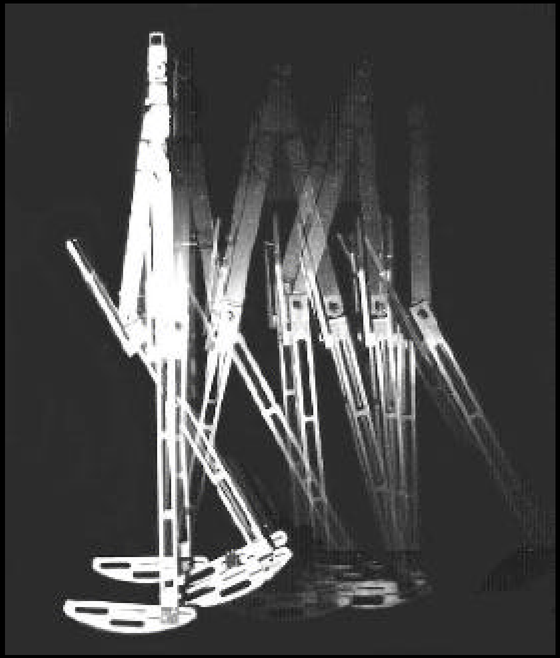
CS148 - Building Intelligent Robots
Lecture 4: Control Theory and Robot Dynamics

Instructor: Chad Jenkins (cjenkins)

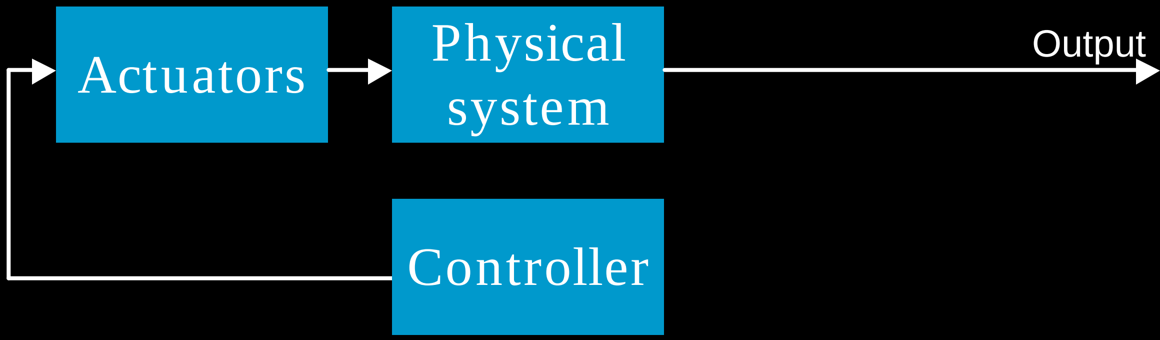
Types of control

- Passive Control
  - no actuation or under-actuated
  - structurally modify the plant dynamics
  - use when viable: cheap, robust



A. Ruina/Cornell

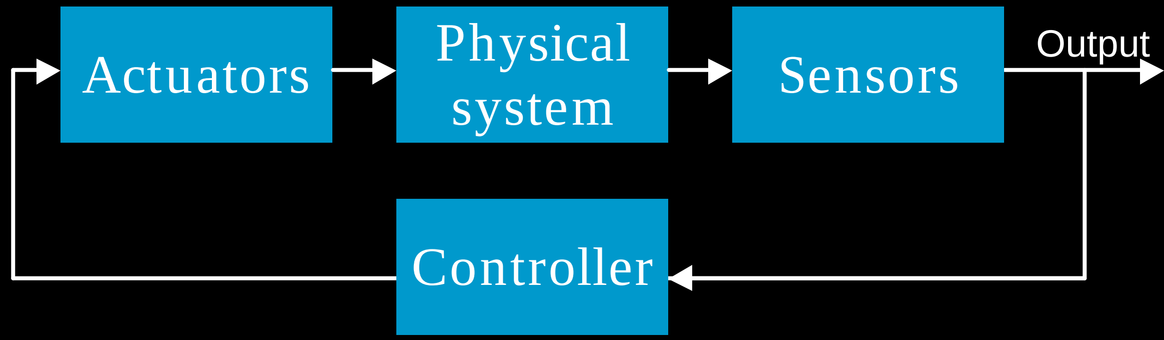
Types of control



- Passive Control

- Open Loop Control
  - actuation without sensing
  - exploit knowledge of system dynamics to compute appropriate inputs
  - requires *very* accurate model of plant dynamics

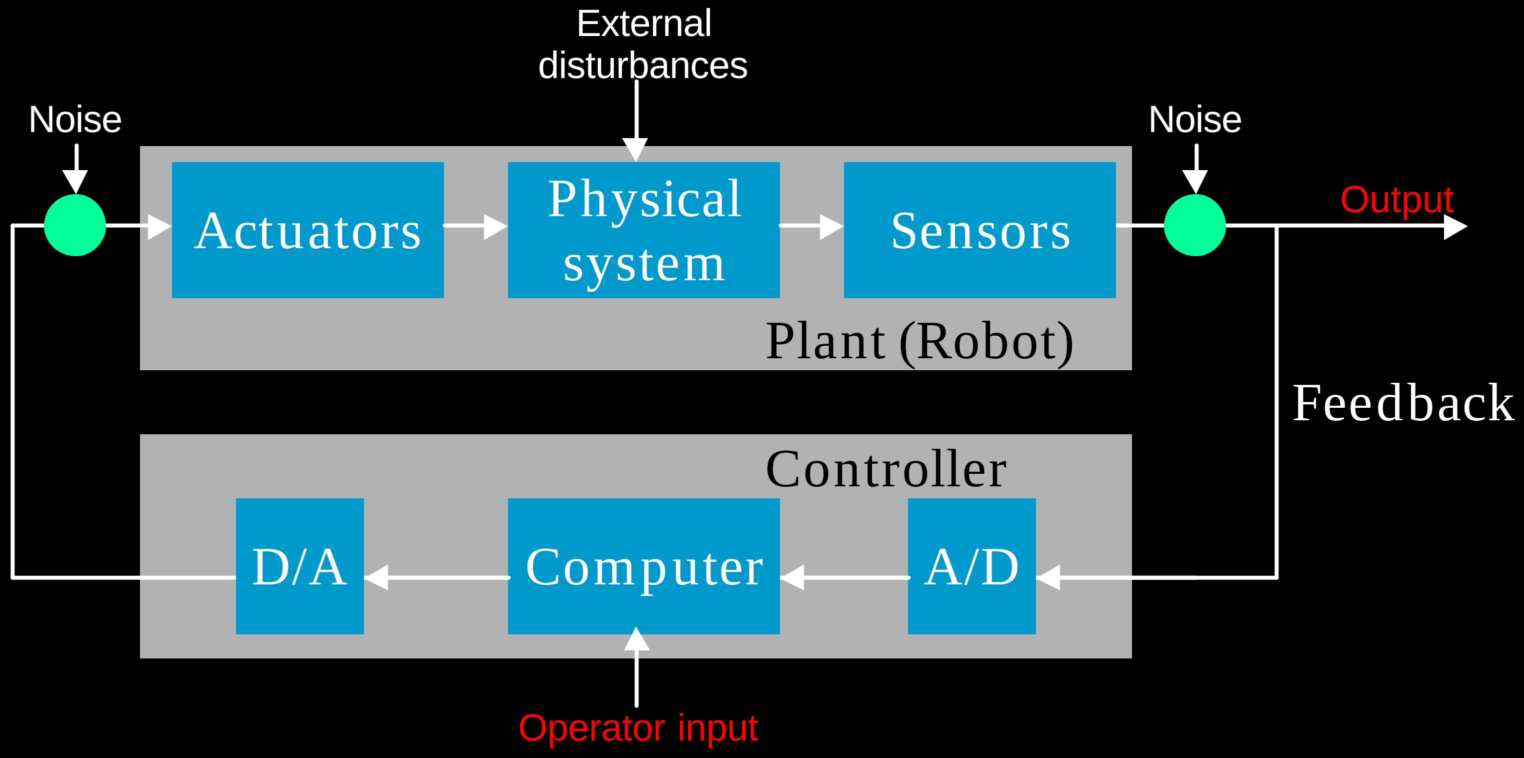
Types of control



- Passive Control
- Open Loop Control
- Active (Feedback) Control
  - autonomous robotics
  - use sensors and actuators connected by a computer to modify dynamics
  - allows for modeling of uncertainty and noise

Modern control system components

- Plant, controller, and feedback
- Modeling through control theory



Defining control theory (from Wikipedia)

- Control theory:
  - deals with the behaviour of dynamical systems over time.
  - a controller tries to manipulate the inputs of the system to realize desired behaviour at the output of the system.

- Dynamical system:
  - a deterministic process in which a function's value changes over time according to a rule that is defined in terms of the function's current value.

Defining control theory

- Controlled dynamical systems consist of
  - a next-state equation (f specifies change in state)

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$$

  - an output equation (g specifies what is observed from state)

$$\mathbf{y} = g(\mathbf{x}, \mathbf{u}, t)$$

  - x = the (internal) state of a system
    - the space of possible states is called the state space
  - y = the observation variable
  - u = control input from a control system
    - specified as a control policy: $\mathbf{u} = \pi(\mathbf{x}, \alpha, t)$

Representing time

- Discrete dynamical systems
  - time is measured in discrete steps
  - system is modeled as a recursive relationship

$$\mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n])$$
$$\mathbf{y}[n] = g(\mathbf{x}[n])$$

- Continuous dynamical systems
  - time is measure continuously
  - system is expressed as an ordinary differential equations

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{y} = g(\mathbf{x}, \mathbf{u}, t)$$

- Linear systems

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

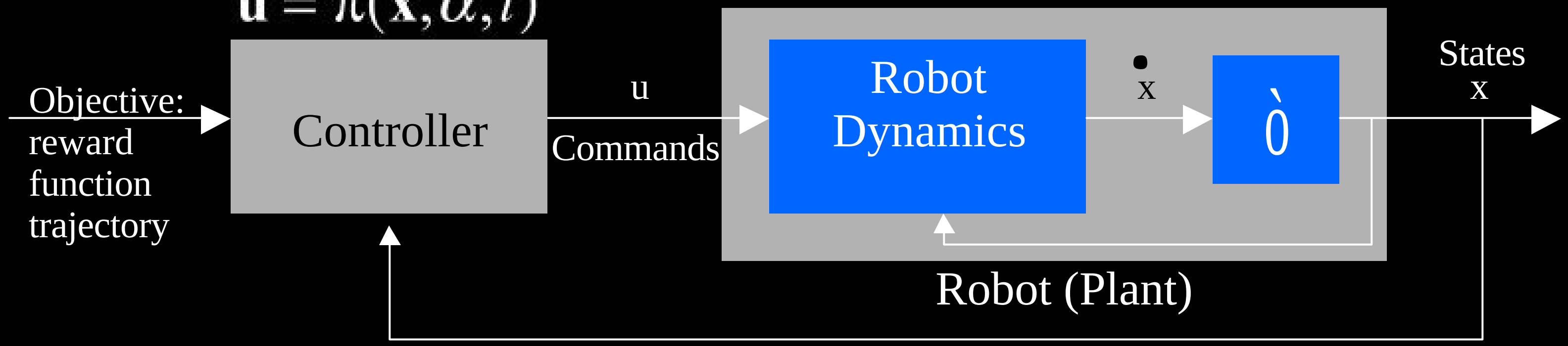
Brown Computer Science

Controllers and control theory

- Forward dynamics: $\mathbf{y}[n+1] = h(\mathbf{x}[n], \mathbf{u}[n])$
- Inverse dynamics: $\mathbf{u}[n] = h^{-1}(\mathbf{x}[n], \mathbf{y}[n+1])$
- Given sensing and actuation platform, provide control policy: $\mathbf{u} = \pi(\mathbf{x}, \alpha, t)$
  - function producing control commands from current state

$$\mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n])$$

$$\mathbf{u} = \pi(\mathbf{x}, \alpha, t)$$



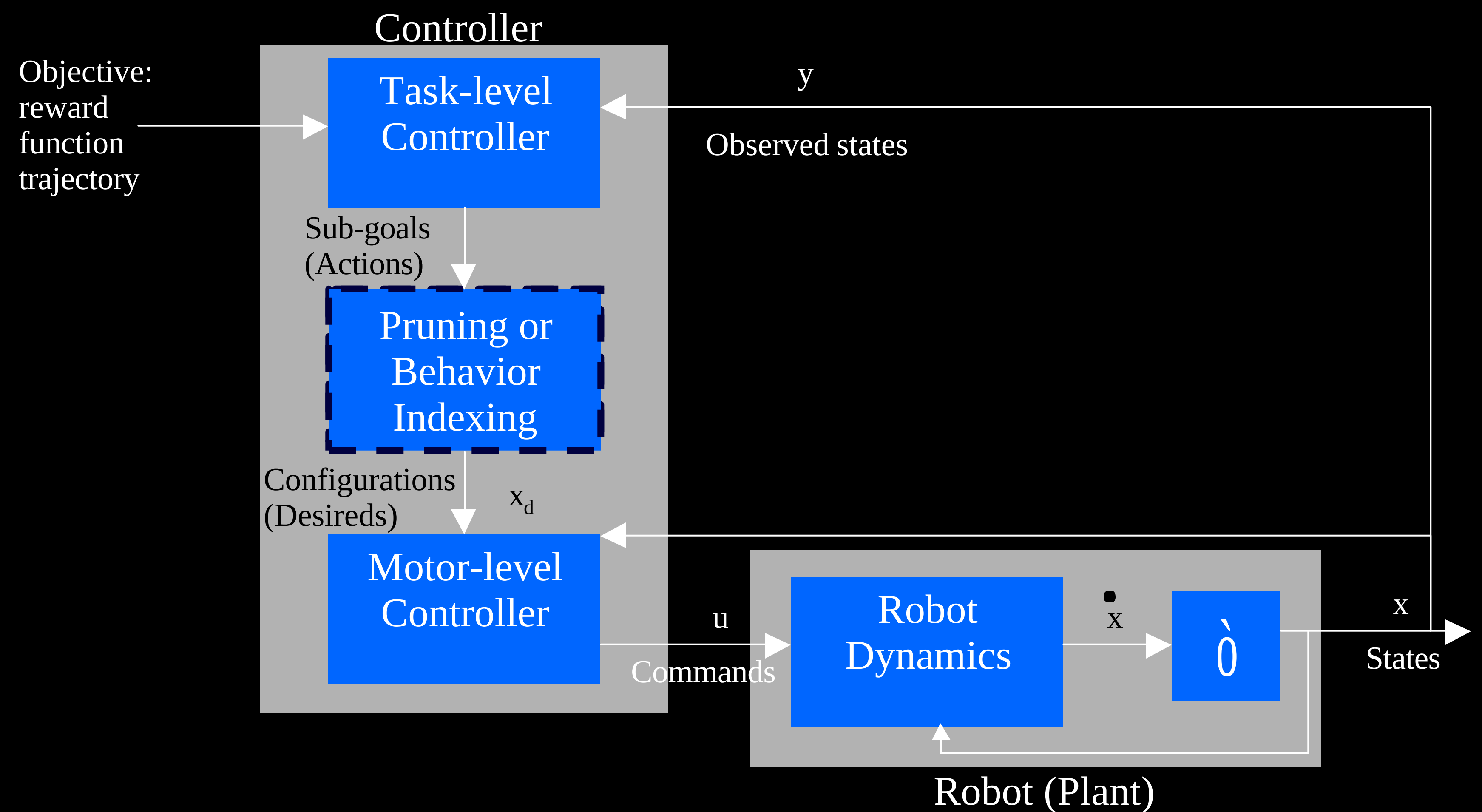
For partial state observability $\mathbf{y}[n] = g(\mathbf{x}[n])$

Control factors

- Stability: bounding the transient behavior of the system
  - preventing instability by bounding inputs (u)
- Controlability: the ability to use a system's external inputs (u) to manipulate its internal state (x)
- Observability: the ability for a system’s internal states (x) to be inferred from its external outputs (y)
- Minimality: a minimal system is both controlable and observable

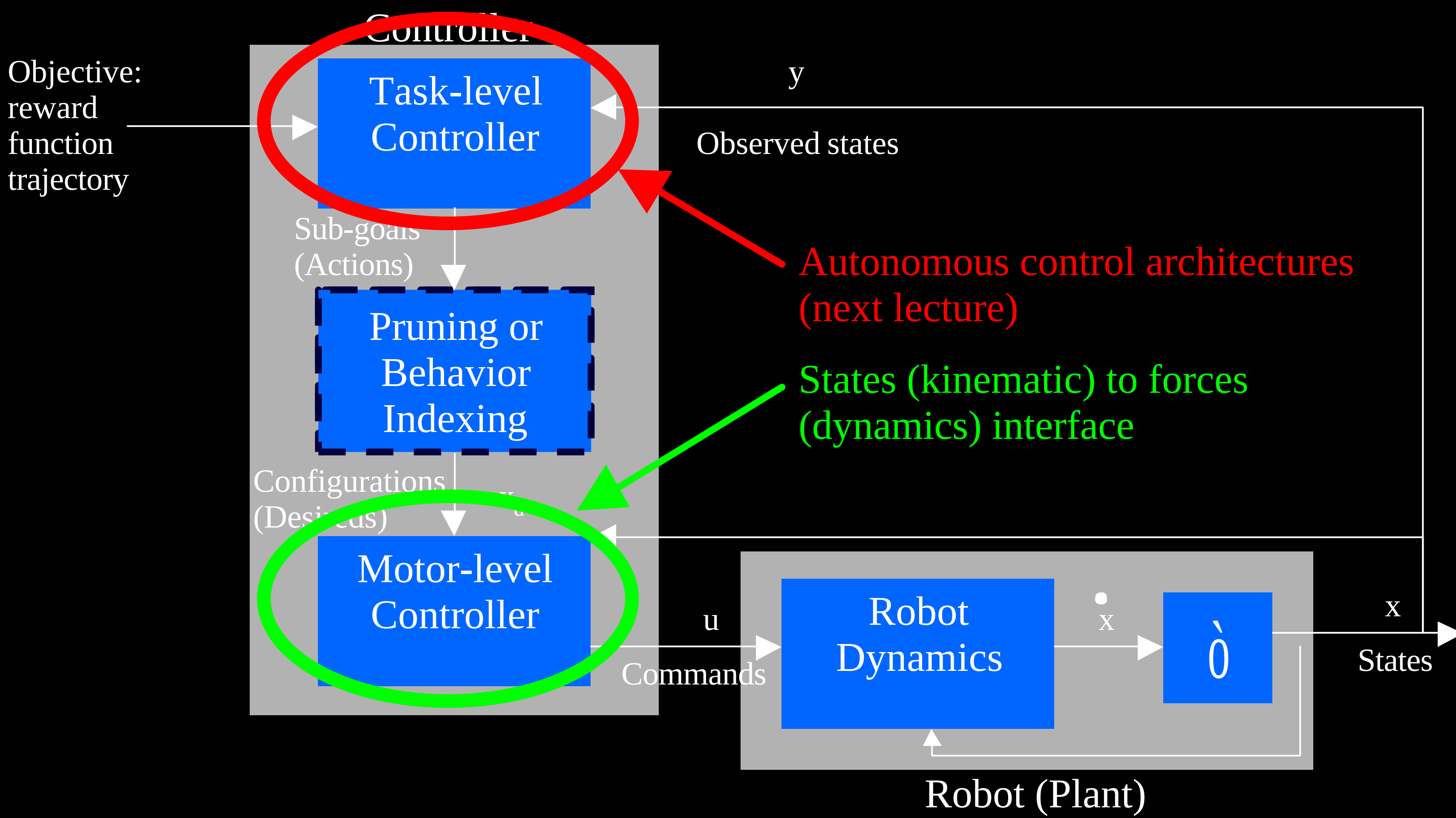
Autonomous controllers, in actuality

- Increasing complexity in DOF requires more sophisticated controllers



Autonomous controllers, in actuality

- Increasing complexity in DOF requires more sophisticated controllers



Motor-level control

- Produce actuation commands (u) that will produce desired states (x_d)
  - $u = f^{-1}(x,x_d)$
- Model inverse dynamics:
  - Derive equations of motion for the robot
  - Approaches: Lagrangian, Newton-Euler, learning (regression)
- Can leverage robot dynamics: feedback control
  - PID Servoing
- Inverse kinematics changes control: $u = f^{-1}(x,y_d)$
- Combinations of feedback and feedforward control

Equations of motion

$$\underline{Q} = \mathbf{M}(\underline{q})\underline{\ddot{q}} + \mathbf{C}(\underline{q}, \underline{\dot{q}})\underline{\dot{q}} + \mathbf{F}(\underline{\dot{q}}) + \mathbf{G}(\underline{q})$$

Friction

Gravity

Coriolis and centripetal effects

acceleration

Inertia matrix (configuration dependent)

Generalized forces

- Generalized coordinates (q) completely describes the system (e.g., position and orientation)
- Forward dynamics: integrate equations using forces
- Inverse dynamics: solve equations for forces

Lagrangian dynamics

- Potential function

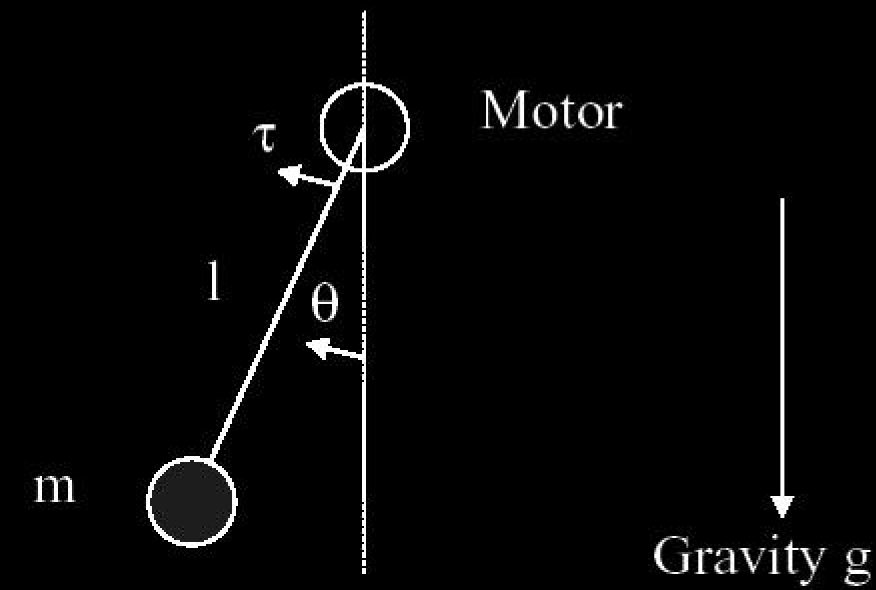
$$L = T - U$$

  - kinetic energy minus potential energy

- Differentiate for equations of motion

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = \tau_i$$

- Pendulum example



- Kinetic Energy

$$T = \frac{1}{2} I \dot{\theta}^2$$

- Potential Energy

$$U = mgl(1 - \cos\theta)$$

- Lagrangian

$$L = T - U$$

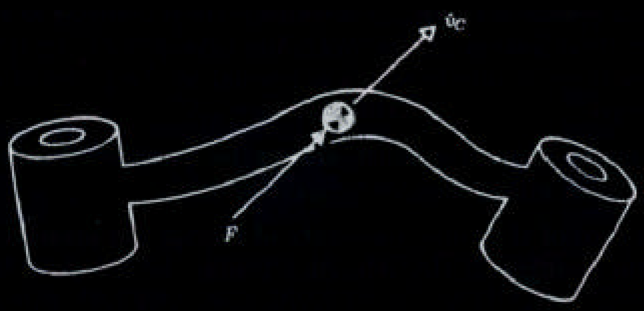
- Equation of Motion

$$I\ddot{\theta} + mgl\sin(\theta) = \tau$$

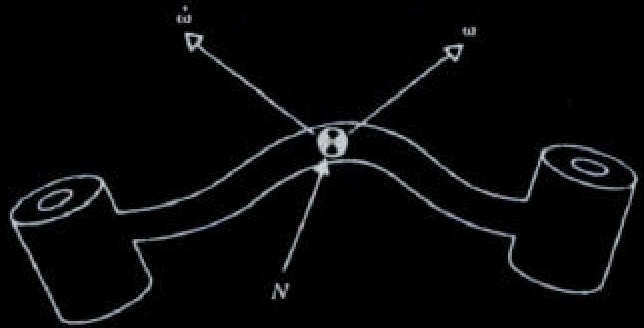
Newton-Euler dynamics

- Lagrangian formulation is simple, but
  - kinetic energy can be difficult to calculate
  - computation can be expensive

- Newtons second law $F = m\dot{v}_C$
  - relate linear force to linear acceleration

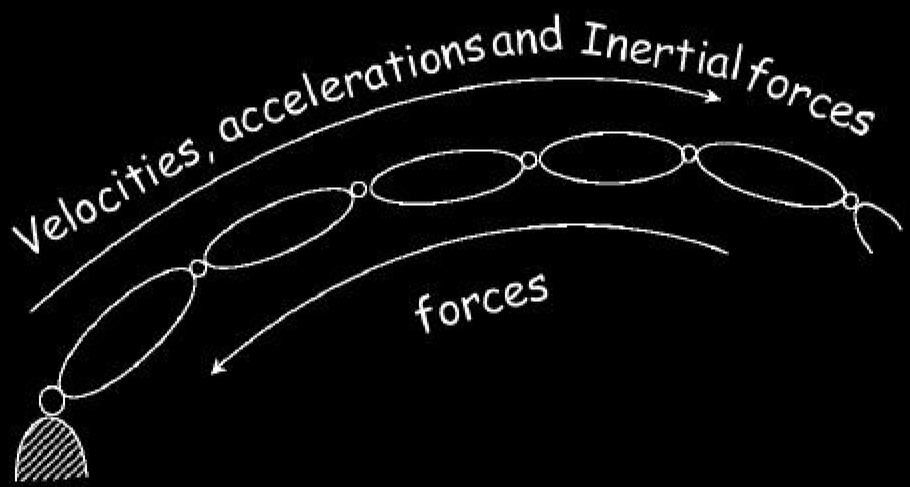


- Euler's equation $N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega$
  - relate torque to angular velocity



Advantage of Newton-Euler

- Recursive algorithm
  - forward: propagate velocity and acceleration forward
  - backward: return forces
- Enables real-time forward and inverse dynamics



Modeling vs. leveraging dynamics

- Modeling dynamics is suited to inverse dynamics
  - open loop or feedforward control
  - predictability of system
  - how accurate is your model?
  - how much time does it take to compute?
- Leveraging dynamics is suited for feedback control
  - decrease the error between actual and desired configurations
  - PID control

P-Servoing

- Position-servo: produce force that reduces error

$$t = \mathbf{K}_P\left(\mathbf{x}_{des}(t) - \mathbf{x}(t)\right)$$

PD-Servoing

- Position-servo: produce force that reduces error

$$t = \mathbf{K}_P\left(\mathbf{x}_{des}(t) - \mathbf{x}(t)\right)$$

- PDerivative-servo: damping to release energy and reduce oscillation

$$t = \mathbf{K}_P\left(\mathbf{x}_{des}(t) - \mathbf{x}(t)\right) + \mathbf{K}_D\left(\dot{\mathbf{x}}_{des}(t) - \dot{\mathbf{x}}(t)\right)$$

PID-Servoing

- Position-servo: produce force that reduces error

$$\tau = \mathbf{K}_P(\mathbf{x}_{des}(t) - \mathbf{x}(t))$$

- PDerivative-servo: damping to release energy and reduce oscillation

$$\tau = \mathbf{K}_P(\mathbf{x}_{des}(t) - \mathbf{x}(t)) + \mathbf{K}_D(\dot{\mathbf{x}}_{des}(t) - \dot{\mathbf{x}}(t))$$

- PIntegralD-servo: eliminate steady state error

$$\tau = \mathbf{K}_P(\mathbf{x}_{des}(t) - \mathbf{x}(t)) + \mathbf{K}_D(\dot{\mathbf{x}}_{des}(t) - \dot{\mathbf{x}}(t)) + \mathbf{K}_I \int_{\tau=0}^{\tau=t} (\mathbf{x}_{des}(t) - \mathbf{x}(t))dt$$

  – compute integral term over recent horizon

Brown Computer Science

Issues in robot programming

- Real-time programming for dynamic environments
- What separates a general robotics from a chess player or a robotic chess player
  - real time demands
- controller must be fast enough for environment
- data to and from the robot and the controller

Additional References

- tld's notes from 2002 CS148
  - http://www.cs.brown.edu/courses/cs148/old/2002/
- M.I. Jordan, "Computational Aspects of Motor Control and Motor Learning"
- P. I. Corke, "Robotics Toolbox for Matlab"
- S. Schaal's robotics notes
  - http://www-clmc.usc.edu/
- Murray's controls tutorial
  - http://www.cds.caltech.edu/~murray/courses/primer-fa01

Brown Computer Science

Additional References

- Sabino's control theory primer
  - http://www.caam.rice.edu/~jnsabino/work/651pres.pdf
- D. Thalmann, "Robotics Methods for Task-level and Behavioral Animation"
- Witkin and Baraff's rigid body dynamics notes
  - http://www.pixar.com/companyinfo/research/pbm2001/