TIMESERIES 2

11.2.2018

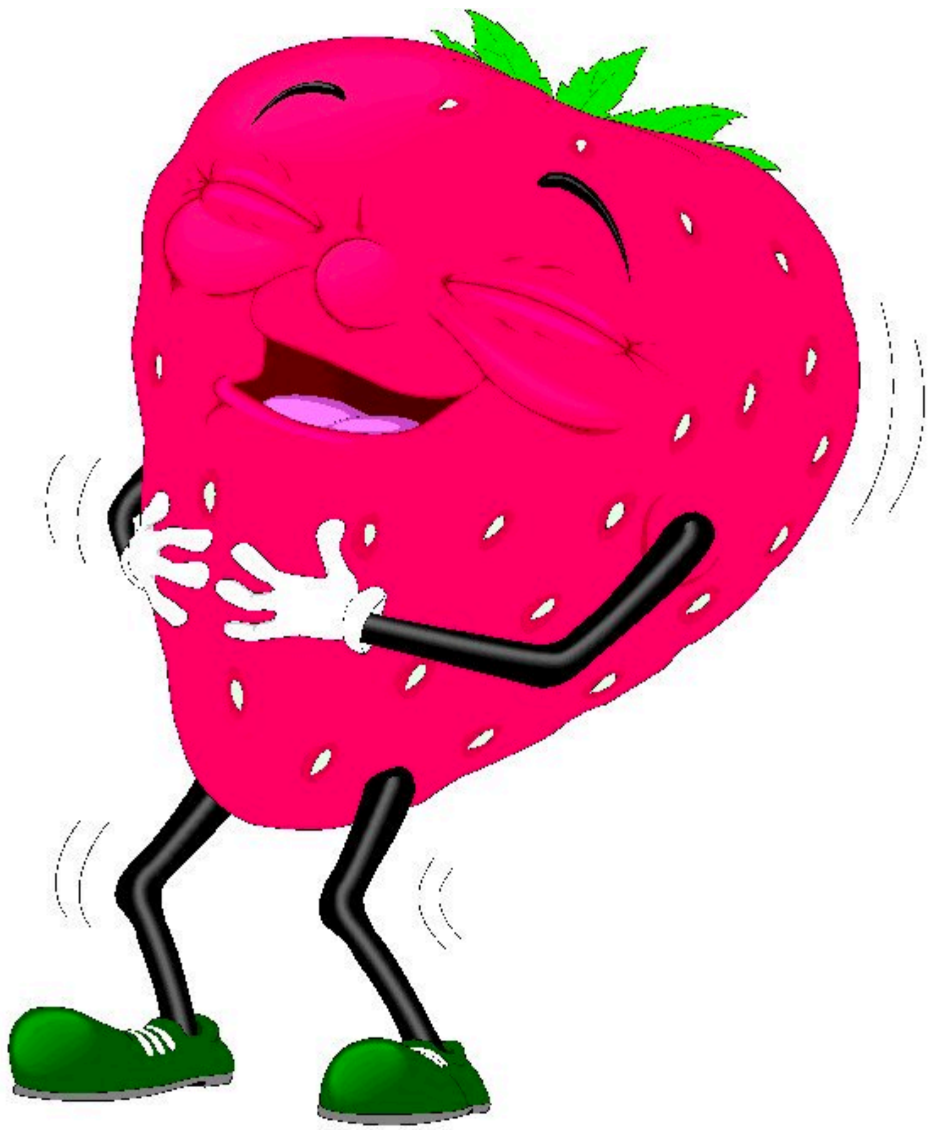
PROBLEM SET 3

* was due today! which means you’re probably done!

PROBLEM SET 3

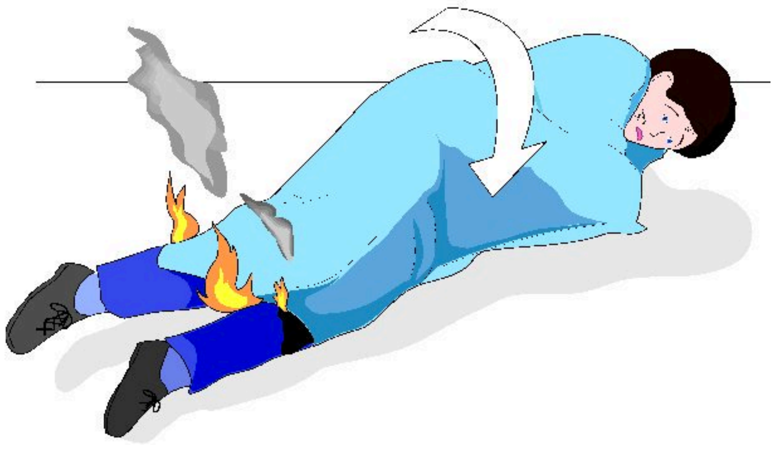
* please take a moment to pat yourself on the back for being great

* because you are great



PROBLEM SET 4

* will be posted end of next week



OSCILLATION

* why do things oscillate/vibrate/wobble/wiggle/whatever?

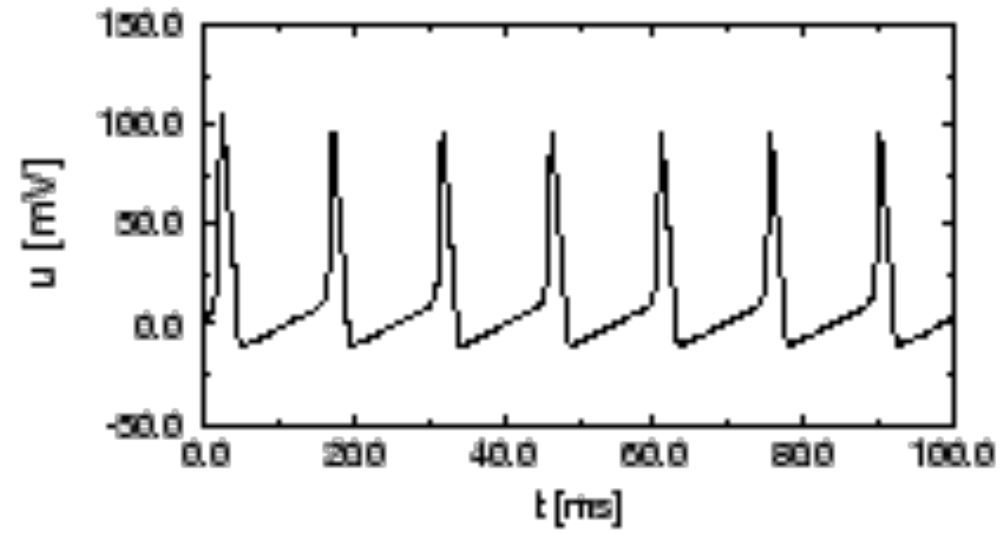
OSCILLATION

* *feedback cycles*
  * A causes B causes C causes … causes A
* (but not all feedback cycles cause oscillations)

OSCILLATION

* some feedback cycles are complicated, involving lots of variables that are related in non-linear ways

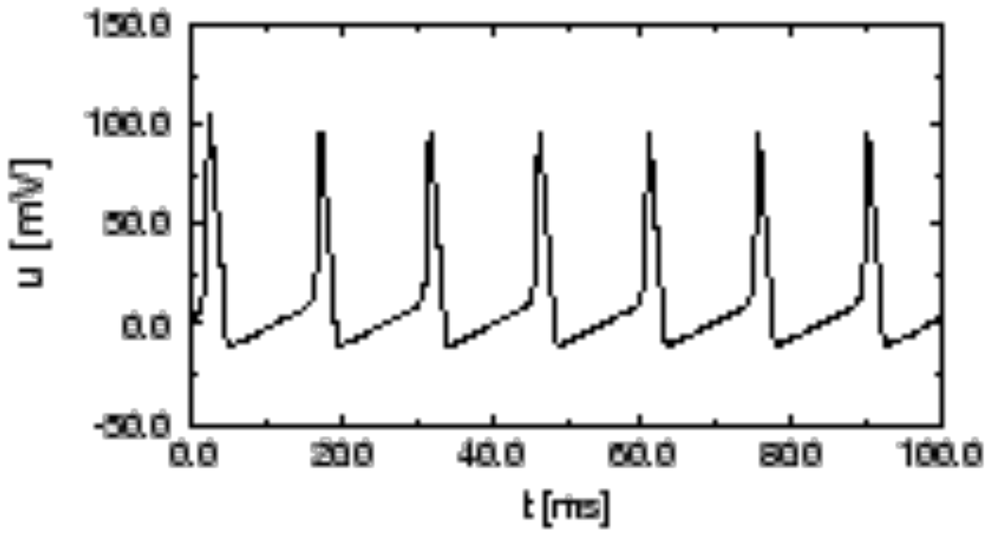
* like the Hodgkin-Huxley equations that (mostly) govern how action potentials work in neurons



OSCILLATION

* these complicated feedback cycles can generate periodic outputs

* but they tend to be weird looking (like action potentials)



OSCILLATION

* but many feedback cycles are quite simple

* a common type is the *harmonic oscillator*

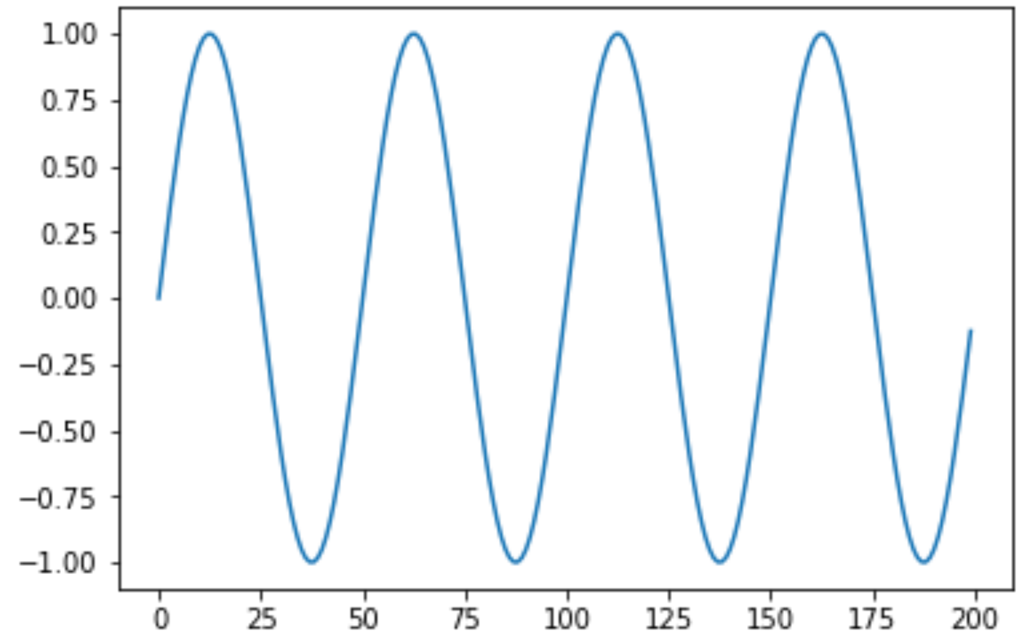
  * these appear wherever acceleration (or force) is negatively proportional to location

  * e.g. spring, rubber band, pendulum, most things bouncy or springy

OSCILLATION

* instead of complicated, weird looking outputs, harmonic oscillators always generate very nice and simple outputs:

  * sine waves



OSCILLATION

* for this (and other, more mathematical) reason(s), it’s often useful to think of timeseries as the sum of a bunch of sine waves with different frequencies

* this is called *fourier analysis*

FOURIER ANALYSIS

* the *fourier transform* is a function that figures out how to represent your timeseries as a sum of sine waves

* every possible timeseries has a fourier transform

* (although it might take infinitely many sine waves)

Joey Fourier

FOURIER ANALYSIS

FOURIER ANALYSIS

* the fourier transform of a timeseries f is often written F

* if the units of f are seconds, then the units of F are (1/seconds) or hertz (Hz)

FOURIER ANALYSIS

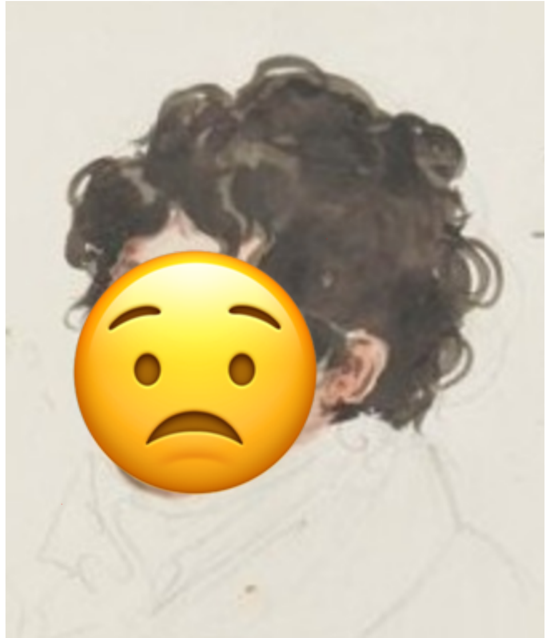
* to take a fourier transform of an array you can use **np.fft.fft**
  * (fft is the “fast fourier transform” algorithm invented by Cooley & Tukey)
* but you *almost never* want to use this directly
* (unless you really know what you are doing)

THE PROBLEM WITH FOURIER TRANSFORMS

* for the fourier transform to be invertible, its input and output have the same dimensionality

* that means the fourier transform of a 1-million-point timeseries gives you 1 million frequencies

* this makes fourier transforms noisy, unwieldy, and unreliable



SPECTRAL ANALYSIS

* if you want to know which frequencies make up a timeseries, you should probably compute the **power spectrum** or **power spectral density (psd)**

* common psd methods (such as *welch's periodogram*) behave much more nicely than plain fourier transforms in many situations

SPECTRAL ANALYSIS

* spectral density estimators work by taking the fourier transforms of many small snippets (aka windows) of the signal, and then averaging the results

* thus the psd can have many fewer points than the original signal

* which means that it’s better behaved, and less sensitive to noise, etc.

THE SPECTROGRAM

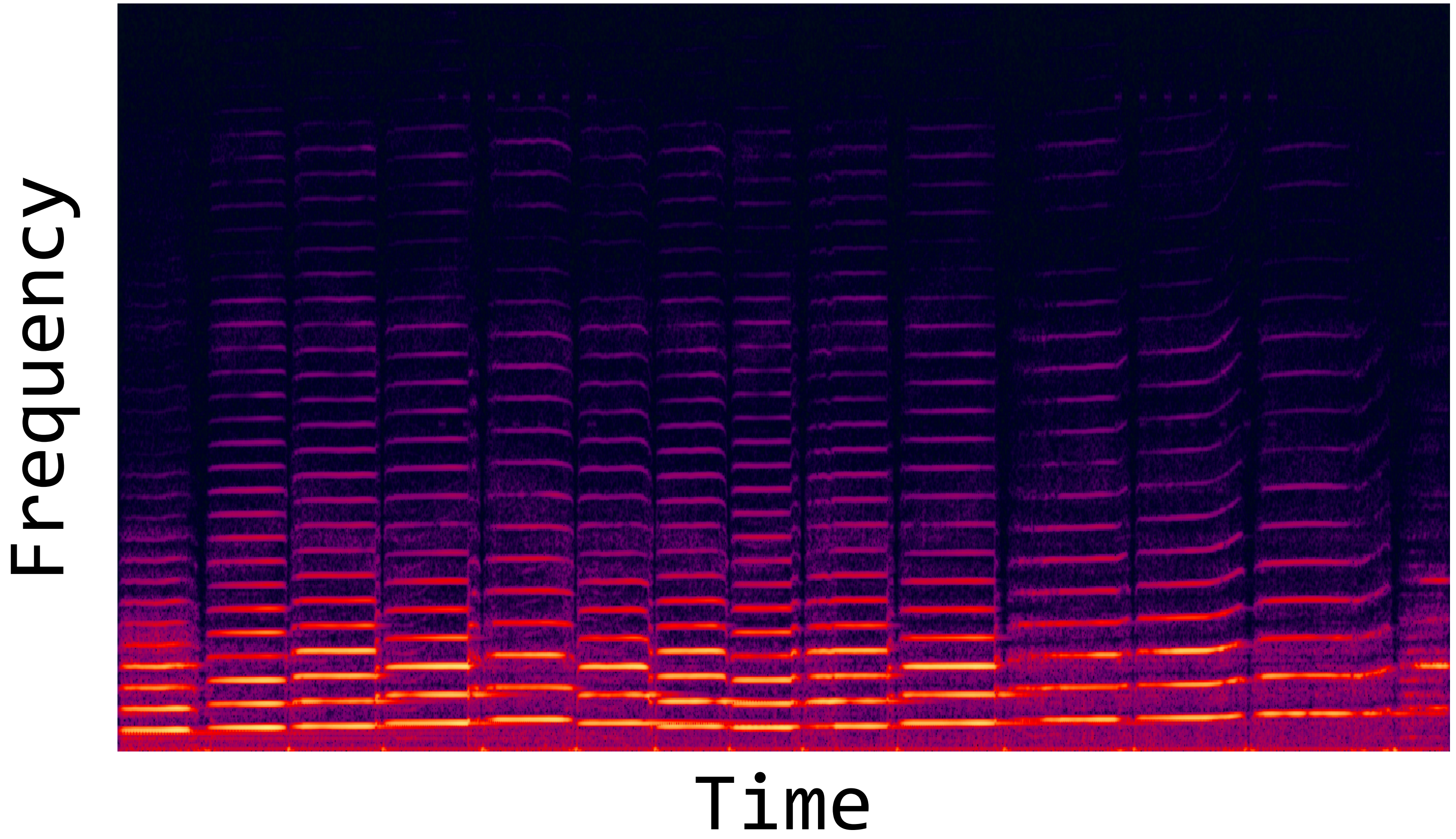
* what if we took the fourier transform of many small snippets of our timeseries, and then just looked at them instead of averaging them together?

* this is called a **spectrogram**

* a spectrogram tells you which frequencies are present in a timeseries *at each time*

THE SPECTROGRAM

* spectrograms are 2-dimensional arrays with time on the x-axis (columns) and frequency on the y-axis (rows)



THE SPECTROGRAM

* matplotlib provides an excellent method for computing spectrograms: **plt.specgram**

GOOGLE SPECTROGRAM

* https://musiclab.chromeexperiments.com/Spectrogram/

CORTEX VORTEX

* http://changlabucsf.github.io/cortexvortex/build/index.html

END