Robotics Studio

Step #6: Simple Gait – “Baby Steps”

Requirements

- Goal: Make your robot walk (manually)
  - Move joints in sinusoidal pattern
  - Hand tune parameters to make robot locomote
  - Measure speed
- Next assignment
  - Train your robot to walk using a Simulation + Hillclimber

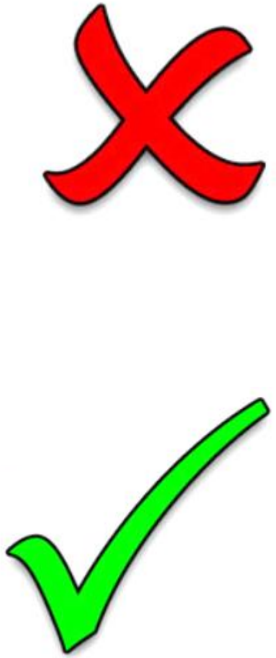
Three approaches to making robots walk

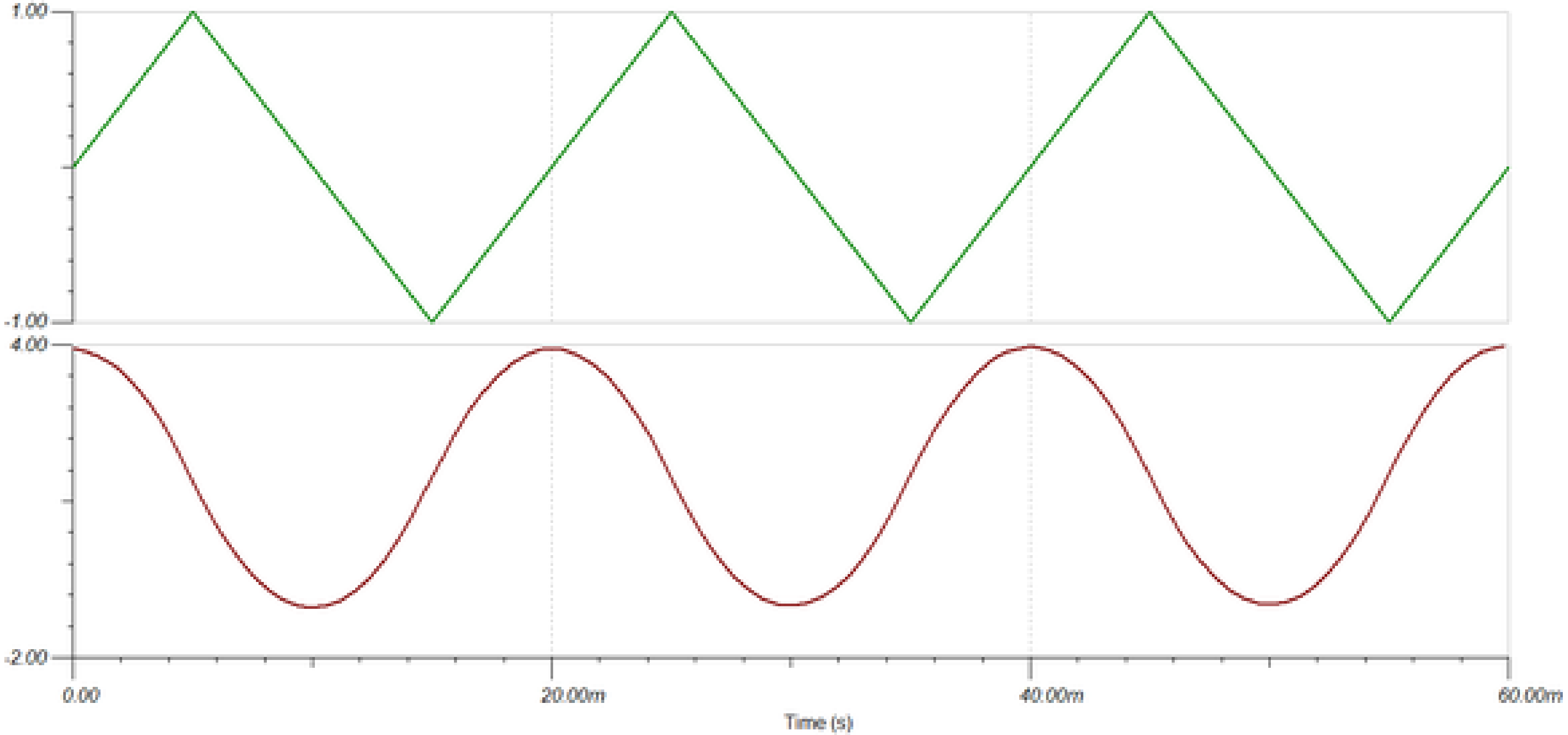
- Trial and Error: Hand coding a locomotion gait and testing
  - Simplest to do, OK results
- Analytical: Inverse Kinematics
  - Harder to do, better results
- Optimization / Machine learning
  - Hardest to do, best results
  - Next assignment, optional

Approach 1: Parameterized locomotion

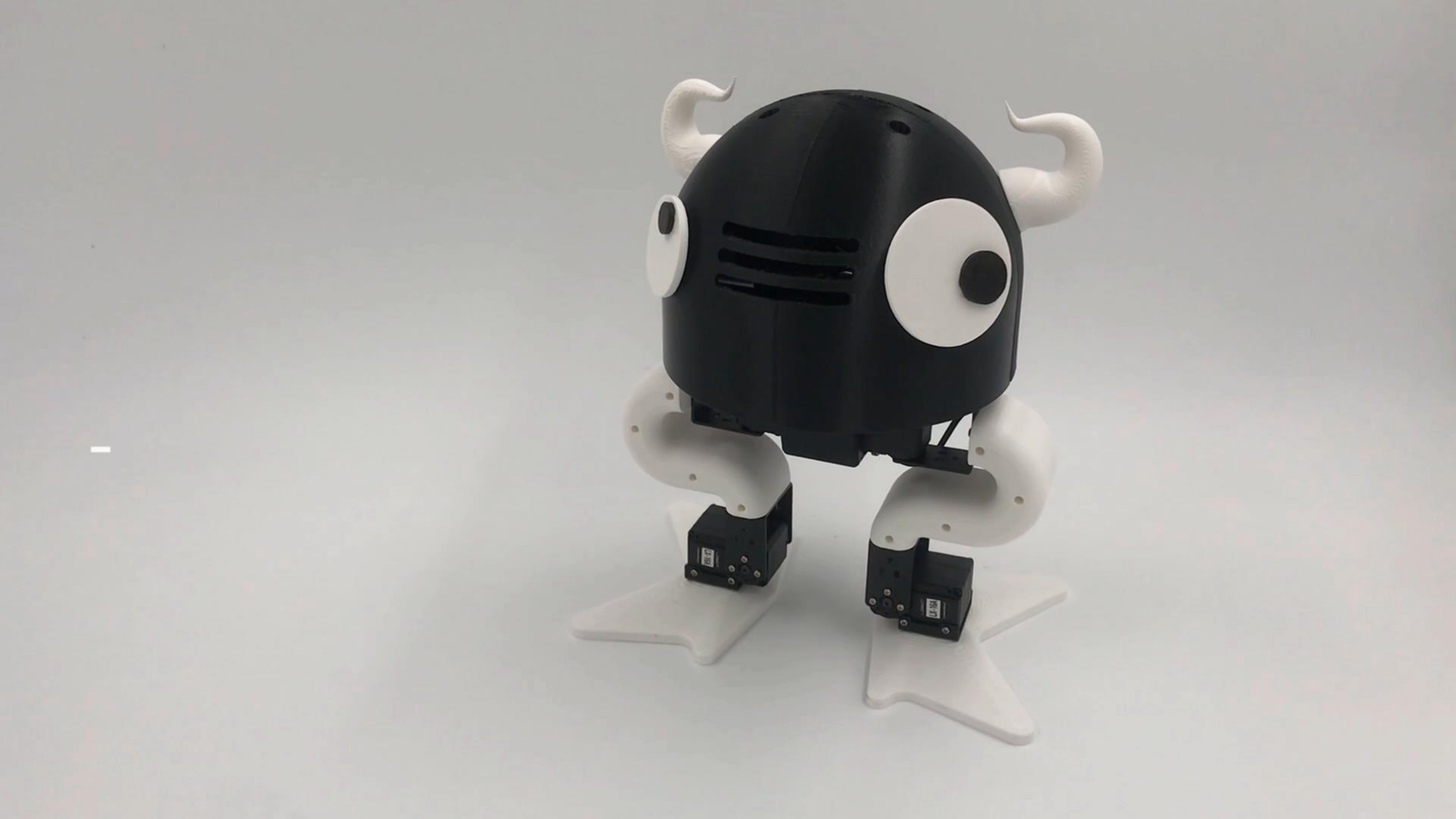
- Create a general locomotion pattern with free parameters
- Optimize the parameters manually

Avoid sharp angular velocity discontinuities





Don’t grind the gears!



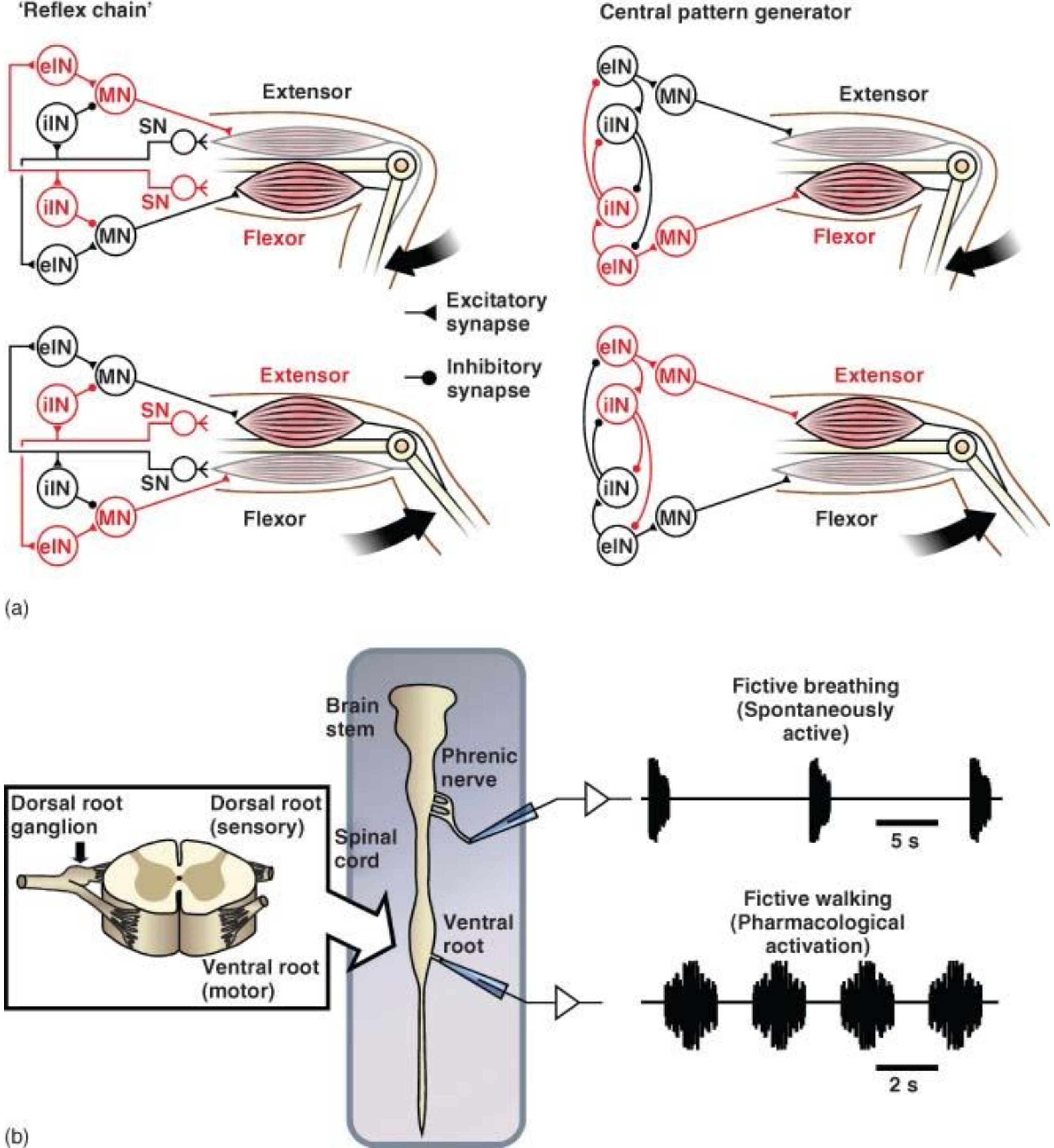
Choose a parametric locomotion pattern

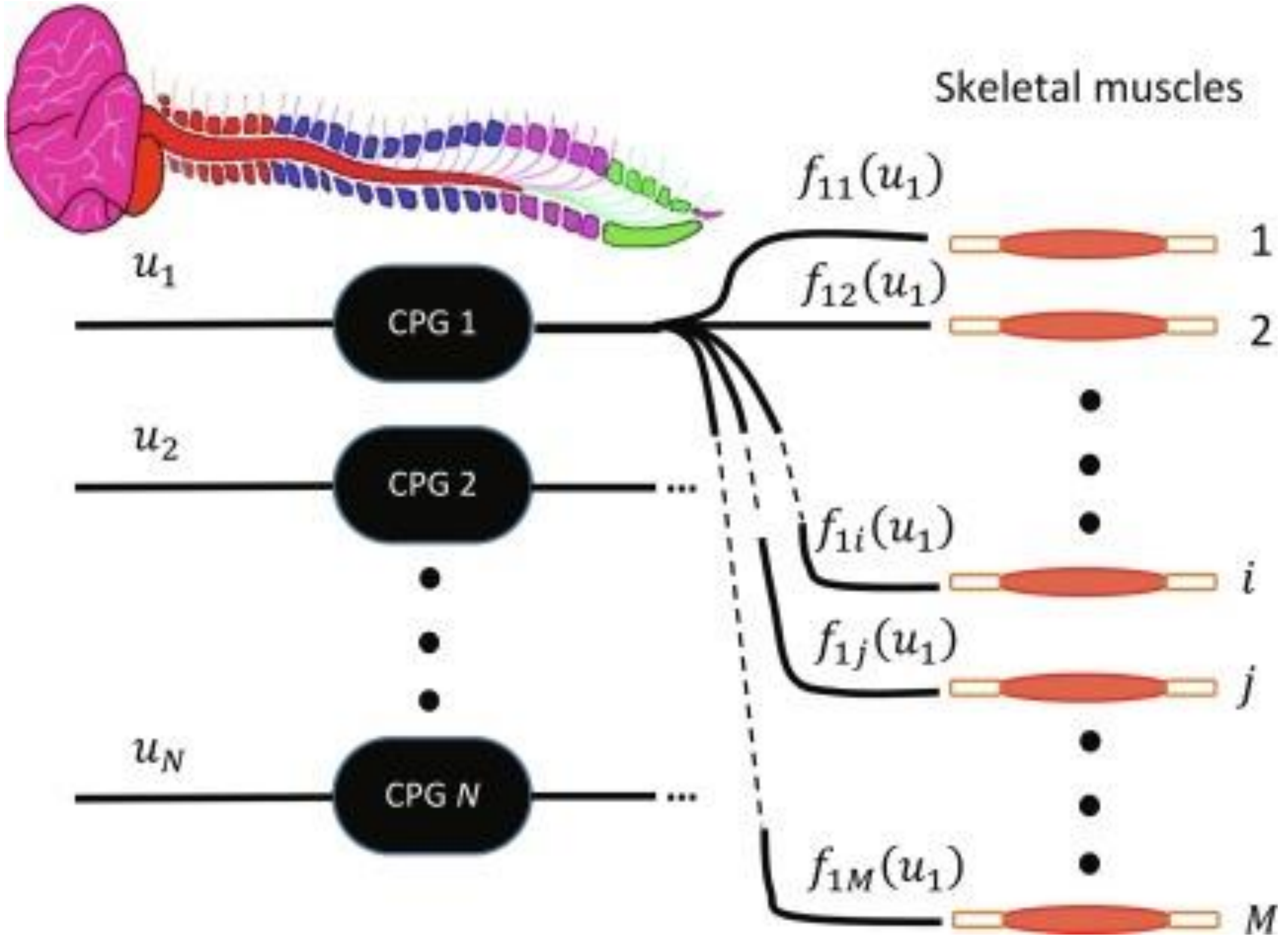
- Periodic gait
  - $\theta_i(t)=\mathbf{a_i}+\mathbf{b_i}*\sin(\omega t+\mathbf{c_i})$ *ω uniform for all legs. $\omega=1$ → period 2π secs*
- Try modifying parameters
  - Until you get the robot to walk
  - Reduce the search space with symmetries
    - E.g. opposite legs identical motion with phase offset

Open Loop Control

$\theta_i(t)$ is the angle of each of the revolute joints *i=0..n*

Q: How many parameters does as 12-joint robot have?


'Reflex chain'
Central pattern generator
eIN
MN
iIN
SN
Extensor
Flexor
Excitatory synapse
Inhibitory synapse
(a)
Brain stem
Phrenic nerve
Spinal cord
Ventral root
Dorsal root ganglion
Dorsal root (sensory)
Ventral root (motor)
Fictive breathing (Spontaneously active)
5 s
Fictive walking (Pharmacological activation)
2 s
(b)

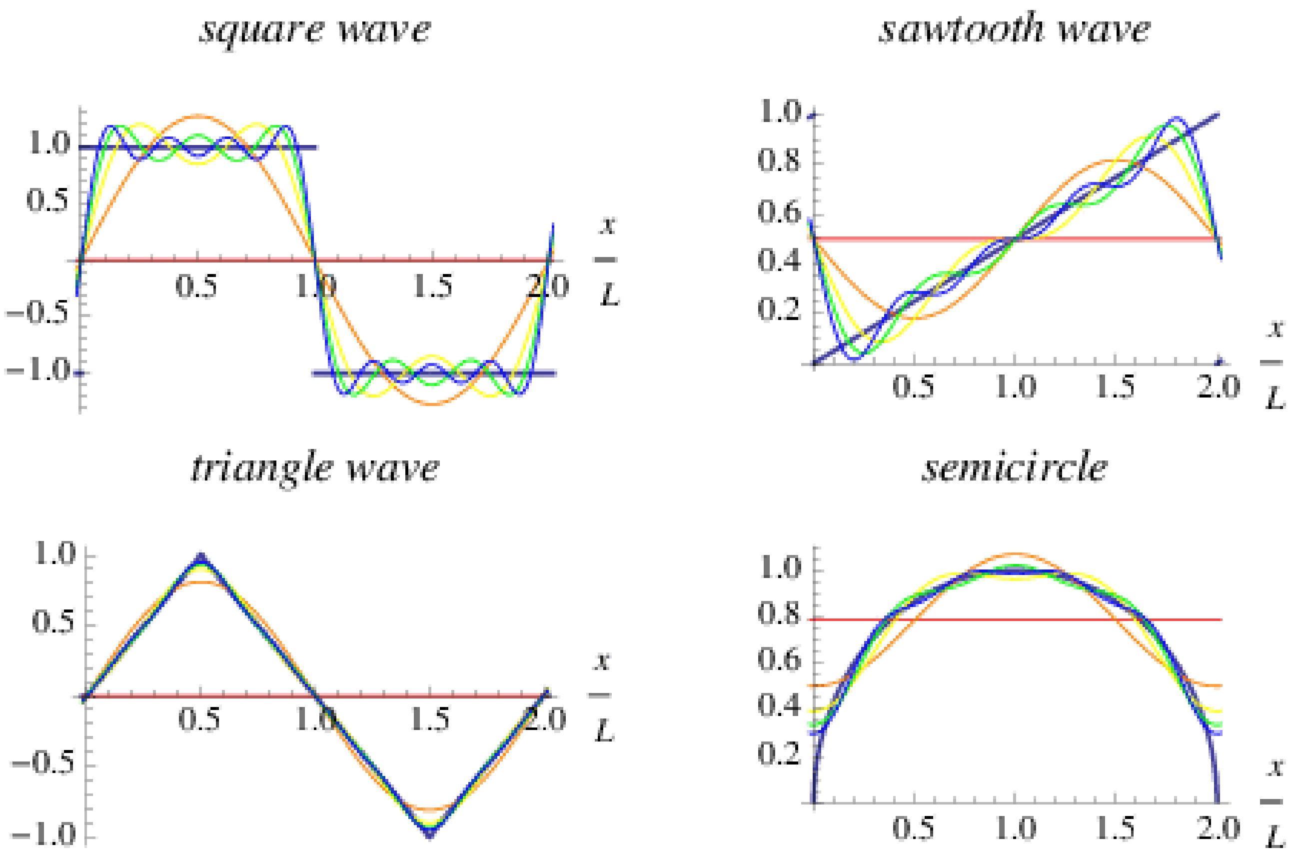


"Essential behaviors such as walking or breathing are rhythmic, that is, characterized by repetitive activation of muscles in a specific temporal sequence. A basic version of the underlying neural activity is generated in the central nervous system by networks called central pattern generators (CPGs). CPGs can produce these patterns even in the absence of inputs that carry specific timing information. The core rhythmic activity can be based on intrinsic oscillatory properties of pacemaker neurons or emerge from the synaptic connectivity of non-oscillatory neurons."

https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470015902.a0000032.pub2

Adding Complexity: Fourier Series

- More complex periodic gait
  - $\theta_i(t) = a_i + b_i * \sin(\omega t + c_i) + d_i * \sin(\mathbf{2}\omega t + e_i) + \ldots$



Combinations of sine waves can create any arbitrary function

Showing the robot how to walk

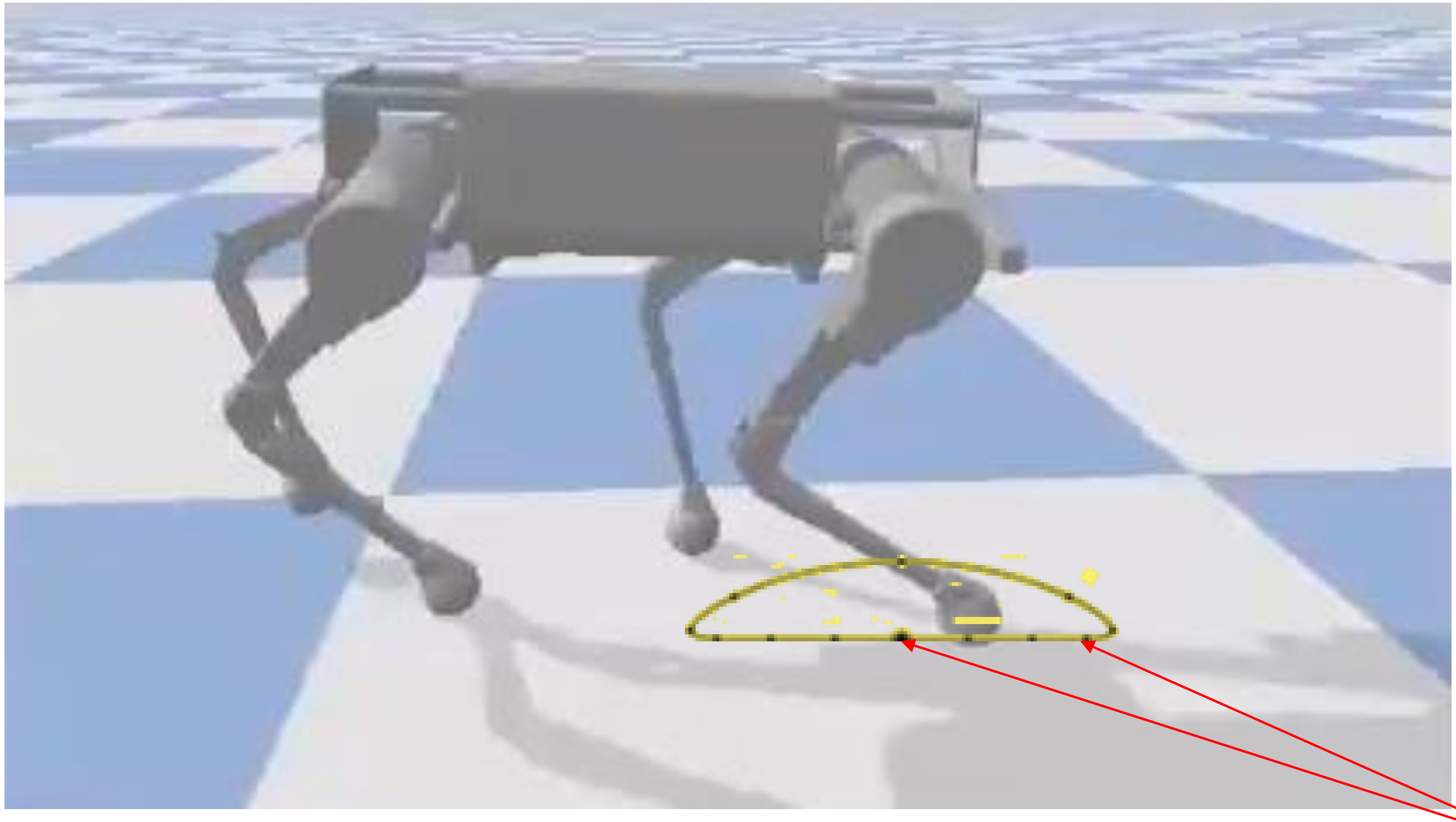
- Slowly move the leg in a desired path
- Record/log the motor positions along the way
- Play back the waypoints slowly
    - Possibly with smooth interpolation
    - Plot the waypoints and interpolation to make sure they make sense

LX16A.getPhysicalPos() # Returns the current physical position of the servo.

You can use **scipy.optimize.curve** to fit a function to data

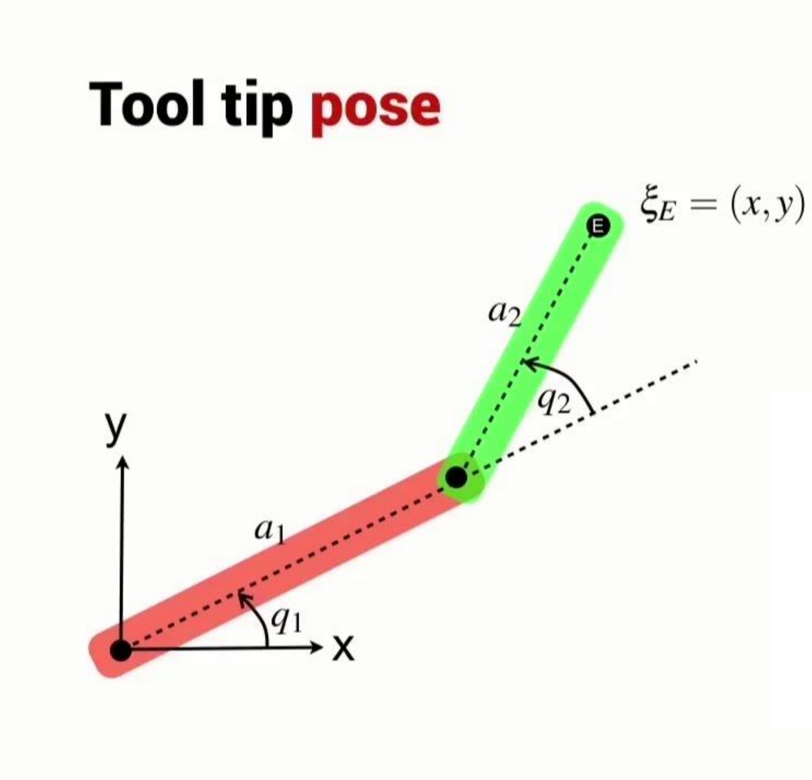
https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.optimize.curve_fit.html

Approach 2: Analytical inverse kinematics



Use inverse kinematics to move each foot in a prescribed trajectory (through manually-prescribed **waypoints**)
Create “motion primitives” for forward motion, backwards, turning, etc.

Kinematics: Tip/Foot position from angles



$$x = a_1\cos(q_1)+a_2\cos(q_1+q_2)$$
$$y = a_1\sin(q_1)+a_2\sin(q_1+q_2)$$

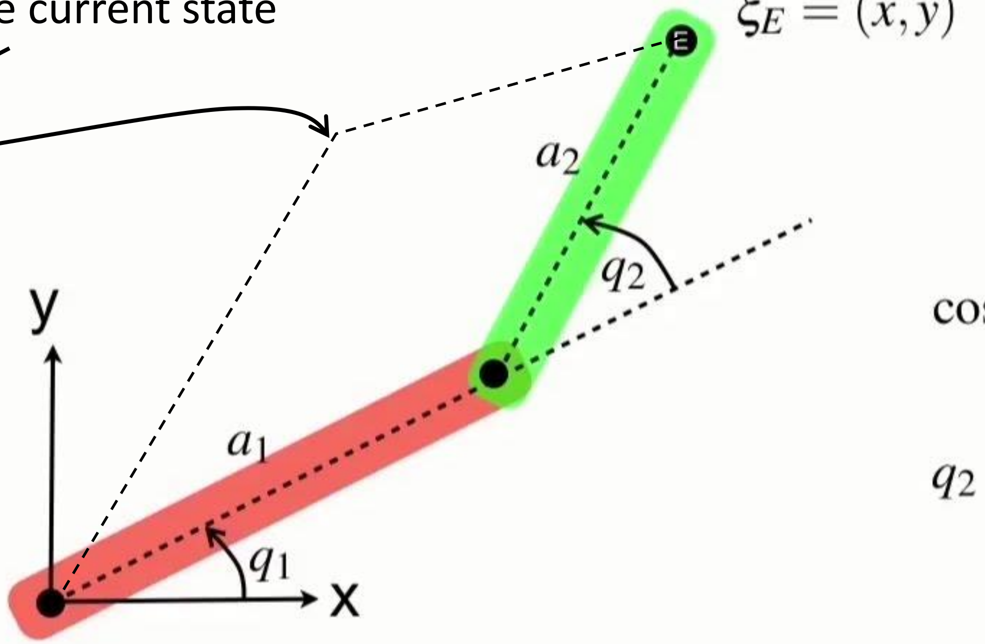
<u>Inverse Kinematics?</u>

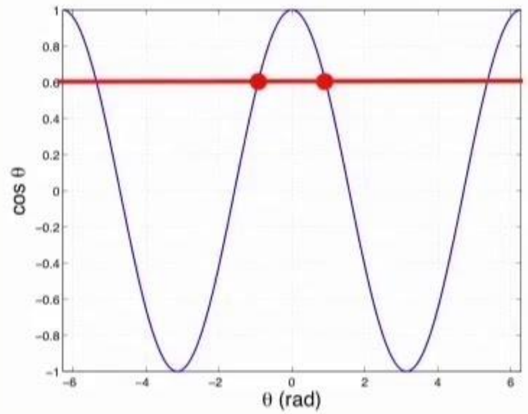
$q_1=f_1(x,y)$

$q_2=f_2(x,y)$

Inverse Kinematics: Angles from tip position

There may be multiple solutions. Make sure you choose the solution that is closest to the current state of the leg





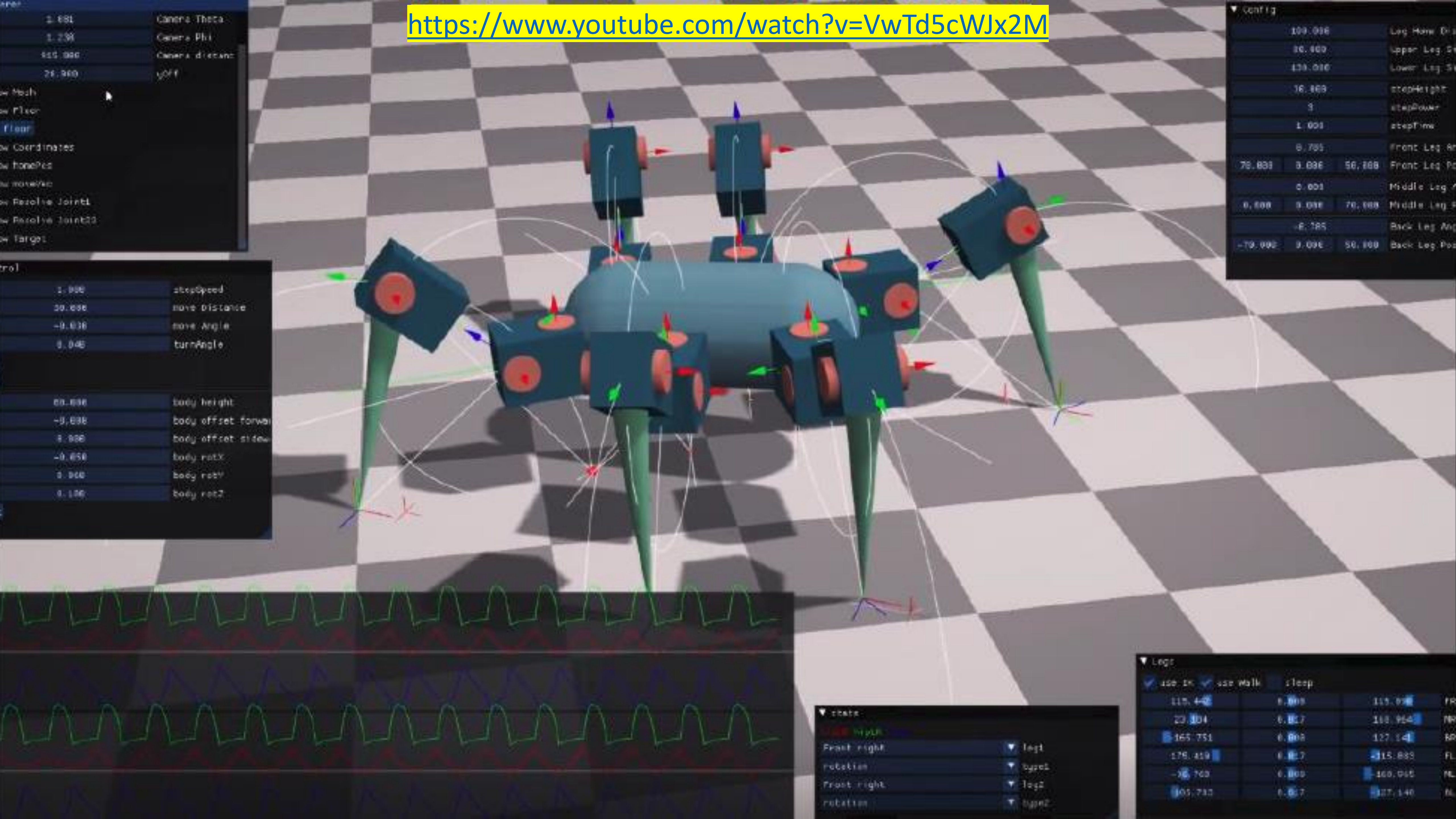
$$\cos q_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2}$$

$$q_2 = \cos^{-1} \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2}$$

$$q_1 = \tan^{-1} \frac{y}{x} - \tan^{-1} \frac{a_2 \sin q_2}{a_1 + a_2 \cos q_2}$$

https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/

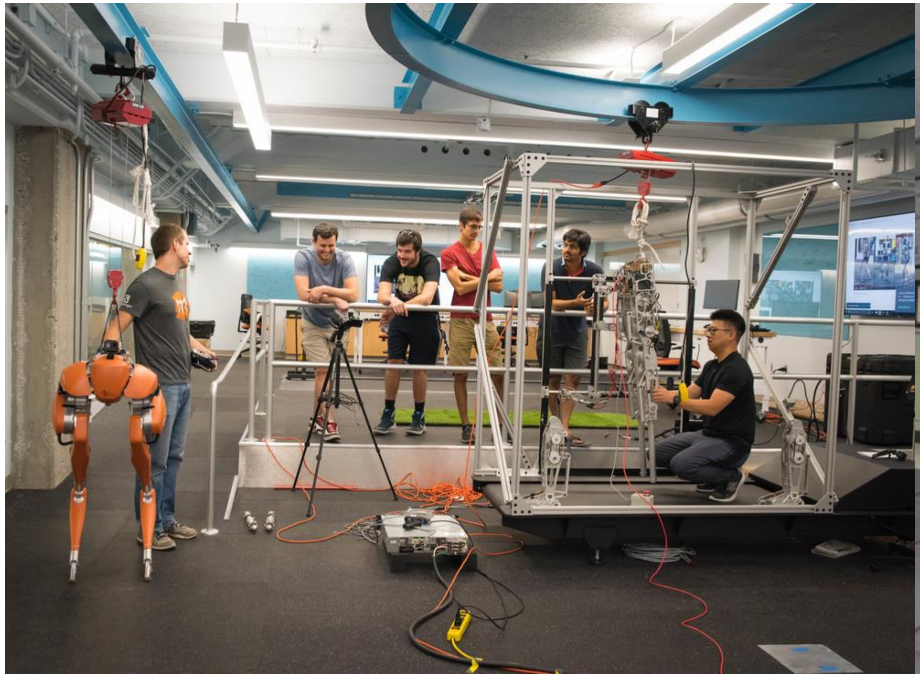
https://www.youtube.com/watch?v=VwTd5cWJx2M




CORSAIR

Assistance:

- Add Harness
  - also helps with stabilization




Robot boot comm/power self test

- When the robot starts up, test it automatically
- Create an autotest procedure
  - Query the motor positions
    - Report an error if you cannot get a reply from any motor (comm error, ID error)
  - Enable/disable each motor
    - Make sure motors have access to power
  - Query motor power
    - Make sure it has enough voltage
  - Flash each robot LED three times
    - Provides visual confirmation that everything's powered and hooked up correctly
    - Use a logical sequence (e.g. left leg then right leg, top to bottom)

Report any errors as text messages, sounds (bleeps), or flashing of motor LEDs

Homing initialization

- When the robot starts up, it may be in undefined motor positions
- Create an initialization procedure
  - Query the motor positions
    - Report an error if you cannot get a reply from the motor (comm error)
  - Move them slowly to their initial “home” positions
    - Report an error temp/current is too low or high
  - Check that robot reaches its home state
    - Report an error if motors don’t reach their target position

Report any errors as text messages, sounds (bleeps), or flashing of motor LEDs

Shutdown procedure

- When the robot ends, put it in a sleep/crouch/safe position
- Create a shutdown procedure
  - Query the motor positions
    - Report an error if you cannot get a reply from any motor
  - Move them slowly to their safe positions
    - Report an error if motors temp/current is too low or high
  - Check that motors get there
    - Report an error if motors don't reach their target position

Report any errors as text messages, sounds (bleeps), or flashing of motor LEDs

Continuous health check

- Once in a while (e.g. at key frames), check status of robot
- Create a health check procedure
  - Query motor positions
    - Check motors are roughly where they are supposed to be – not stuck
  - Query motor temp
    - Check motors are not overheating
  - Query motor current
    - Check motor currents are not too low (fall over) or too high (stuck)
  - Query IMU (if available)
    - Check robot has not fallen over/tipped
  - Comm/Checksum error?
    - Resend message a few times; abort of repeated checksum errors.

Report any errors as text messages, sounds (bleeps), or flashing of motor LEDs

Reminders…

Final presentation (5-8 min)

- Last class
- Show your project from start to finish
  - Concept sketch (images)
  - CAD rendering (video)
  - Working leg (video)
  - Walking robot (video)
  - Walking Simulation (video)
- Embed all videos
  - Videos should start automatically
  - Video file should be embedded (work without internet access)
    - Test by running your presentation in “airplane mode”

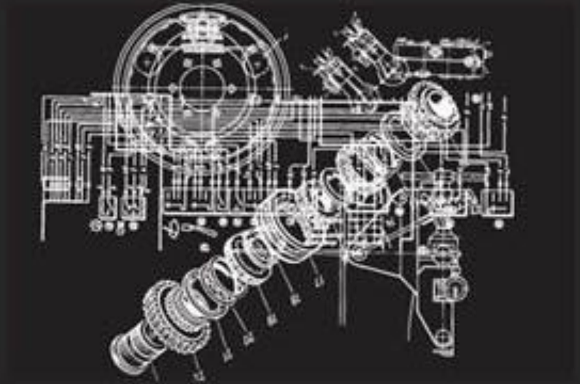
Get points for creating a…


15:55
88 %
MINGU
KIM
MECHANICAL
ENGINEER
RESUME
PORTFOLIO

Portfolio

PROJECT LOREM IPSUM I

SHOW DETAILS >



DEMOAN TULIO

SHOW DETAILS >

ARMEK TIOLA TUBAS REM

SHOW DETAILS >