabolic expenditure during a sustained contraction when multiple redundant motor units are available for rotation. In the lecture, I demonstrated that spike-triggered averaging (STA) shows oscillation in twitch

force magnitude during a sustained contraction in the first dorsal interosseous (FDI), suggesting motor unit rotation. Someone might ask whether these oscillations would be expected to emerge from STA during a sustained contraction even if there were no motor unit rotation. Here is a good opportunity to use a simple model to address this point, and illustrates integration in the Databaser environment between modeling and data analysis.

Load and select the file "FuglevandParameters.mat" as above. Now run "Step5_FuglevandModelTimeSeries.m". It should take a few seconds to run, and the program status bar will indicate when it is finished. The program has run the Fuglevand model and saved the data in exactly the same format as I saved the experimental data, so the exact same analysis code can be used on both. We will load this simulated data in just as if it were real experimental data.

From the menubar, select "Experiment ... Open Experiment". Then choose "Experiments ... MotorUnitRotation". Next, from the menubar select "Session ... Open Session". Then choose "P001S001". In the "Analyze" section of the program (lower left), select the "+" under "Trials to Analyze". Under "...Using function" select "Choose Class" and then "Muscle Switching". Run "Step0_PlotForceAndEmg" by either clicking the "Analyze" button or double clicking "Step0_PlotForceAndEmg". This will just give you a picture of the simulated force and EMG. Next, run "Step1_StaOverTime". This function will bring up a plot of EMG first. The figure window prompts you to make two clicks. The first should have the vertical line at the beginning of what you consider the steady portion of the trial (trivial in this case because the model is completely steady). The horizontal line of the first click should be at the level that you consider noise. The second click should have the vertical line at the end of what you consider the steady portion, and the horizontal line at the second click is not used. Once you have made these clicks, the status indicator will show that it is processing, and when it is finished, you will see plots of spike amplitude as a function of time, and spike triggered average as a function of time. Notice that under these conditions of no fatigue and no motor unit rotation, the STA is not expected to vary in time.