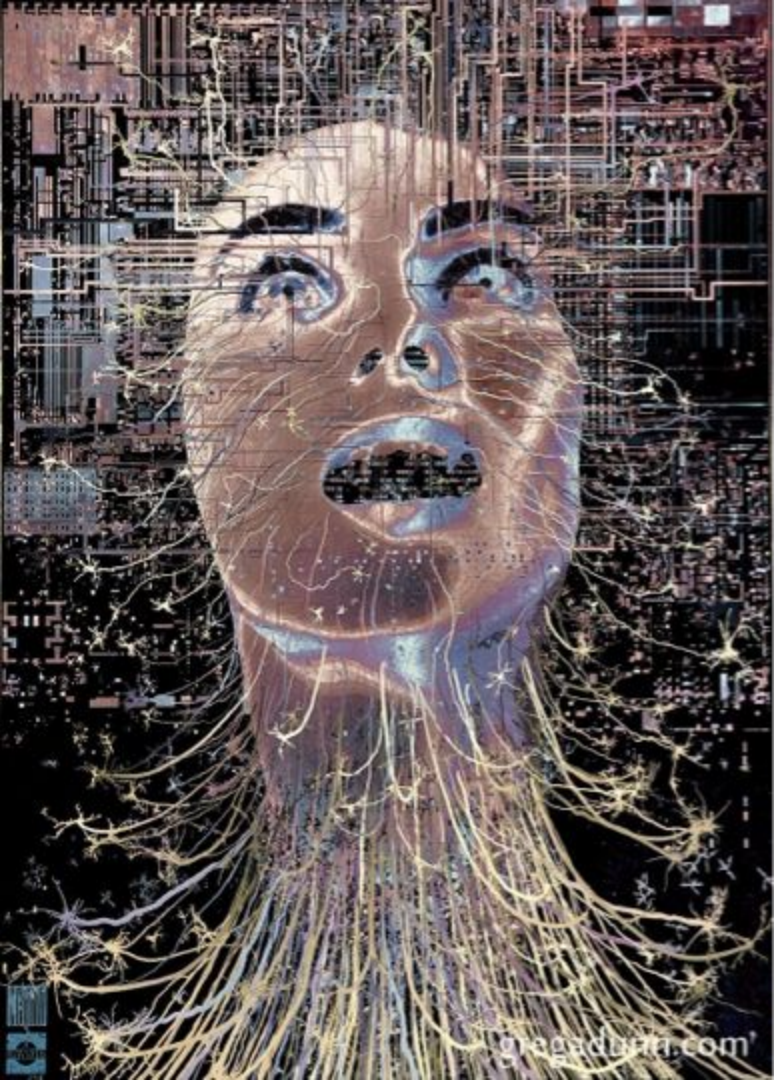


Section:
Athalye et al. 2023

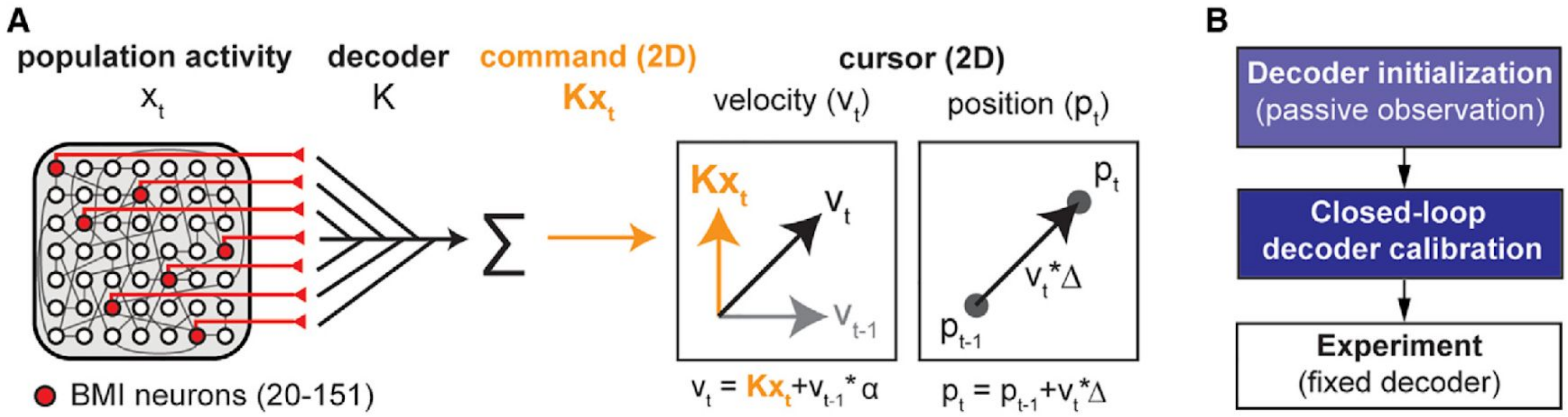
& Muscle SpikerBox Demo! 💪

Background Recap:

Section W11

Note: This week's background is mostly in the second part of the lecture

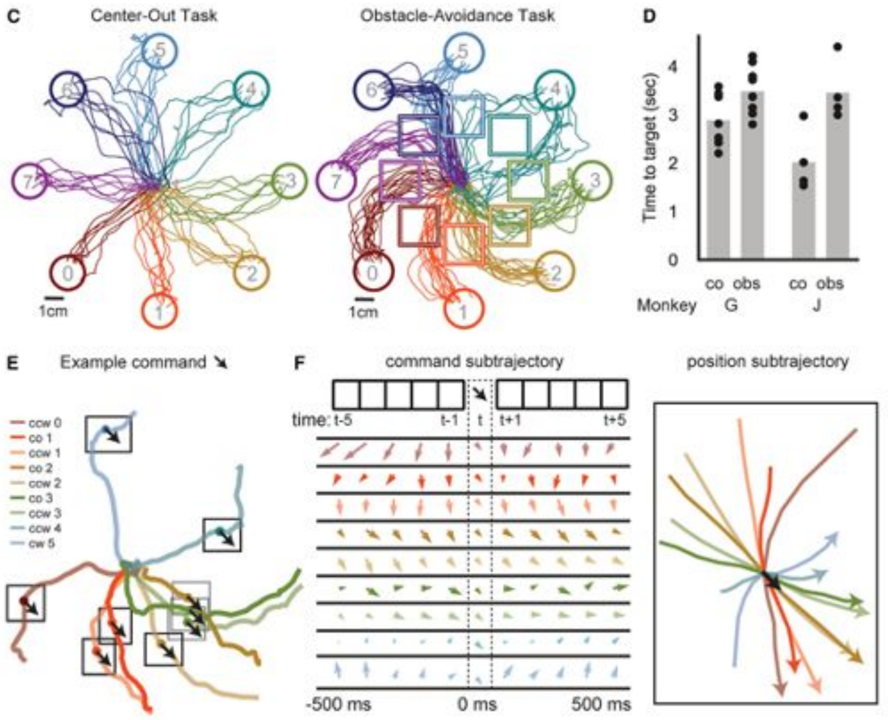
Experiment setup: training the BMI decoder



16x8 multi-shank arrays were implanted in the upper limb area of motor cortex.

Monkeys were trained to control a cursor using the implanted array, but without moving their arm.

Different neural activity patterns can give rise to the same command

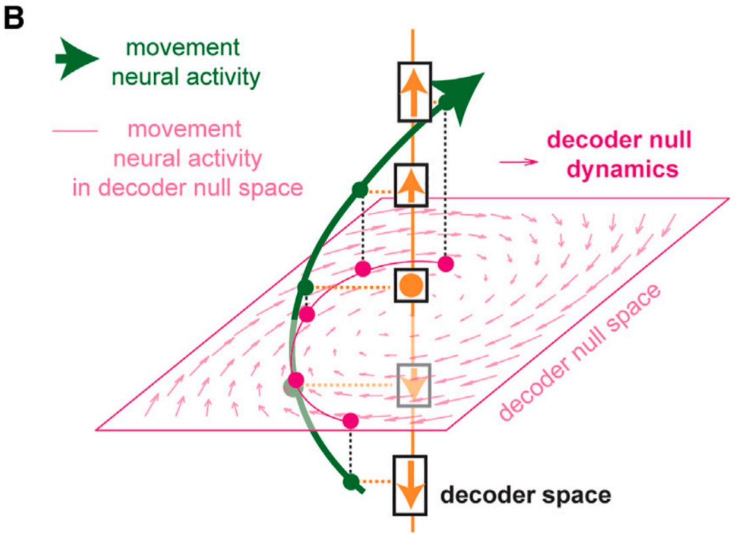


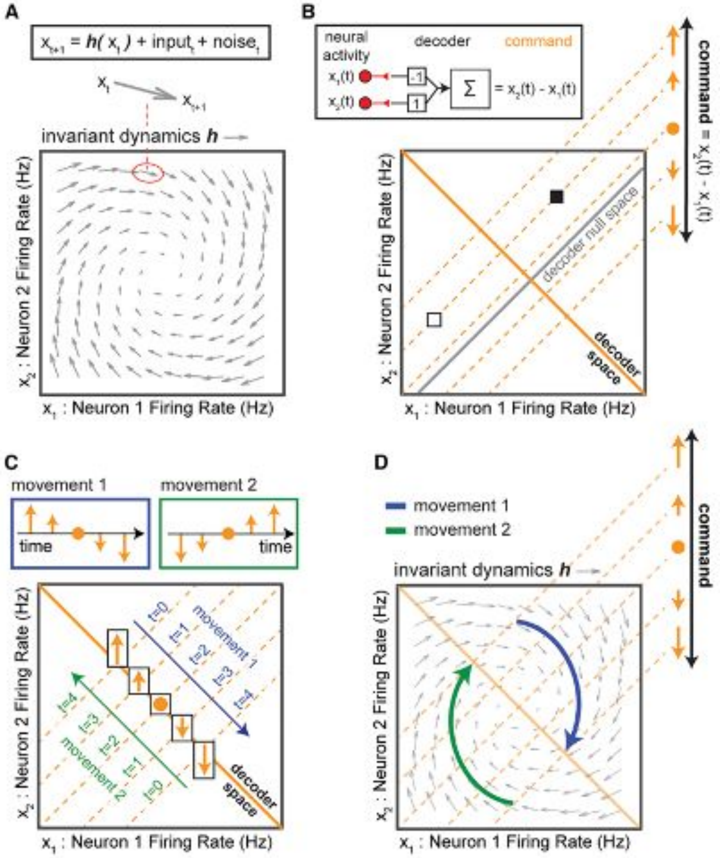
Animals were trained on two tasks, center-out and obstacle avoidance.

Because the decoder is fixed for both tasks, the motor command produced by the neural activity could be compared across tasks and trials.

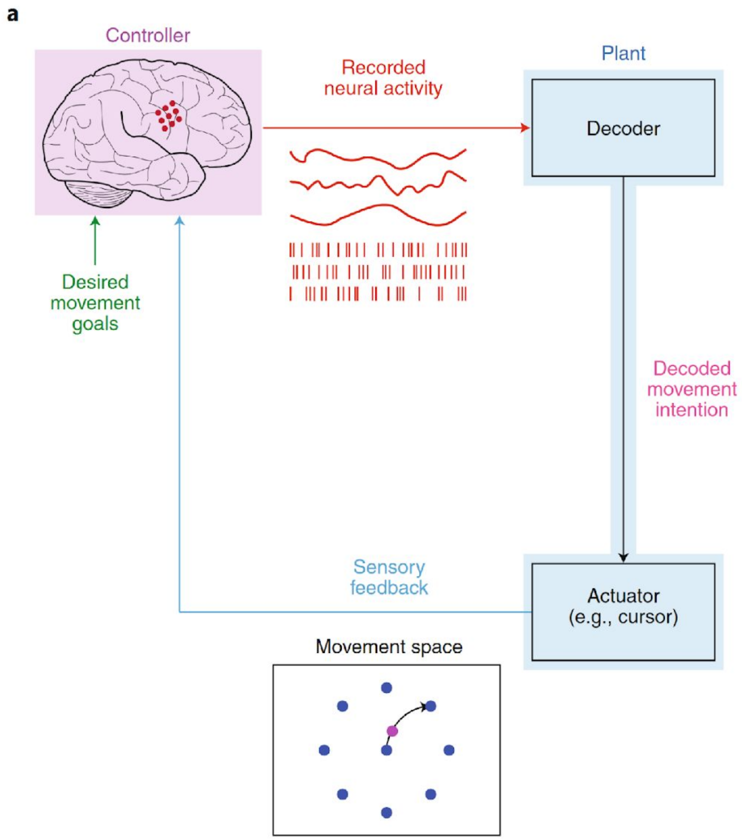
Are invariant dynamics used to control different movements?

Researchers wanted to test whether the invariant dynamics in the neural activity they recorded produced subsequent motor commands in a way that was relevant to the task and condition of each trial.





Can we demonstrate a causal relationship between invariant dynamics in motor cortex and a task-relevant command?



Section Paper:

Invariant neural dynamics drive commands to control different movements

Vivek R. Athalye,[1,7,9,14,*] Preeya Khanna,[2,7,10,*] Suraj Gowda,[4] Amy L. Orsborn,[3,11] Rui M. Costa,[1,8,12,*] and Jose M. Carmena[4,5,6,8,13,*]

It has been proposed that the nervous system has the capacity to generate a wide variety of movements because it reuses some invariant code. Previous work has identified that dynamics of neural population activity are similar during different movements, where dynamics refer to how the instantaneous spatial pattern of population activity changes in time. Here, **we test whether invariant dynamics of neural populations are actually used to issue the commands that direct movement**. Using a brain-machine interface (BMI) that transforms rhesus macaques' motor-cortex activity into commands for a neuroprosthetic cursor, we discovered that the same command is issued with different neural-activity patterns in different movements. However, these different patterns were predictable, as we found that the **transitions between activity patterns are governed by the same dynamics across movements**. These invariant dynamics are low dimensional, and critically, they align with the BMI, so that they predict the specific component of neural activity that actually issues the next command. We introduce a model of optimal feedback control (OFC) that shows that invariant dynamics can help transform movement feedback into commands, reducing the input that the neural population needs to control movement. Altogether our results demonstrate that invariant dynamics drive commands to control a variety of movements and show how feedback can be integrated with invariant dynamics to issue generalizable commands.

Figure 3. The same command is issued by different neural-activity patterns in different movements

Reminder:

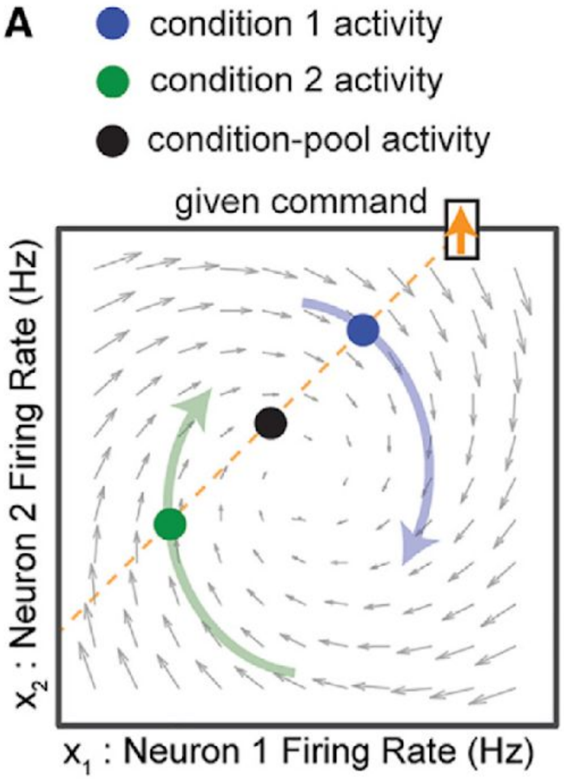
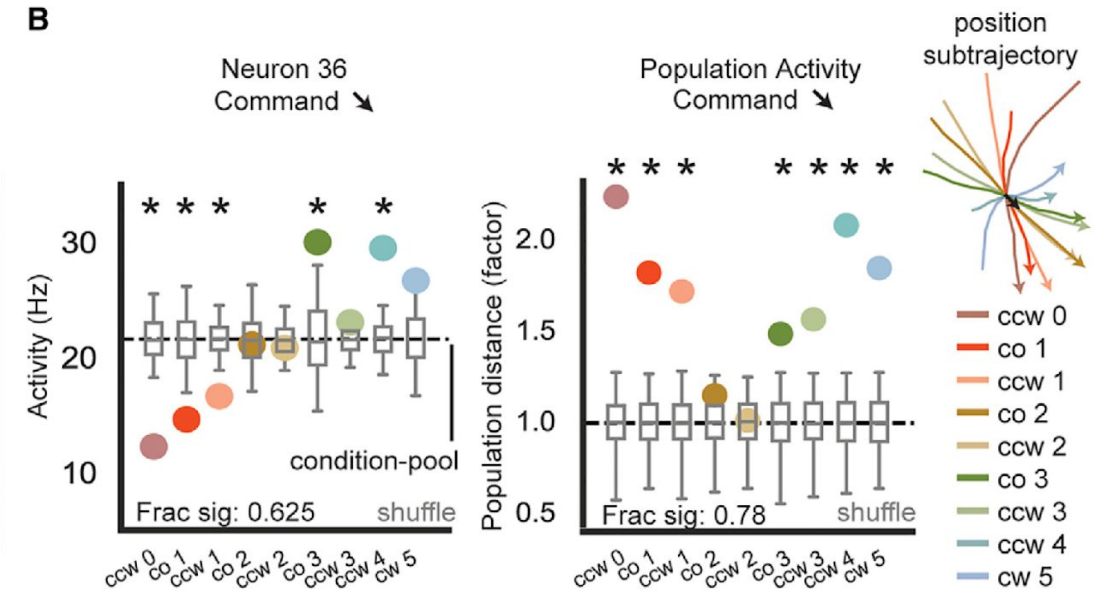


Figure 3. The same command is issued by different neural-activity patterns in different movements



What can you say about the relationship between position sub-trajectory and population distance?

The further a sub-trajectory is from the middle of the trajectory distribution, the more the neural activity producing that command differs from the average activity to produce the command.

Figure 4. Invariant dynamics predict the different neural activity patterns used to issue the same command

Reminder:

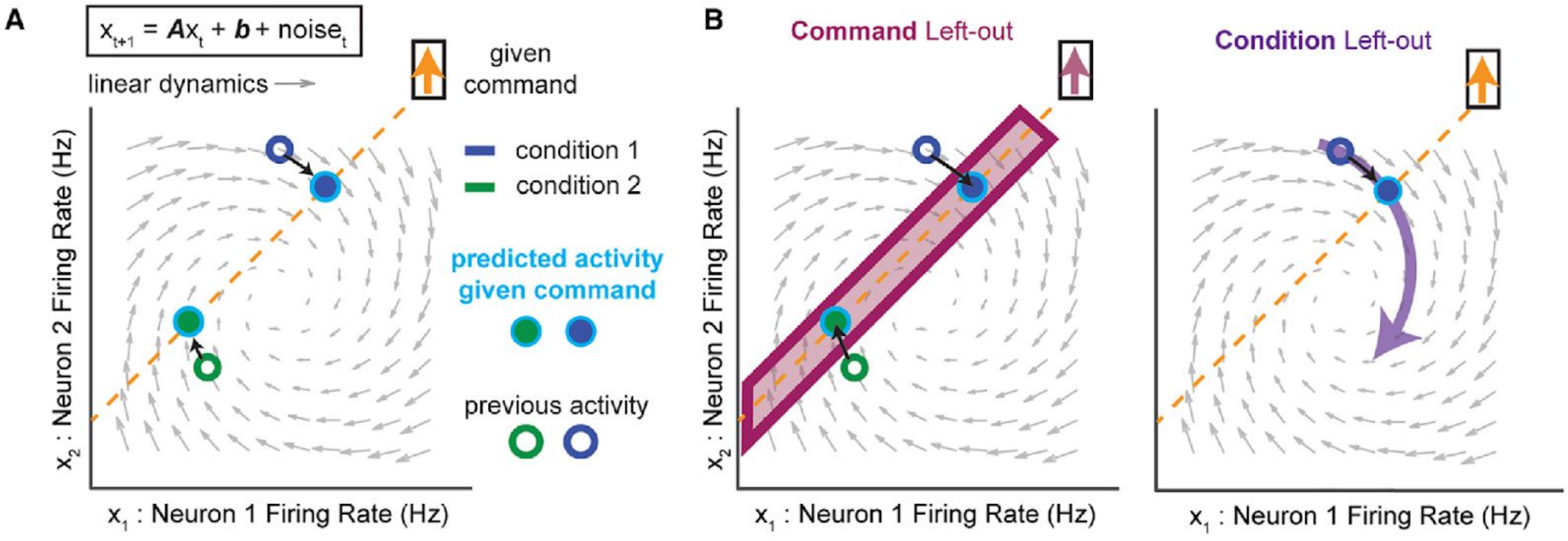
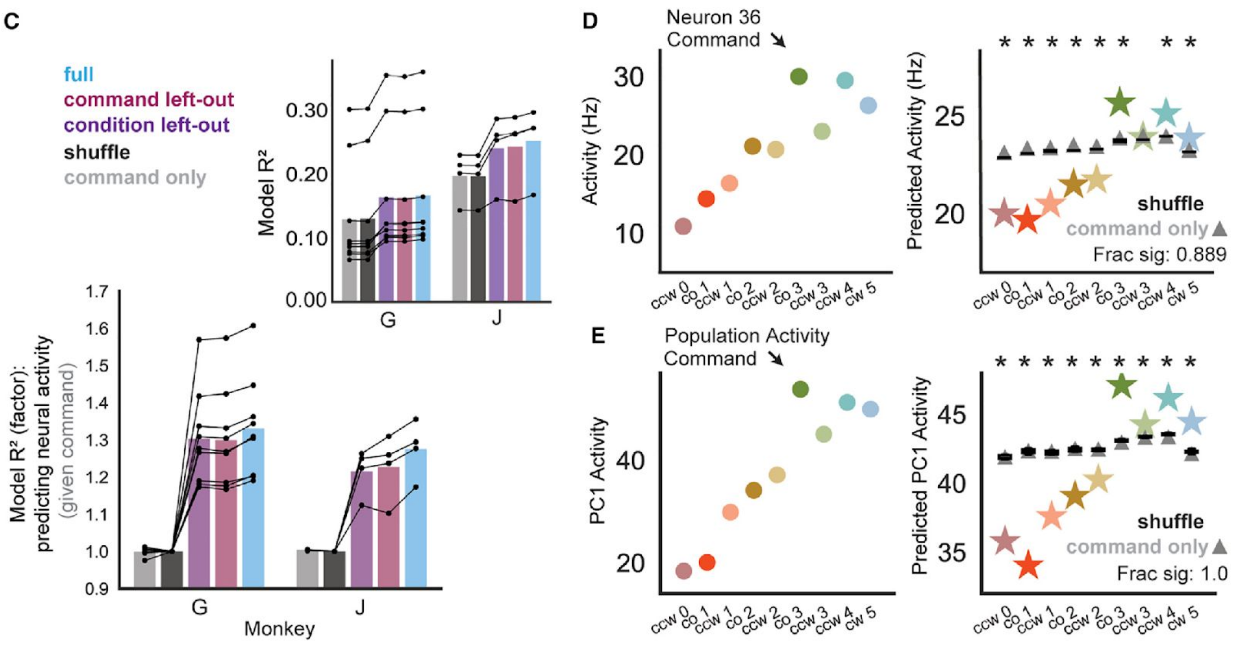


Figure 4. Invariant dynamics predict the different neural activity patterns used to issue the same command



There is very little decline in neural activity prediction accuracy when either the command or the condition is removed. What does this suggest about the role that invariant dynamics play in this circuit?

Invariant dynamics drive a large portion of the neural response in the circuit, as removing task-related inputs does not disrupt activity predictions.

Figure 5. Invariant dynamics align with the decoder, propagating neural activity to issue the next command

Reminder:

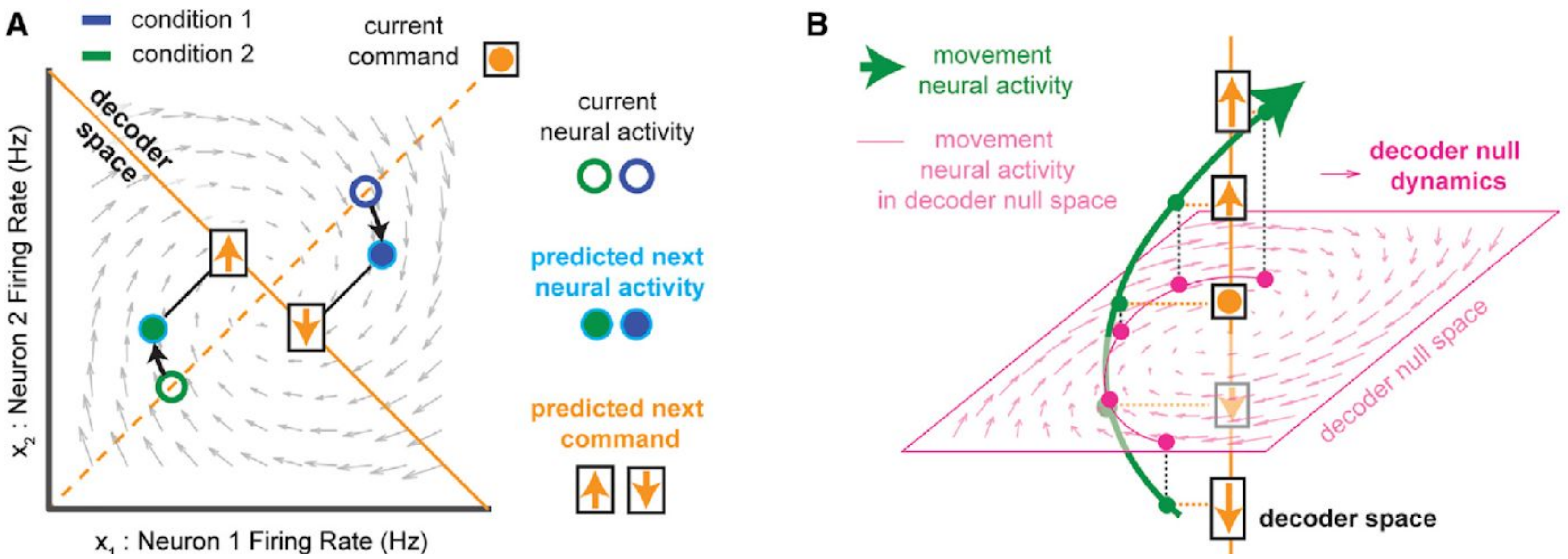
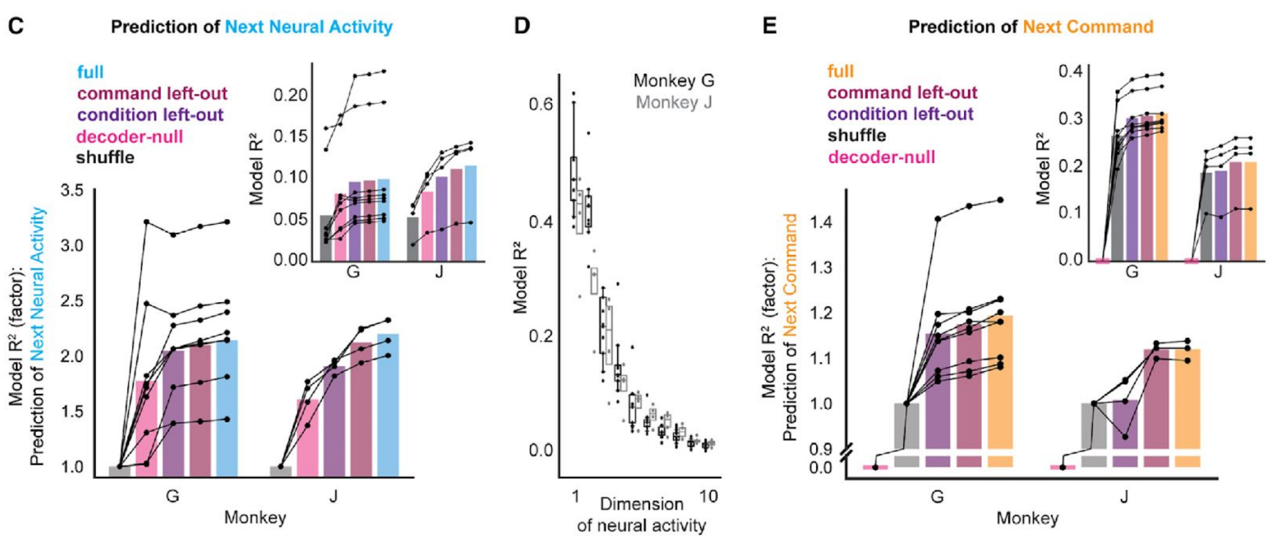


Figure 5. Invariant dynamics align with the decoder, propagating neural activity to issue the next command



How does this predictive model test differ from the one in Figure 4?

The previous model decodes the neural activity for a command at time t, since many activity patterns drive the same command.

This model predicts the command and neural activity at t+1.

Figure 6. An OFC model reveals that invariant dynamics reduce the input that a neural population needs to issue commands based on feedback

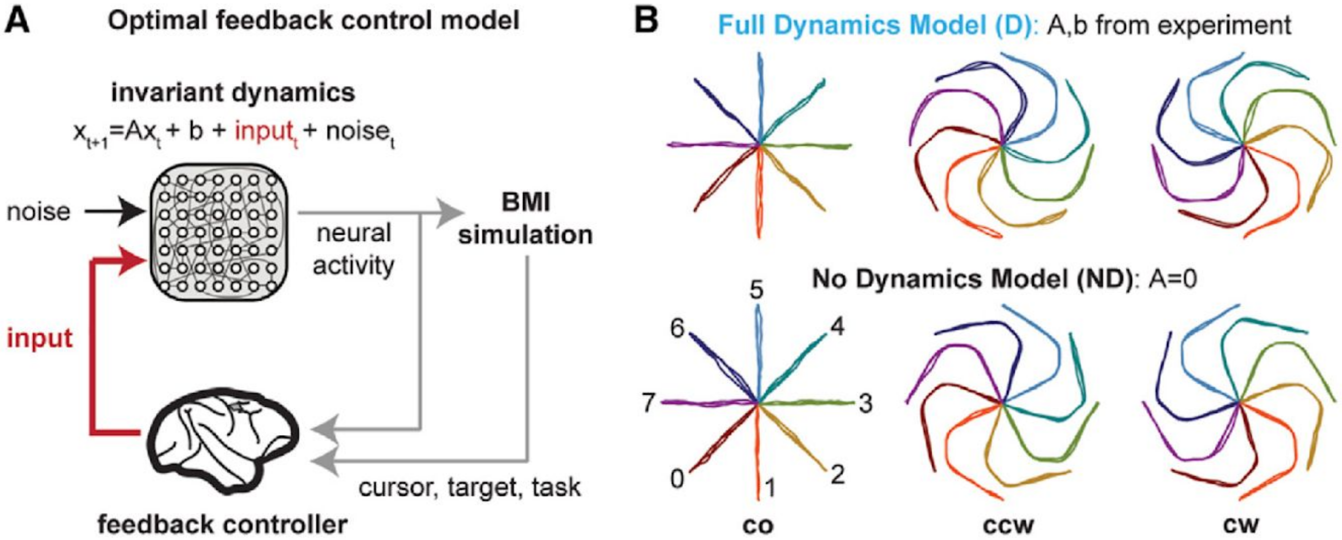
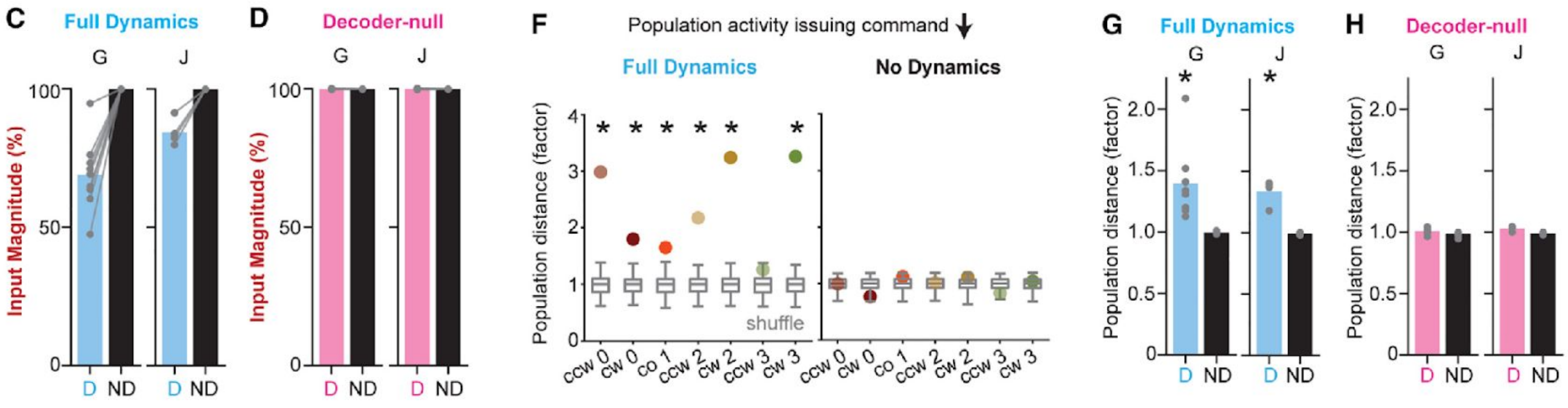


Figure 6. An OFC model reveals that invariant dynamics reduce the input that a neural population needs to issue commands based on feedback



Summarize the two main findings from this figure (this is a boring question, I will try to find a better one).

The optimal feedback controller results in more efficient inputs to produce movement. The population distance across commands only resembles the in vivo data if an optimal feedback controller is included in the model.

Paper roundup

- Monkeys were trained to control a cursor in several tasks using a BMI based on neural activity recorded in motor cortex.
- Invariant dynamics in the recorded neurons could predict the neural activity that was used to produce a motor command, even when task inputs were removed from the model.
- Invariant dynamics alter neural activity in dimensions relevant to the decoder, demonstrating a causal link between invariant dynamics and motor commands (at least in this BMI setting).
- Adding an optimal feedback controller to an *in silica* model of invariant dynamics trained to perform the center out tasks reduced the amount of inputs needed for successful execution.

What did we learn? What questions do we have?

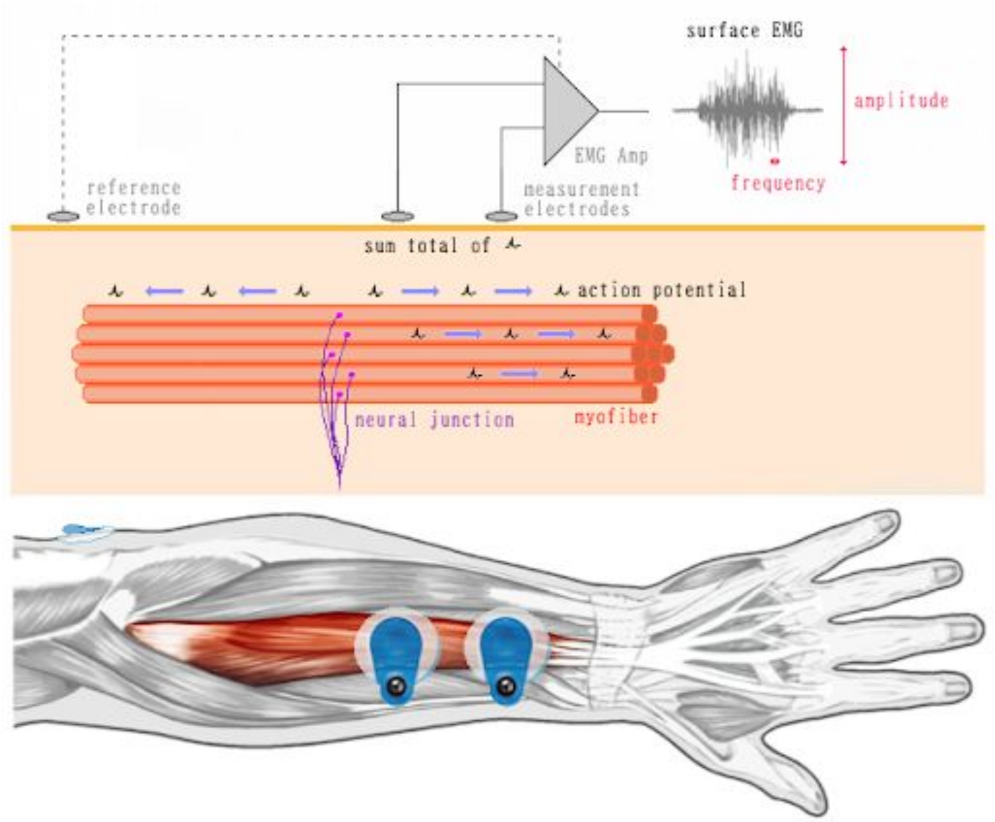
- What points do they make in the discussion?
- Is anything unclear?
- What would you do next if you had to design an experiment?
  - *Invariant dynamics → how irrelevant is it? It's related to the decoder they defined.*

Muscle SpikerBox Demo! 💪

Electromyography (EMG)

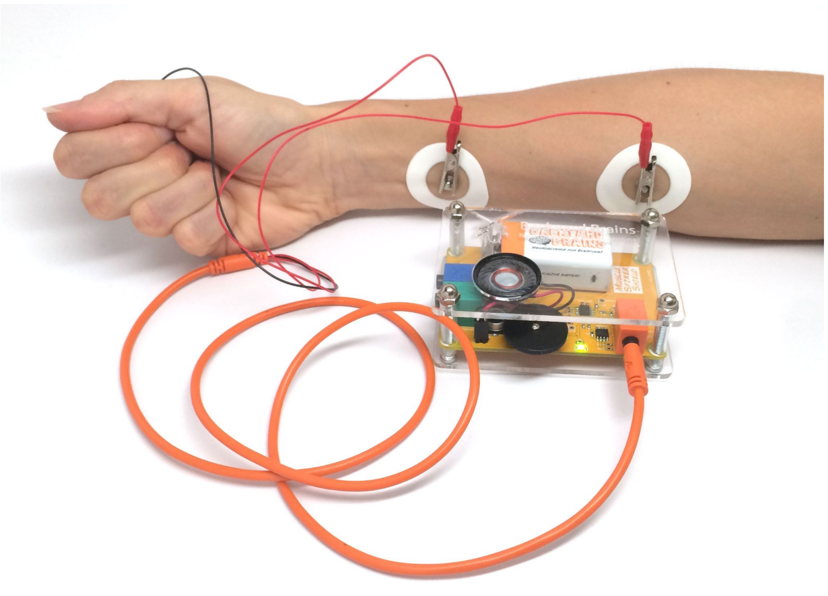
EMG records muscle potentials that are triggered by lower motor neurons.

Stronger muscle contractions recruit more muscle fibers, resulting in higher frequency spiking.

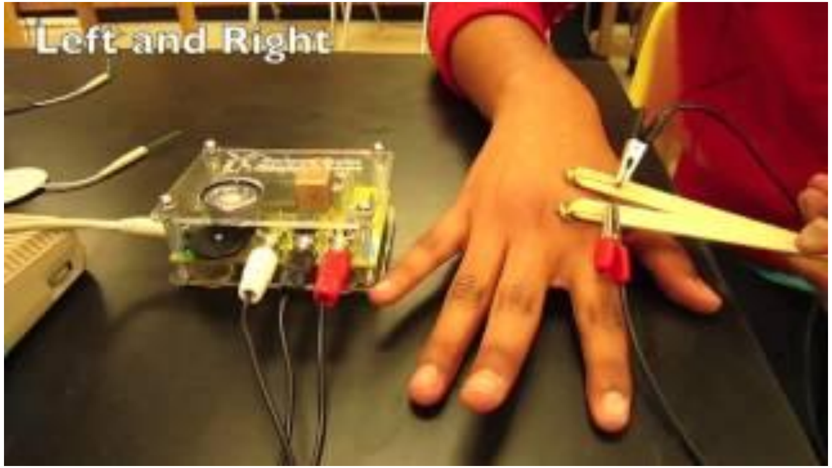


Muscle SpikerBox

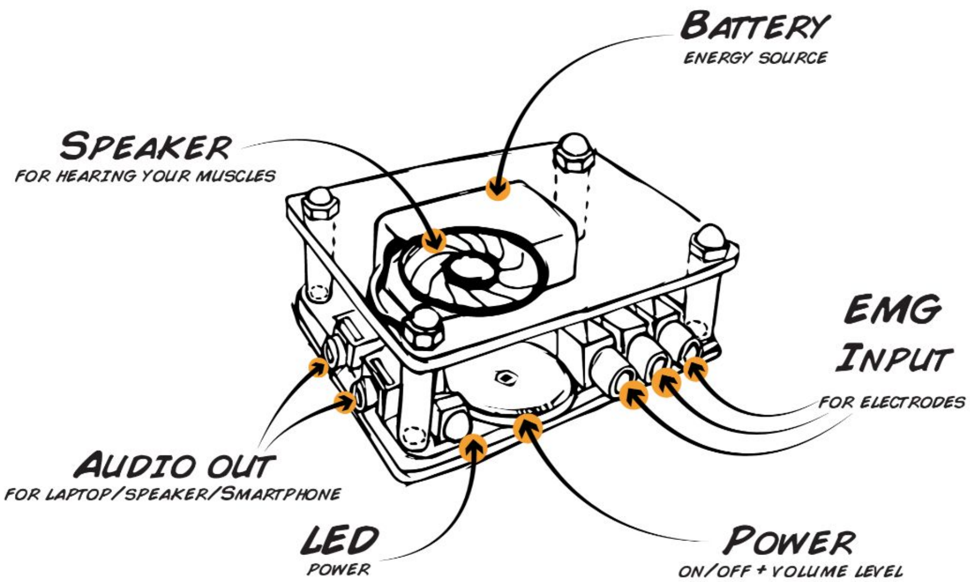
The muscle spikerbox is an electromyograph powered by an arduino.

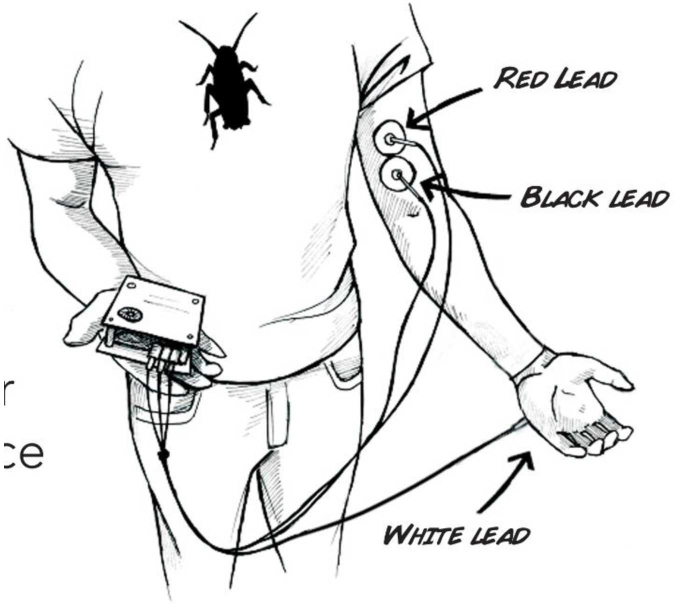


BACKYARD BRAINS



Muscle SpikerBox





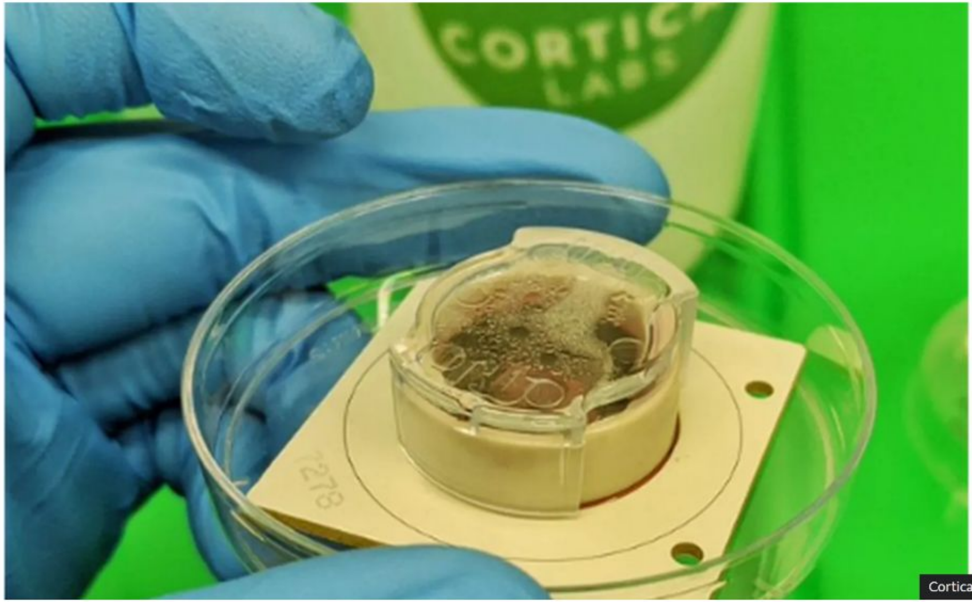
Docs:https://backyardbrains.com/products/files/EMG_insert.pdf

Lab-grown brain cells play video game Pong

12 October 2022

Share

By Pallab Ghosh, Science correspondent



Cortical Labs

These 800,000 lab-grown brain cells can play a 1970s video game, albeit badly

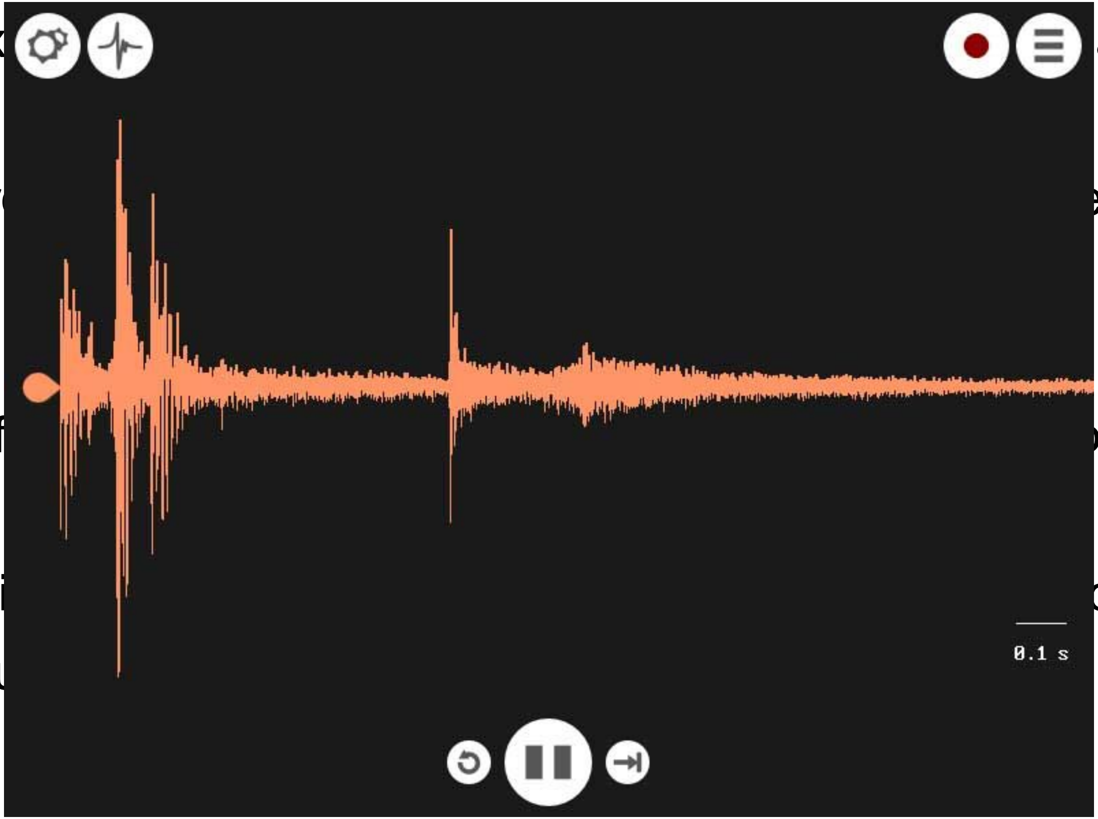
EMG Pong

We will be using the SpikerBox today to make a very simple BMI to play pong on your computer!

If you have not already downloaded the repository from the course github please do so now. Once downloaded, build the conda environment specified in the docs.

Also be sure to download the SpikerBox PC app.
http://www.backyardbrains.com/experiments/files/Backyard_Brains_Neuron_Recorder_Install.air.zip

The SpikerBox PC app

The SpikerBox 

al time.

It can record fr e right

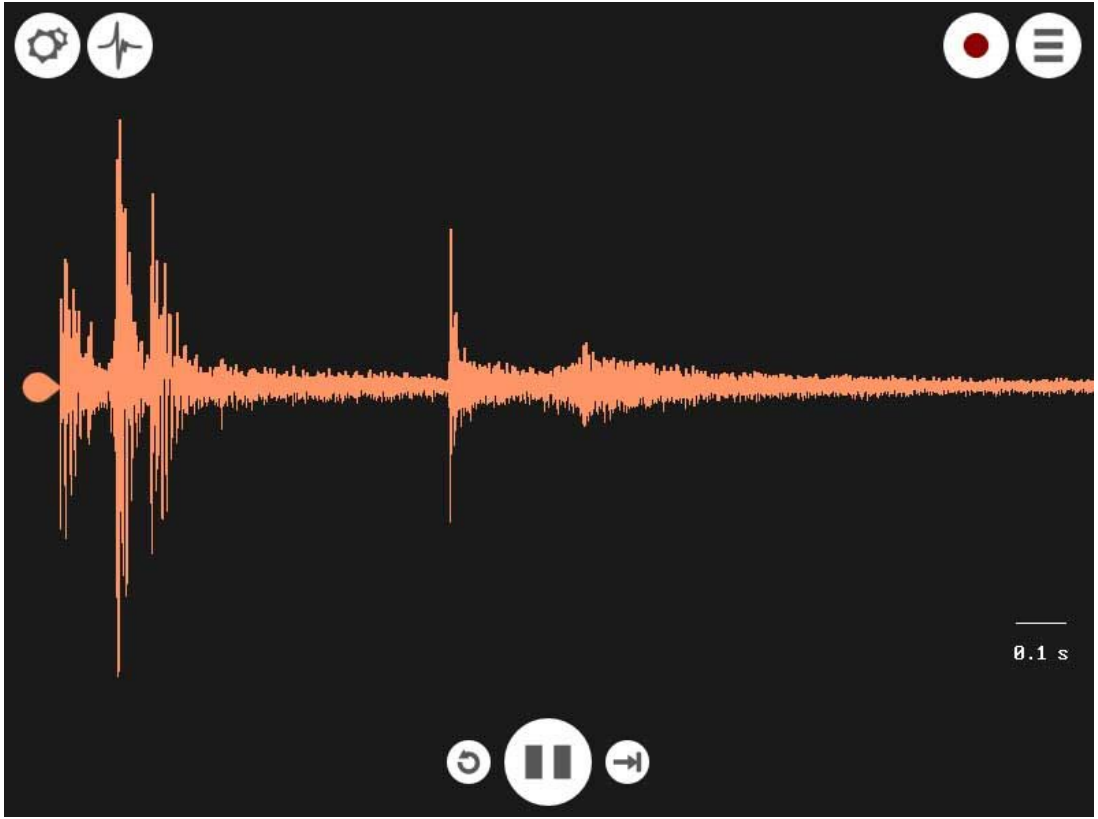
hardware.

It can also perf on.

The one thing i ch is why we

will be using ou

The SpikerBox PC app



The SpikerBox PC app

The SpikerBox app is used visualize the your EMG activity in real time.

It can record from multiple data streams at once, if you have the right hardware.

It can also perform online thresholding for waveform visualization.

The one thing it can’t do, is stream easily to other software, which is why we will be using our own streaming script for pong.

EMG Pong Instructions

These instructions can also be found in the repository docs. Do not hesitate to ask for help if needed, have fun!

1. Create the conda environment.
2. Plug in the Muscle SpikerBox to a USB port on the computer and setup your contacts
3. Activate the environment and open the spike_stream_threshold notebook.
4. Run through the notebook and determine some thresholds you want to try for game control.
    a. Once you are finished, make sure to stop the kernel to make the serial port available for the game.
5. Open the stream.py file and edit the variables at the top of the code.
6. Run the script, and start playing!

Building a better BMI

While you are working with the BMI controlled pong, consider how you might alter or optimize this system. Here are a few questions to consider:

The current game controls the paddle by sampling a running mean from the EMG, how else could you threshold the controller?

Could you add a case that would allow the paddle to stay in place, instead of just rising or falling?

The game update speed is linked to the stream buffer. Can you think of ways to make the system faster?