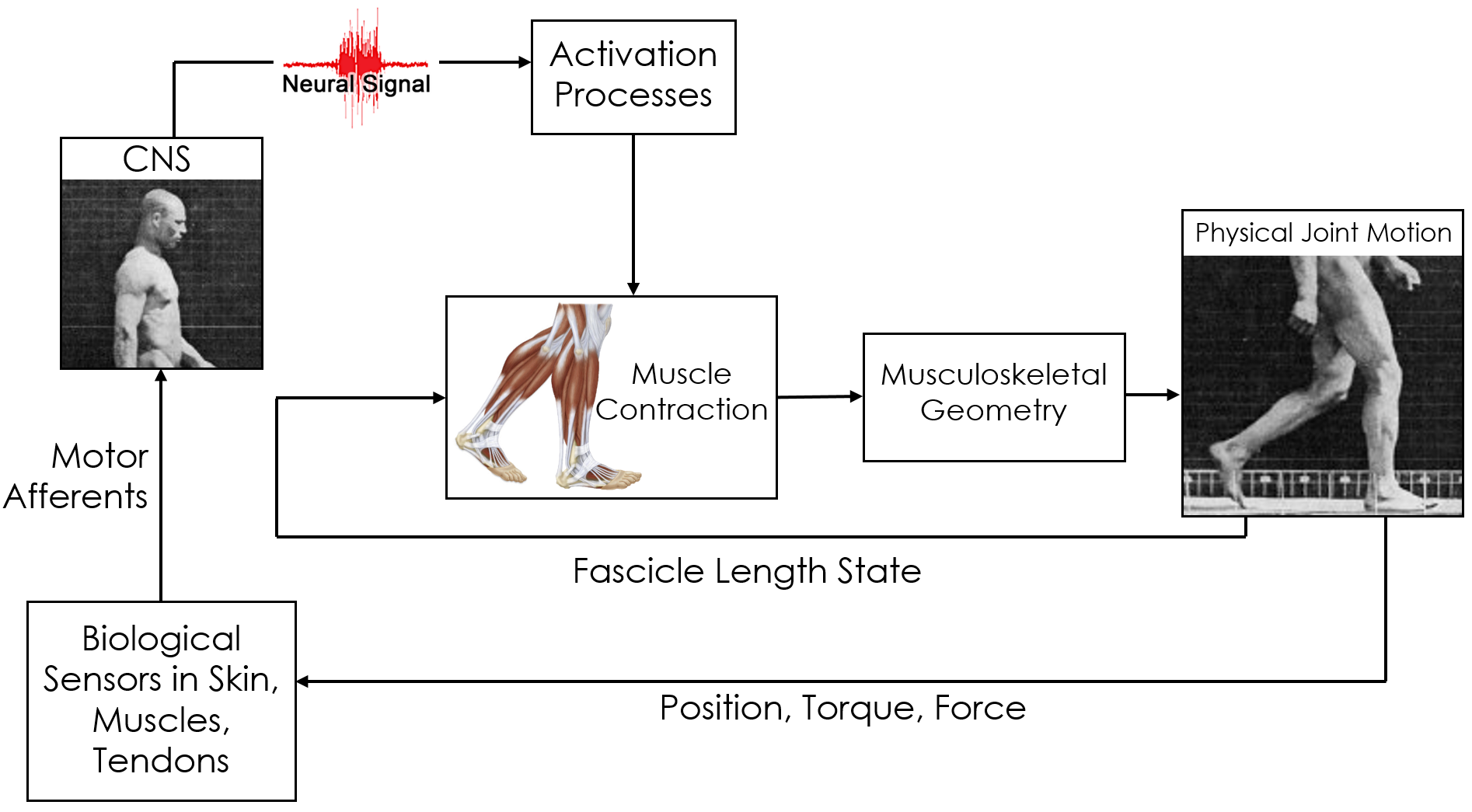
# MATLAB Problem 2

A quick note before you start – you will answer all questions for this problem within this Word document. When finished, please save the document as a .pdf, which you will upload directly to Gradescope.

The field of neuromusculoskeletal biomechanics uses mathematical models to understand how neural commands are converted into movement and locomotion. These models allow us to predict, with impressive fidelity, muscle forces, joint moments, and joint angles from neural excitations and external loads. There are several steps to these models, as laid out in the following diagram:



In this problem, we’re going to focus on the box called “Activation Processes”, which uses differential equations to describe how neural excitation in the form of action potentials translates to muscle activation and force production. If you’re interested in the underlying physiology or the full model, check out the manuscript PDF included with the problem set on CCLE. I’m admittedly biased, but my mind was blown open when I first read about these models in their entirety.

## 2a: Build a mathematical model and analytically solve the resulting differential equation.

Activation dynamics describe the chemical processes of calcium release and uptake within muscle tissue, as well as the physiological formation of actin-myosin cross bridges that are responsible for muscle force production. These processes can be described by the following differential equation:

Where is muscle activation, is neural excitation, and is the activation time constant.

Let’s consider an admittedly contrived scenario where neural excitation is some function of muscle activation – the physiological implications of this would be a reflex-type scenario, where a currently-activated muscle is given more neural excitation in a positive feedback loop, with some sinusoidal modulator on top (don’t worry if that sounds like jargon, I’m just defending the formulation of the problem for anyone who knows muscle physiology). We can model this scenario as:

We will also define and set the initial conditions . Important note: don’t worry about units here - both activation and excitation are normalized to be unitless.

Your first task is to **classify this differential equation, identify an analytical solution method, and find a specific solution analytically. You can do this by hand, then take a picture and insert it into this document. Show your classification and your work, and make sure that your work is readable. Important note – if you’ve done this correctly, you will find that your specific solution is better left in implicit form. Please do not try to solve this algebraically!**

**Tip:** It’s really important that you get the correct value here for your constant of integration. It is probably worth checking with a friend or one of the TAs to avoid a calculator mistake that would make the rest of the problem very difficult.

[YOUR ANSWER HERE]

## 2b: Use numerical methods to solve your implicit equation for .

Download Homework2\_MatlabProbs.m, muscleActODEfun.m, and eulerSolver.m. Without changing any of the file names for these three files, put all three files into a single folder. Navigate to that folder in Matlab’s “current folder” pane.

Open all three files in MATLAB. In problem 2b, you’re going to use this code to solve the implicit equation you found in 2a for . Throughout the code, you’ll see double asterisks (“\*\*”) – this indicates places where you need to fill in missing code snippets. **Note:** if you run this right away, you will get errors, because the code is not complete. That’s where you come in!

The first step in this process is to prepare our implicit equation for entry into MATLAB. To do this, rearrange your implicit solution in part 2a to take the form:

This will allow us to use a built-in function in MATLAB called fzero() to find solutions to this equation at different values of . To make this happen, follow along in the code for problem 2b.

**Save the resultant figure as a .png of .jpg (don’t just screenshot it), and insert the image into this document.**

[INSERT FIGURE 1 HERE]

**What do the two curves in this plot represent? How do they relate to each other?**

In this problem, we used analytical tools to solve the differential equation, and then numerical tools for just the algebra. **Compared to a fully numerical approach like Euler’s method, how accurate do you think this mixed approach is? Explain your answer.**

## 2c. Use Euler’s method to solve your differential equation.

We are now going to build our first numerical solver (WOOHOO!) and use it to solve this differential equation numerically. Follow along in the code for problem 2c as we set this up.

First, fill in values for the initial conditions, time range of interest, and step size. For this first pass, we will use the same initial conditions as above, with a time range of 0 to 2s, and a step size of 0.005 seconds.

Next, go to muscleActODEfun.m and code up your differential equation. You should all be experts at this by now, from Pset 1.

When that’s done, we need to build our solver. Open eulerSolver.m and fill in the missing code.

Now back to Homework2\_MatlabProbs.m. Split your code after figure 2 finishes plotting (around line 86). Run the top half, and drop your figure .png here:

[INSERT FIGURE 2 HERE]

**How does our Euler solution compare to our implicit solution? Does it look the same?**

**If they’re different, which one gives you more confidence?**

**What are some reasons we might be seeing this outcome?**

Let’s play with the step size, and see if that helps us. Fill in the rest of the code in section 2c, and drop the resulting figure here:

[INSERT FIGURE 3 HERE]

**What happens to our Euler solution as the step size decreases?**

**What are some tradeoffs associated with decreasing step size? Why can’t we take infinitely small steps?**

## 2d. Compare to ODE 45

Now that you’re all experts at working with ODE45, let’s check to see how our Euler solver stacks up. Head to problem 2d in the code, fill it in, and run it. Put your resulting figure here:

[INSERT FIGURE 4 HERE]

**How does ODE45’s solution compare to your implicit result?**

**What is the average step size for ODE45? How does this compare to the step sizes we tried in 2c?**

**When we run the Euler with the average step from ODE45, what do we see? What does this tell us about ODE45?**

Look at MATLAB’s documentation for ODE45. **What are some ways other than step size that ODE45 might achieve the performance we see in Figure 4?**