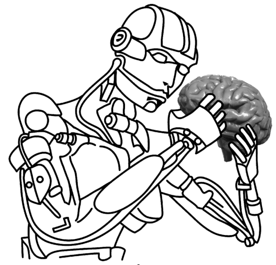
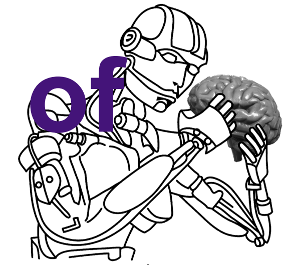
CS545—Contents XIII

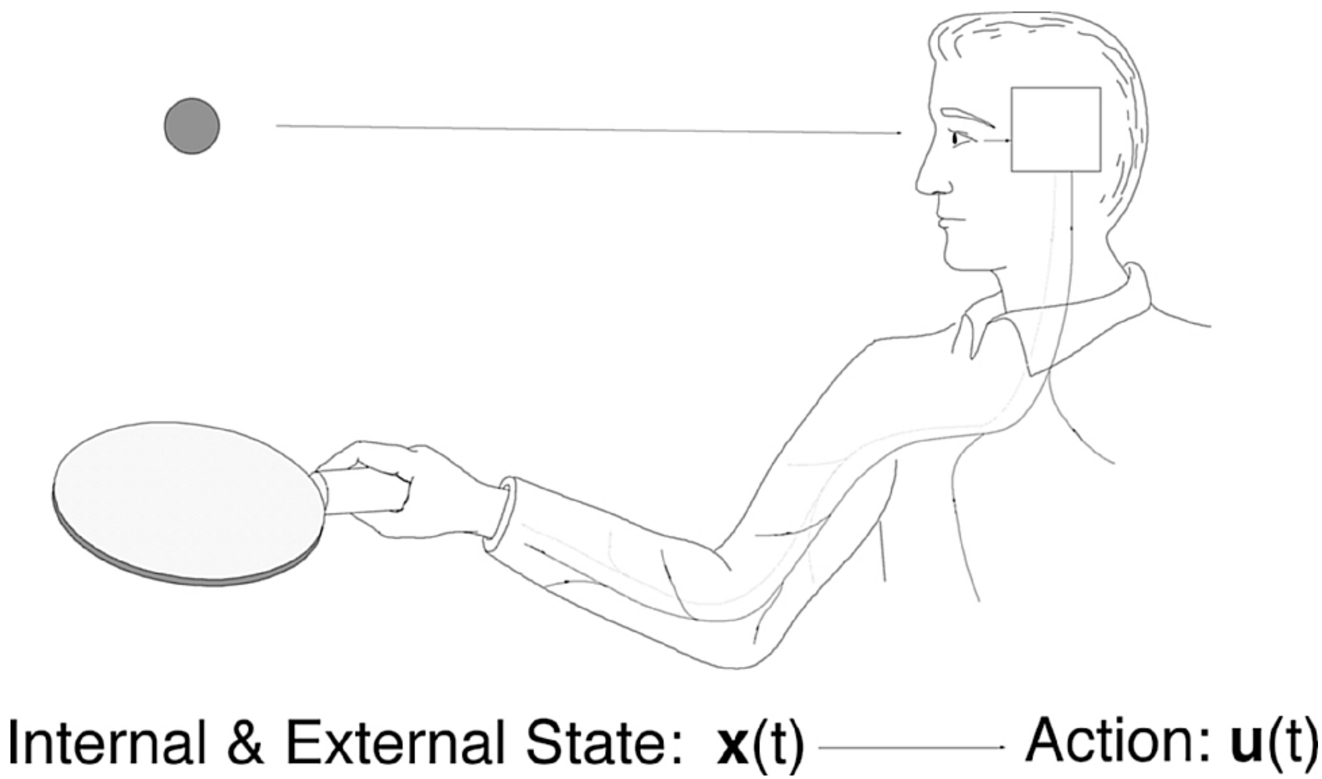


- Trajectory Planning
  - Control Policies
  - Desired Trajectories
  - Optimization Methods
  - Dynamical Systems
- Reading Assignment for Next Class
  - See http://www-clmc.usc.edu/~cs545

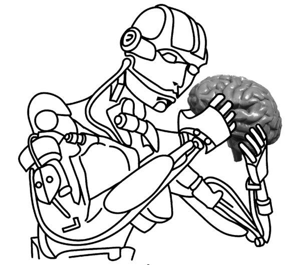
Learning Policies is the Goal of Learning Control

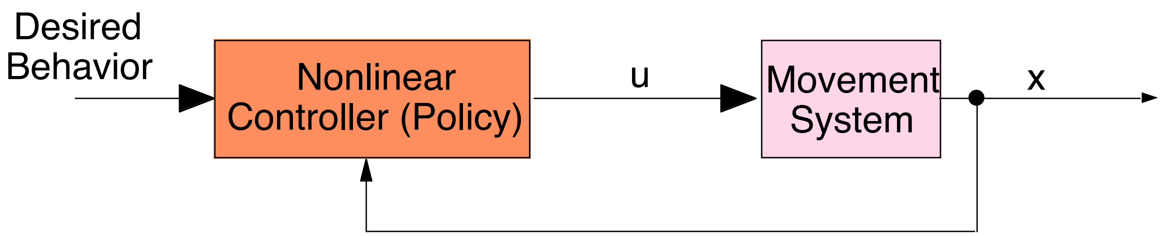


- Policy: $$\mathbf{u}(t) = p(\mathbf{x}(t), t, \alpha)$$



Dynamic Programming & Reinforcement Learning



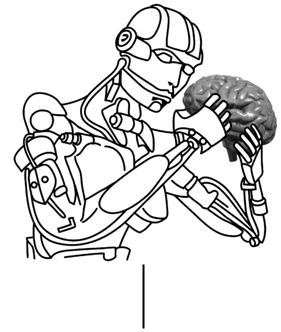


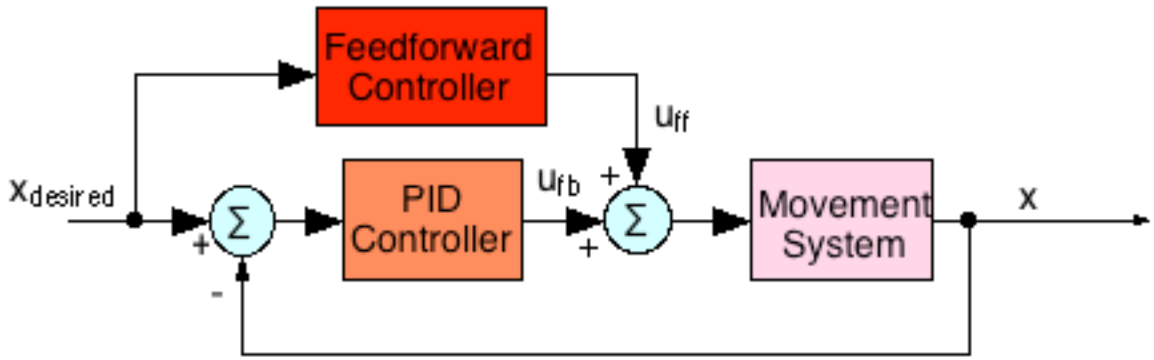
(HJB-Eqn.)

$$V = \max_{\mathbf{u}} \left[r(\mathbf{x},\mathbf{u}) + \tau \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x},\mathbf{u}) \right]$$

- Dynamic Programming
  - requires a model of the movement system
- Reinforcement Learning
  - can work without models of the movement system
- Essentials
  - both techniques require to learn a high-dimensional “value function” that assesses the quality of an action **u** in a state **x**
  - learning the value function is a complex nonstationary, nonlinear learning process
  - both methods die the curse of dimensionality

Desired Trajectories





- Essentials
  - prescribe a desired trajectory

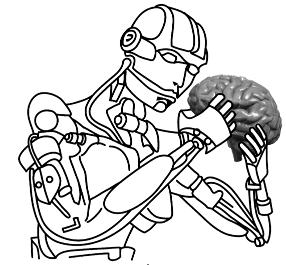
$$\left(\theta,\dot{\theta}\right)_{desired} = f\left(\xi_{initial},\xi_{target},t\right)$$

  - convert desired trajectory into a (time-dependent) control policy, e.g., by PD-controller

$$\mathbf{u} = p\left(\mathbf{x},t,\alpha\right) = \mathbf{k}_{\theta}\left(\theta\left(t\right)_{desired} - \theta\right) + \mathbf{k}_{\dot{\theta}}\left(\dot{\theta}\left(t\right)_{desired} - \dot{\theta}\right)$$

- Problems
  - Where do desired trajectories come from
  - How to accomplish reactive control
  - How to generalize to new tasks or new situations

Desired Trajectories (cont'd)



- There is a difference between PATH and TRAJECTORY planning
  - A trajectory involves geometry AND time
  - A path involves only geometry
- Planning can happen either in joint or operational space

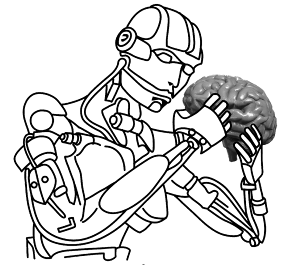
$$\mathbf{x}_d = g(t, \alpha)$$

$$or$$

$$\theta_d = f(t, \alpha)$$

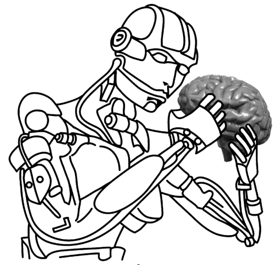
- There is usually an infinity of possible desired trajectories
- How is the desired trajectory represented?
  - Every point in time?
  - Only start & final point?
  - Via points?
- Movement Primitives

Joint Space Planning



- What could one plan?
  - Arbitrary trajectories from start to end
  - Trapizoidal (or any aother kind of) velocity profiles
  - Polynomials:
    - 1.order: straight lines
    - 2.order: parabolas
    - 3.order: cubic splines
    - 5 order: quintic splines
    - Interesting:
      - Analyze the shape of the trajectories in position, velocity, acceleration, and jerk space.
      - How many constraints are needed to specify a trajectory

Example:Cubic Polynomial



- Cubic Polynomial:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$$
$$\ddot{q}(t) = 2a_2 + 6a_3 t$$

- Given: Start & Endpoint

$$q_s, q_f$$

- Plan a cubic polynomical through the start and endpoint
  - Two additional constraints are needed, for instance:

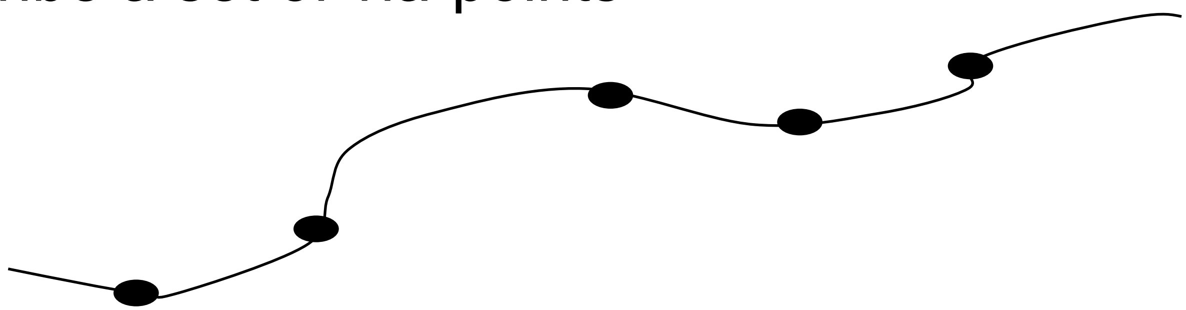
$$\dot{q}_s, \dot{q}_f \quad \text{or} \quad \dot{q}_s, \ddot{q}_s \quad \text{or} \quad \dot{q}_f, \ddot{q}_f$$

  - Determine the coefficients by using 4 boundary conditions, e.g.,

$$q_s = a_0$$
$$\dot{q}_s = a_1$$
$$q_f = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$\dot{q}_f = a_1 + 2a_2 t + 3a_3 t^2$$

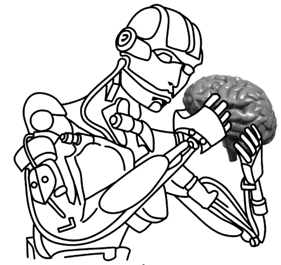
Planning Complex Paths

- Prescribe a set of via-points



- Plan simple trajectories between via-points
- Ensure smooth transitions between trajectory segments
  - E.g., the tangent of two adjacent trajectory segments should match

Optimization Approaches to Desired Trajectories

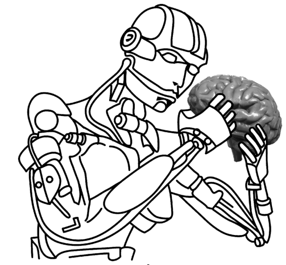


- Given:
  - "hard constraints", e.g., q_s, q_f, t
  - "soft constraints", i.e., an optimization criterion

$$J = \int_0^{\tau} g(\mathbf{q}, \dot{\mathbf{q}}, \ldots) dt$$

- Goal:
  - Find the trajectory that fulfills the hard constraints while minimizing (or maximizing) the soft constraint
- Solution Methods:
  - Calculus of Variation
  - Dynamic Programming

Optimization Approaches Examples



- Minimum kinetic energy

$$J = \int_0^{\tau} \dot{q}^2 dt$$

  - Results in a quadratic polynomial as solution
- Minimum Jerk

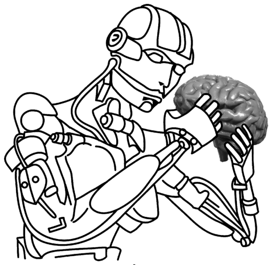
$$J = \int_0^{\tau} \dddot{q}^2 dt$$

  - Results in a qunitic polynomial as solution
- Minimum Torque Change

$$J = \int_0^{\tau} \dot{u}^2 dt$$

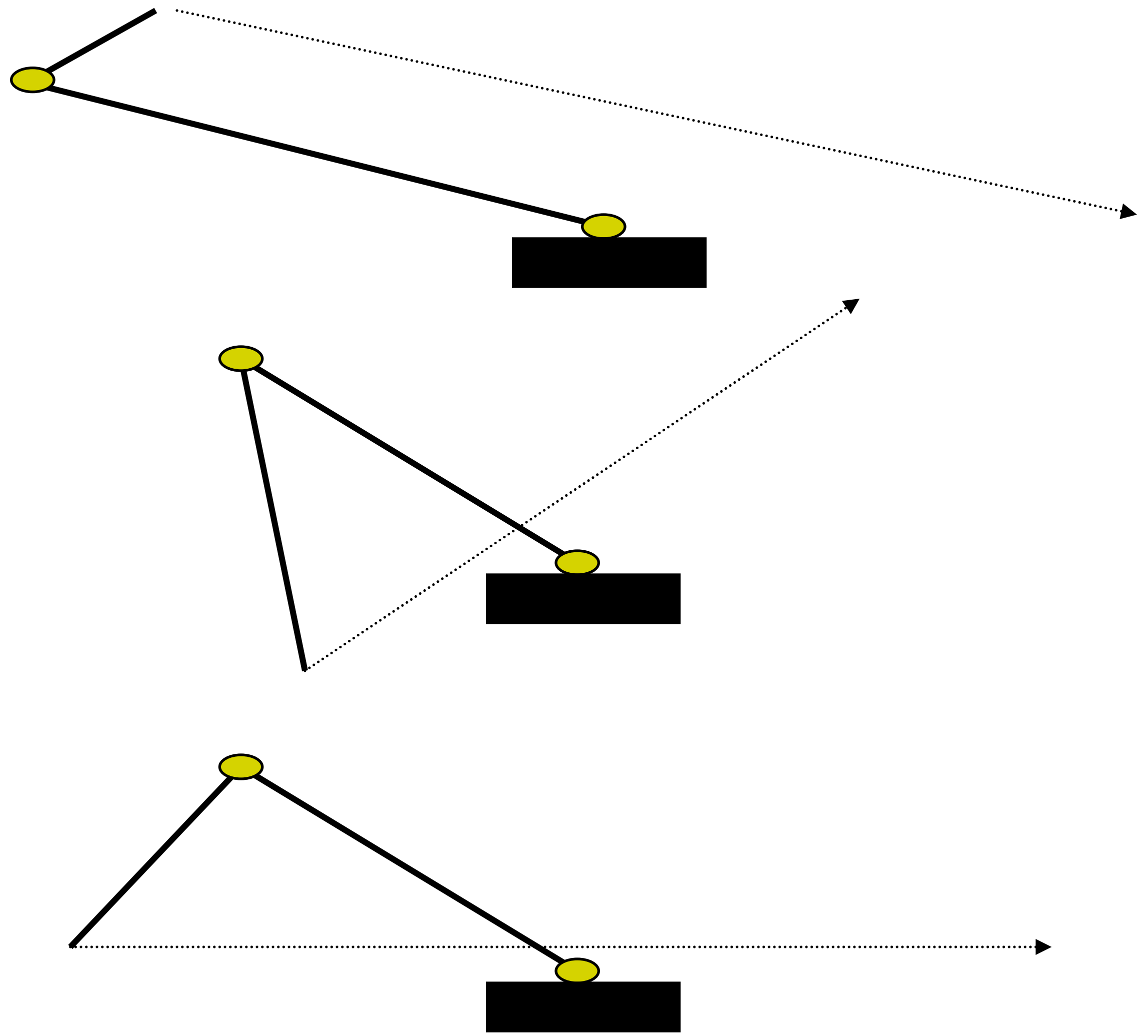
  - Results in something that does not have an analytical description

Operational Space Planning

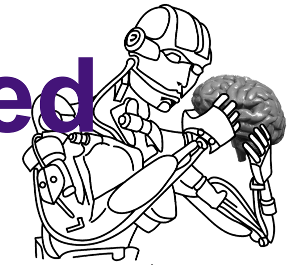


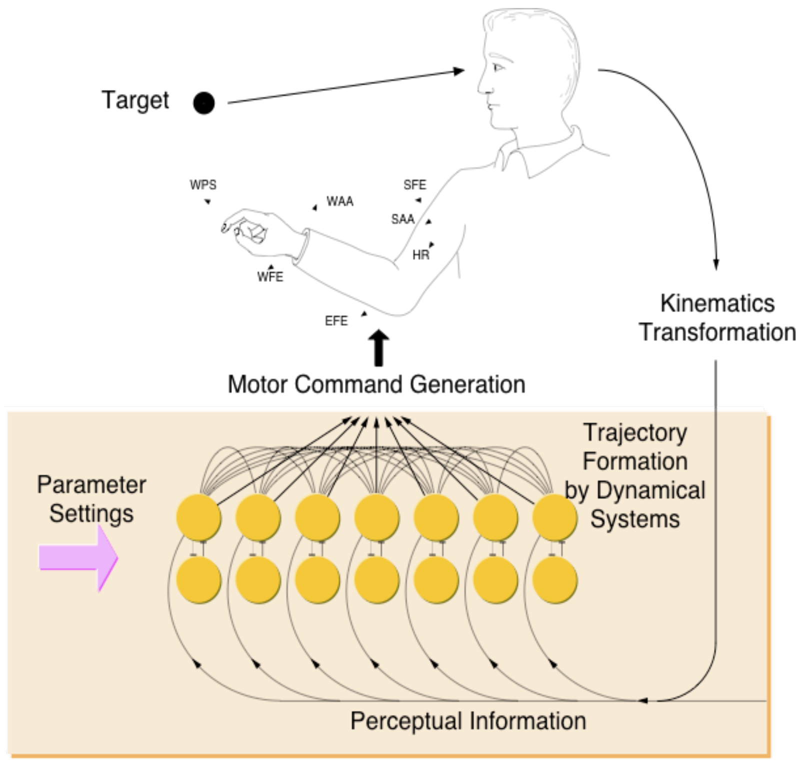
- All joint space planning methods can also be used in operational space
- Inverse kinematics is needed to convert operational space trajectories into joint space
- The resulting joint space motion is usually quite complex
- Geometric problems can arise:
  - Intermediate points are unreachable
  - High joint space motion near singular postures
  - Start and goal reachable in different solutions

Examples of Geometric Problems



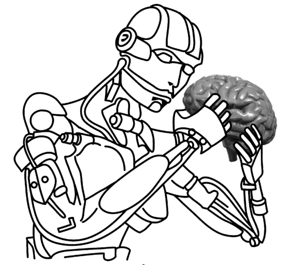
Pattern Generators for Desired Trajectories





- Use Pattern Generators to Create Kinematic Trajectory Plans
  - Use open parameters in pattern generator to generate different movement durations and target settings

Pattern Generators for Trajectory Planing



- What is a pattern generator?
  - A dynamical system (differential equation) with a particular behavior
    - E.g.: Reaching movement can be interpreted as a point attractive behavior:

$$\dot{q}_d = \underbrace{\alpha}_{\text{Speed}}\left(\underbrace{q_f}_{\text{Target}} - q_d\right)$$

- What is the advantage of a pattern generator?
  - Independent of initial conditions
  - Online planning
  - Online modification through additional “coupling” terms. i.e., planning can react to sensory input

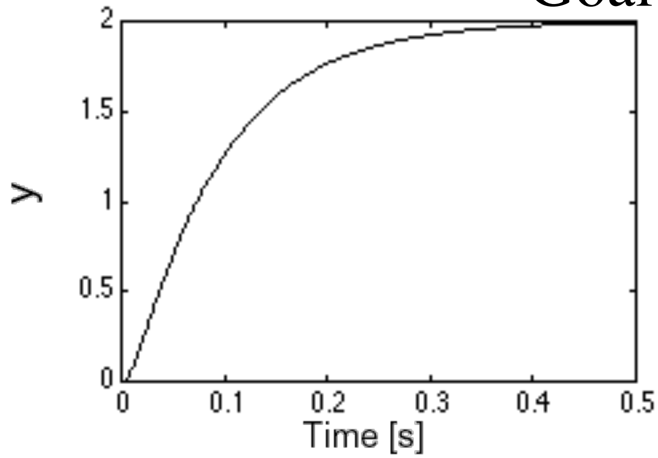
$$\dot{q}_d = \alpha\left(q_f - q_d\right) + \beta\left(q_d - q\right)$$

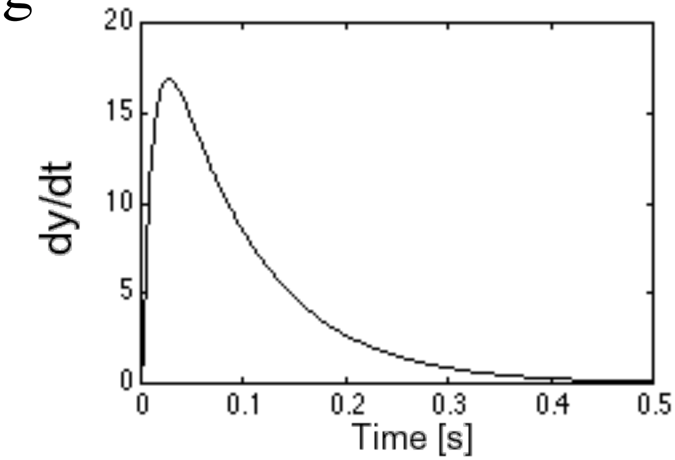
Pattern Generators for Trajectory Planning

- Disadvantages of Pattern Generators
  - Analysis of behavior is non trivial
  - Need to integrate the equation of motion of the pattern generator at sufficiently high frequency
  - Exact shape of desired trajectories that are generated by the pattern generator are not easy to predict if external coupling is added
  - Modeling of with pattern generators usually requires the manipulation of nonlinear dynamical equations, which is non trivial again

Shaping Attractor Landscapes

Goal=g



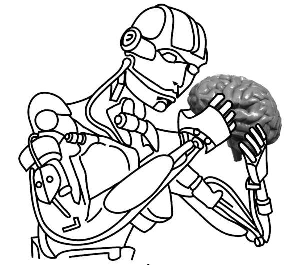


Second order dynamics:

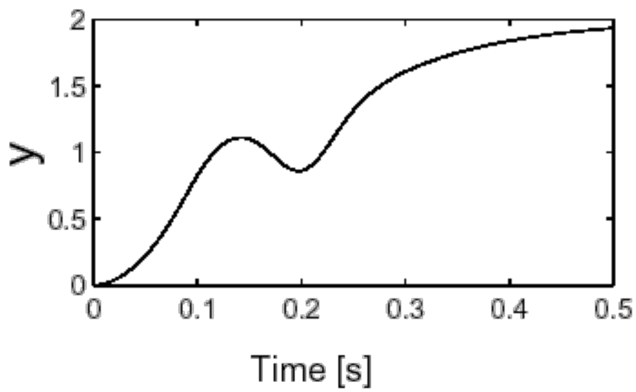
$$\dot{z} = \alpha_z\left(\beta_z(g - y) - z\right)$$

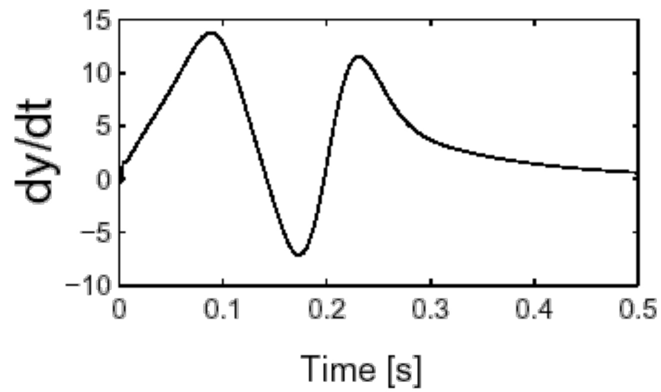
$$\dot{y} = z$$

Shaping Attractor Landscapes



Goal=g



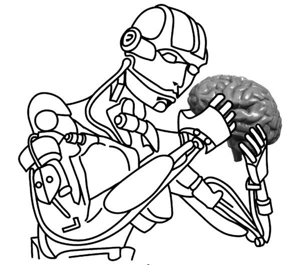


Can one create more complex dynamics by nonlinearly modifying the simple second order system?

$$\dot{z} = \alpha_z\left(\beta_z(g-y)-z\right)$$

$$\dot{y} = \left(f(?)+z\right)$$

Shaping Attractor Landscapes



- A globally stable learnable nonlinear point attractor:

Trajectory Plan Dynamics

$$\begin{cases} \dot{z} = \alpha_z\left(\beta_z\left(g - y\right) - z\right) \\ \dot{y} = \alpha_y\left(f\left(x, v\right) + z\right) \end{cases}$$

where

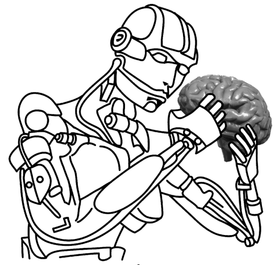
Canonical Dynamics

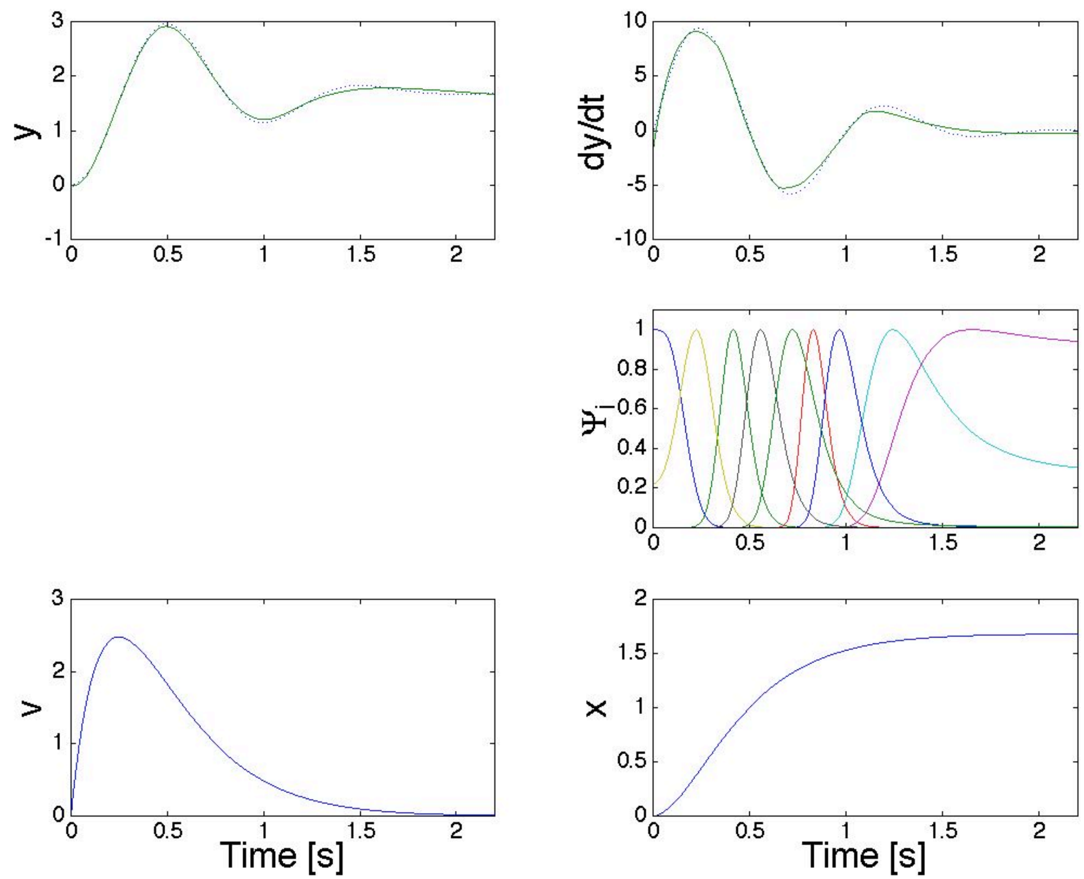
$$\begin{cases} \dot{v} = \alpha_v\left(\beta_v\left(g - x\right) - v\right) \\ \dot{x} = \alpha_x v \end{cases}$$

Local Linear Model Approx.

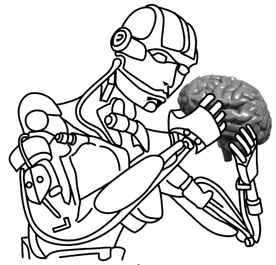
$$\begin{cases} f\left(x, v\right) = \dfrac{\sum_{i=1}^{k} w_i b_i v}{\sum_{i=1}^{k} w_i} \\ w_i = \exp\left(-\dfrac{1}{2} d_i \left(\bar{x} - c_i\right)^2\right) \text{ and } \bar{x} = \dfrac{x - x_0}{g - x_0} \end{cases}$$

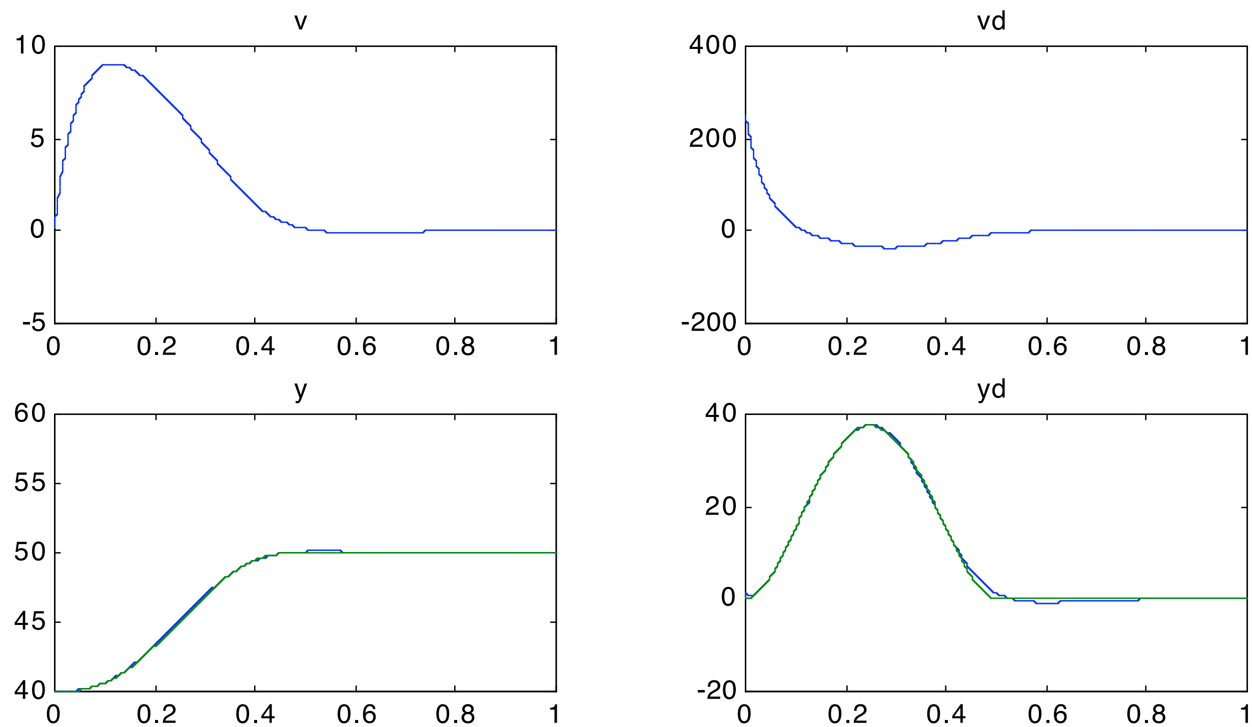
Example: A Trajectory with Movement Reversal





Example: A Minimum Jerk Trajectory





Learning The Attractor from Demonstration

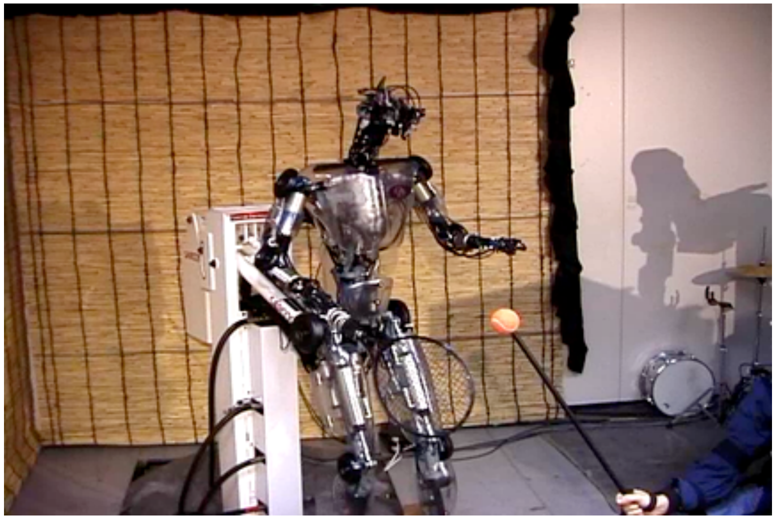
- Given a demonstrated trajectory $y(t)_{demo}$ and a goal g
    - Extract movement duration
    - Adjust time constants of canonical dynamics to movement duration
    - Use LWL to learn supervised problem

$$\dot{y}_{\text{target}} = \frac{\dot{y}_{demo}}{\alpha_y} - z = f(x,v)$$

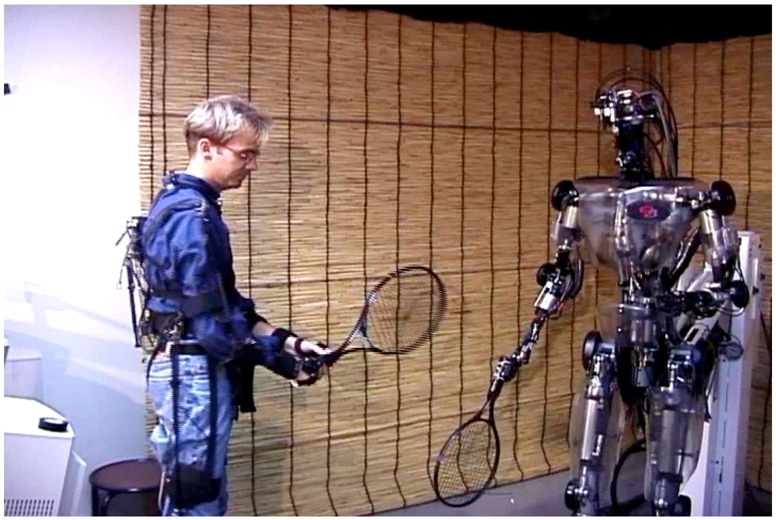
- Usually 1-5 learning epochs suffice to get good approximation

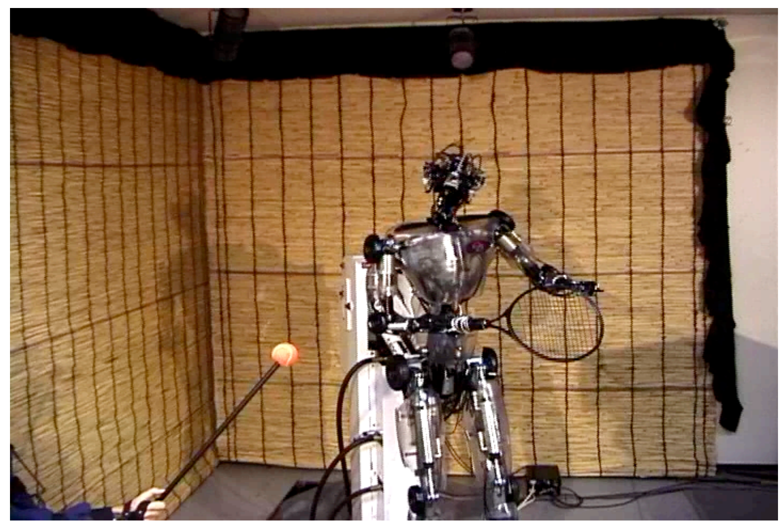
Imitation Learning of a Tennis Forehand

Note: All 30 joint space trajectories are fitted independently



Imitation Learning of a Tennis Backhand





Limit Cycle Dynamics for Rhythmic Movement

- A globally stable learnable limit cycle:

Trajectory Plan Dyanmics $\begin{cases} \dot{z} = \alpha_z\left(\beta_z\left(g - y_m\right) - z\right) \\ \dot{y} = \alpha_y\left(f\left(r,\varphi\right) + z\right) \end{cases}$

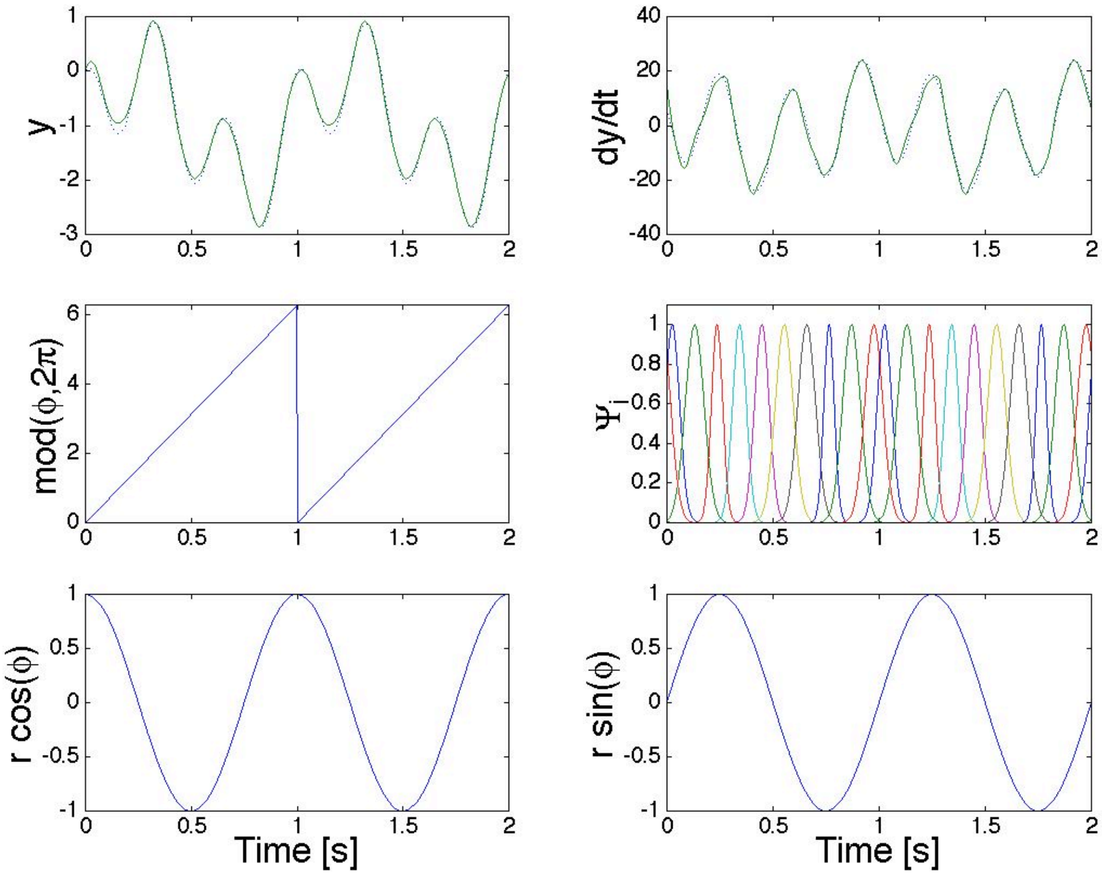
where

Canonical System $\begin{cases} \dot{r} = \alpha_r\left(A - r\right) \\ \dot{\varphi} = \omega \end{cases}$

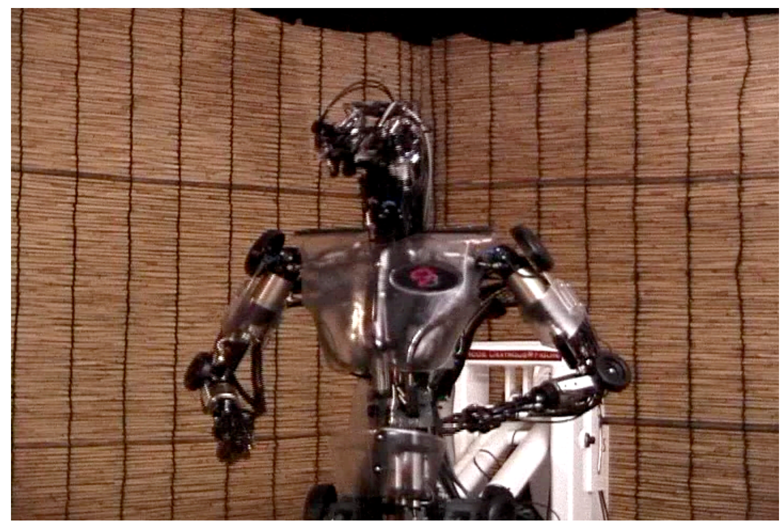
Local Linear Models using van Mises bases

$$\begin{cases} f\left(x,v\right) = \dfrac{\sum_{i=1}^{k} w_i \mathbf{b}_i^T \mathbf{x}}{\sum_{i=1}^{k} w_i} \quad \text{where} \quad \mathbf{x} = \begin{bmatrix} r\cos\varphi \\ r\sin\varphi \end{bmatrix} \\ w_i = \exp\left(d_i\left(\cos\left(\varphi - c_i\right) - 1\right)\right) \end{cases}$$

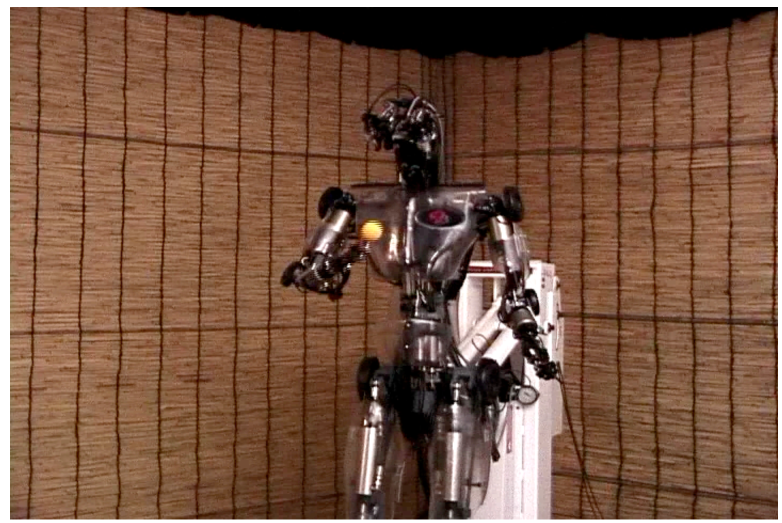
Example: A Complex Rhythmic Trajectory



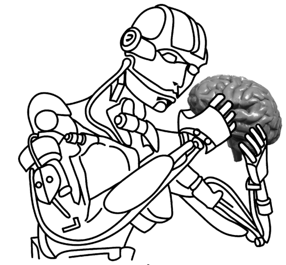
Imitation Learning of a Drumming Motion

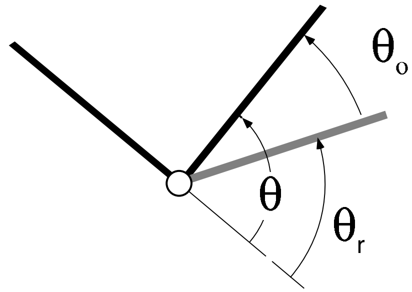


Imitation Learning of a Figure-8 Motion



Pattern Generators for Rhythmic and Discrete Movement





Discrete Movement

$$\Delta v_1 = \left[t_1 - p_{1,r}\right]^+ \qquad \Delta v_2 = \left[t_2 - p_{2,r}\right]^+ \qquad \theta_r = p_{1,r} = -p_{2,r}$$

$$\dot{v}_1 = a_v\left(-v_1 + \Delta v_1\right) \qquad \dot{v}_2 = a_v\left(-v_2 + \Delta v_2\right) \qquad \dot{\theta}_r = \dot{p}_{1,r} = -\dot{p}_{2,r}$$

$$\dot{x}_1 = -a_x x_1 + \left(v_1 - x_1\right)c_r + C_{1,r} \qquad \dot{x}_2 = -a_x x_2 + \left(v_1 - x_2\right)c_r + C_{2,r}$$

$$\dot{y}_1 = -a_y y_1 + \left(x_1 - y_1\right)c_r \qquad \dot{y}_2 = -a_y y_2 + \left(x_1 - y_2\right)c_r$$

$$\dot{r}_1 = a_r\left(-r_1 + \left(1 - r_1\right) b\, v_1\right) \qquad \dot{r}_2 = a_r\left(-r_2 + \left(1 - r_2\right) b\, v_2\right)$$

$$\dot{z}_1 = -a_z z_1 + \left(y_1 - z_1\right)\left(1 - r_1\right)c_r \qquad \dot{z}_2 = -a_z z_2 + \left(y_2 - z_2\right)\left(1 - r_2\right)c_r$$

$$\dot{p}_{1,r} = a_p c_r\left(z_1 - z_2\right) \qquad \dot{p}_{2,r} = a_p c_r\left(z_2 - z_1\right)$$

Rhythmic Movement

$$\Delta\omega_1 = \left[A - \left(p_1 - p_{1,r}\right)\right]^+$$

$$\dot{\xi}_1 = a_\xi\left(-\xi_1 + \Delta\omega_1\right)$$

$$\dot{\psi}_1 = -a_\psi \psi_1 + \left(\xi_1 - \psi_1 - b\zeta_1 - w\left[\psi_2\right]^+ + C_{1,o}\right)c_o$$

$$\dot{\zeta}_1 = \frac{1}{5}\left(-a_\zeta \zeta_1 + \left(\left[\psi_1\right]^+ - \zeta_1\right)c_o\right)$$

$$\dot{p}_1 = c_o\left(\left[\psi_1\right]^+ - \left[\psi_2\right]^+\right)$$

$$\Delta\omega_2 = \left[A - \left(p_2 - p_{2,r}\right)\right]^+$$

$$\dot{\xi}_2 = a_\xi\left(-\xi_2 + \Delta\omega_2\right)$$

$$\dot{\psi}_2 = -a_\psi \psi_2 + \left(\xi_2 - \psi_2 - b\zeta_2 - w\left[\psi_1\right]^+ + C_{2,o}\right)c_o$$

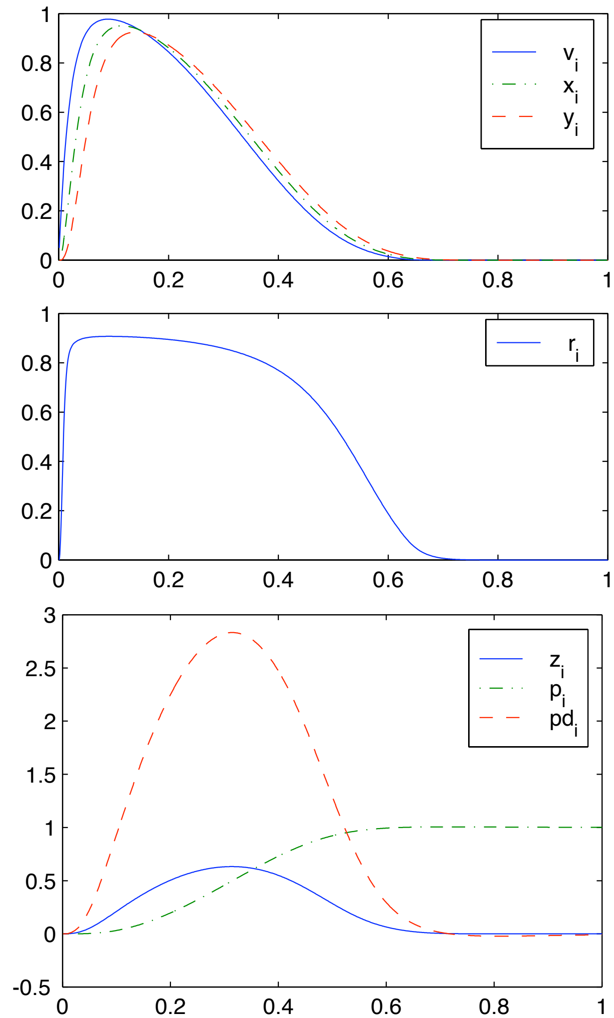
$$\dot{\zeta}_2 = \frac{1}{5}\left(-a_\zeta \zeta_2 + \left(\left[\psi_2\right]^+ - \zeta_2\right)c_o\right)$$

$$\dot{p}_2 = c_o\left(\left[\psi_2\right]^+ - \left[\psi_1\right]^+\right)$$

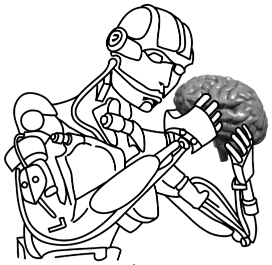
$$\theta = p_1 = -p_2$$

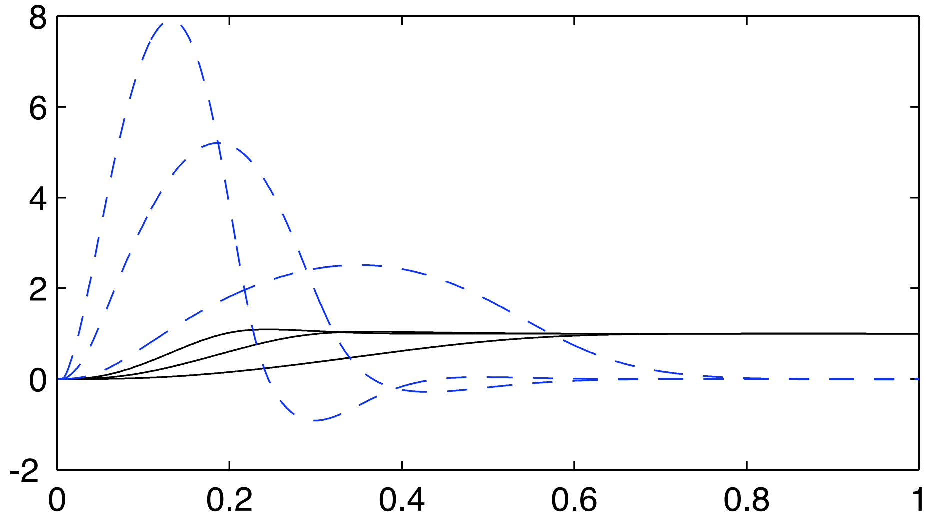
$$\dot{\theta} = \dot{p}_1 = -\dot{p}_2$$

Example from the Discrete Pattern Generator



Discrete Movements at Different Speeds





Example from the Rhythmic Pattern Generator