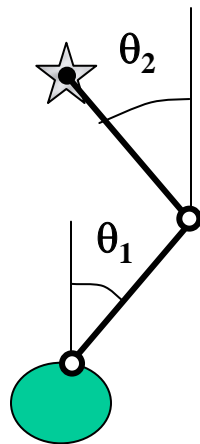
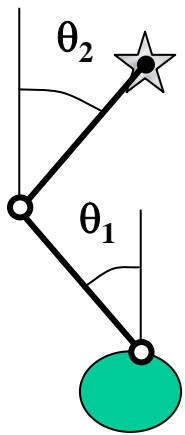


Motor Control

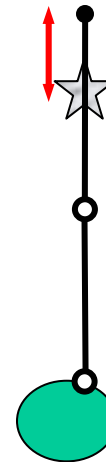
Beyond babbling

- Three problems with motor babbling:
 - Random exploration is slow
 - Error-based learning algorithms are faster but error signals are available in sensory coordinates only
 - Real arms have too many degrees of freedom

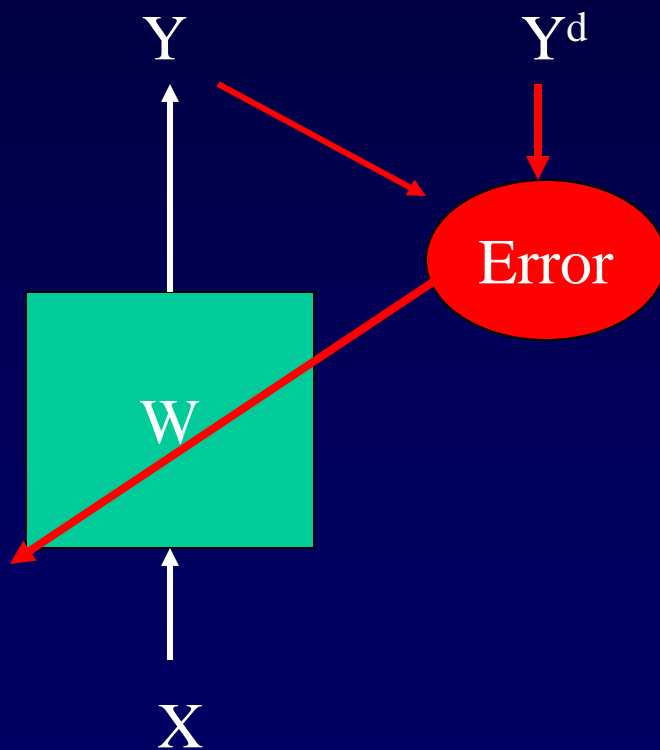
Degrees of freedom



Average
position



General Learning Principles

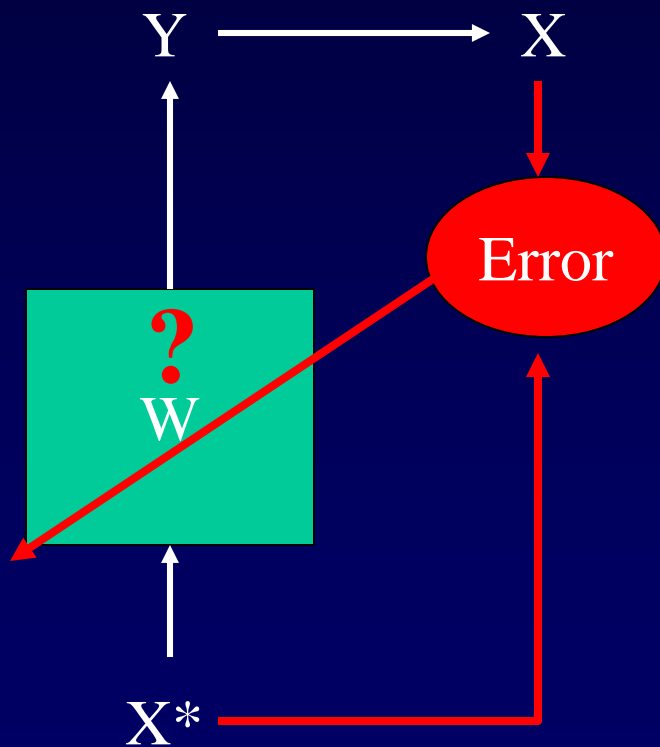


$$E = f(Y^d, Y) = \frac{1}{2} \|Y^d - Y\|^2$$
$$= \frac{1}{2} \sum_{i=1}^N (Y_i^d - Y_i)^2$$

$$\Delta W = -\alpha \frac{\partial E}{\partial W} = -\alpha \frac{\partial E}{\partial Y} \frac{\partial Y}{\partial W}$$
$$= \alpha (Y^d - Y) \frac{\partial Y}{\partial W}$$

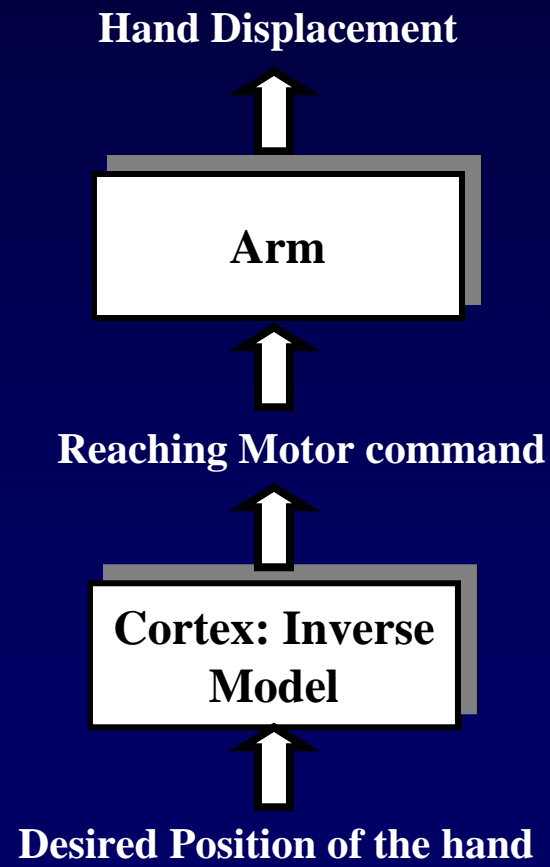
This works if we have equations for this term...

General Learning Principles

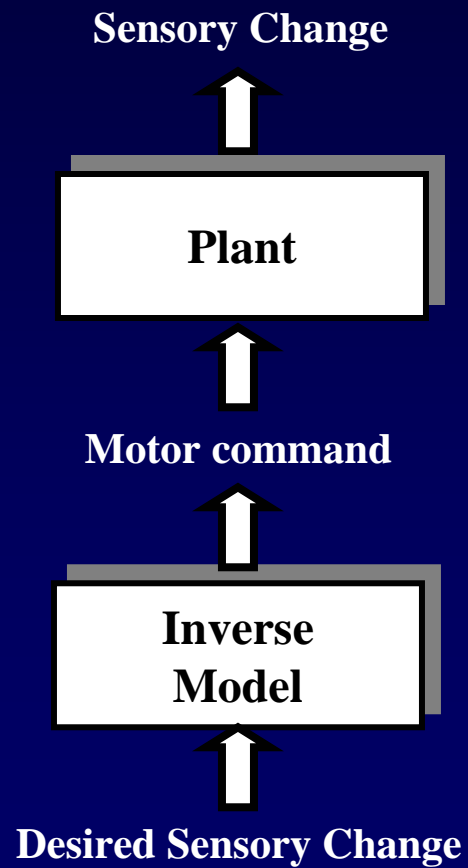


$$\Delta W = -\alpha \frac{\partial E}{\partial W} ?$$

Forward Modeling



Forward Modeling

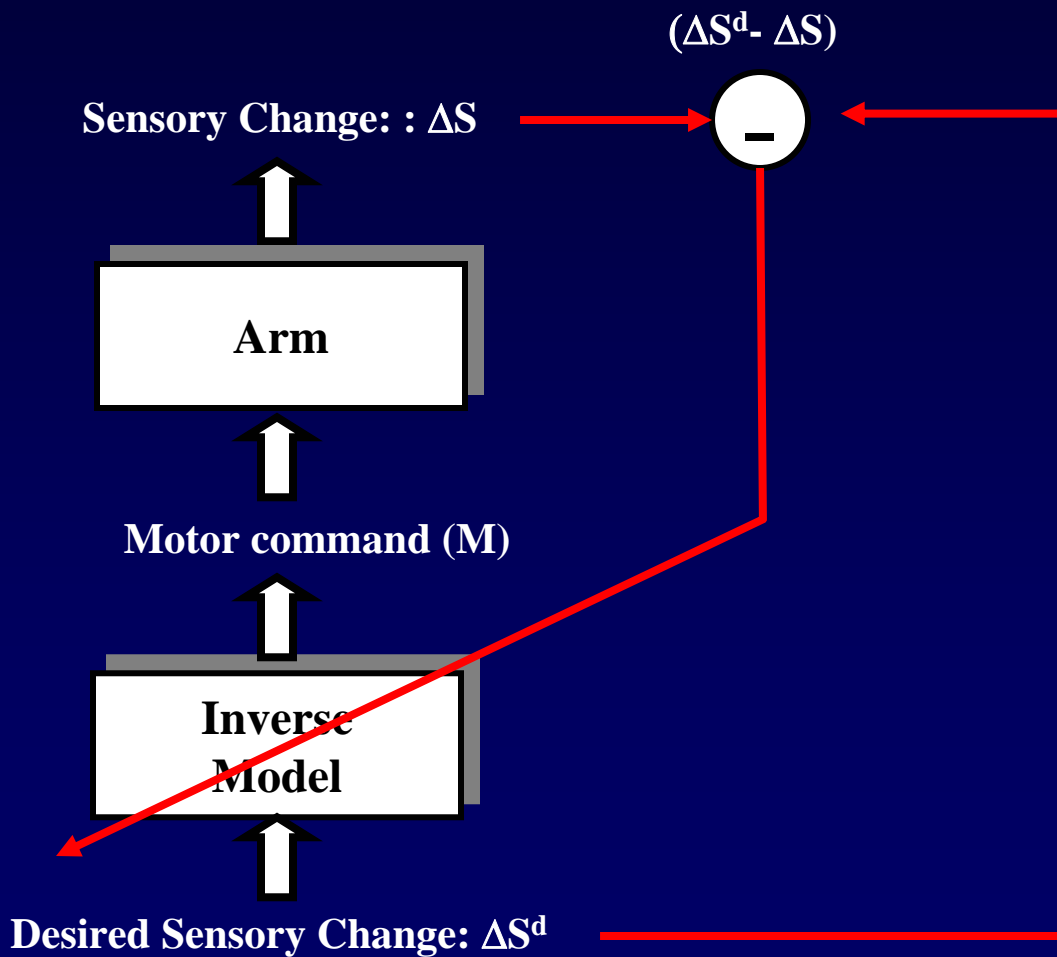


Forward Modeling

Learning starts with desired change, not a spontaneous movement.

$$E = \frac{1}{2} (\Delta S^d - \Delta S)^2$$

$$\delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$$

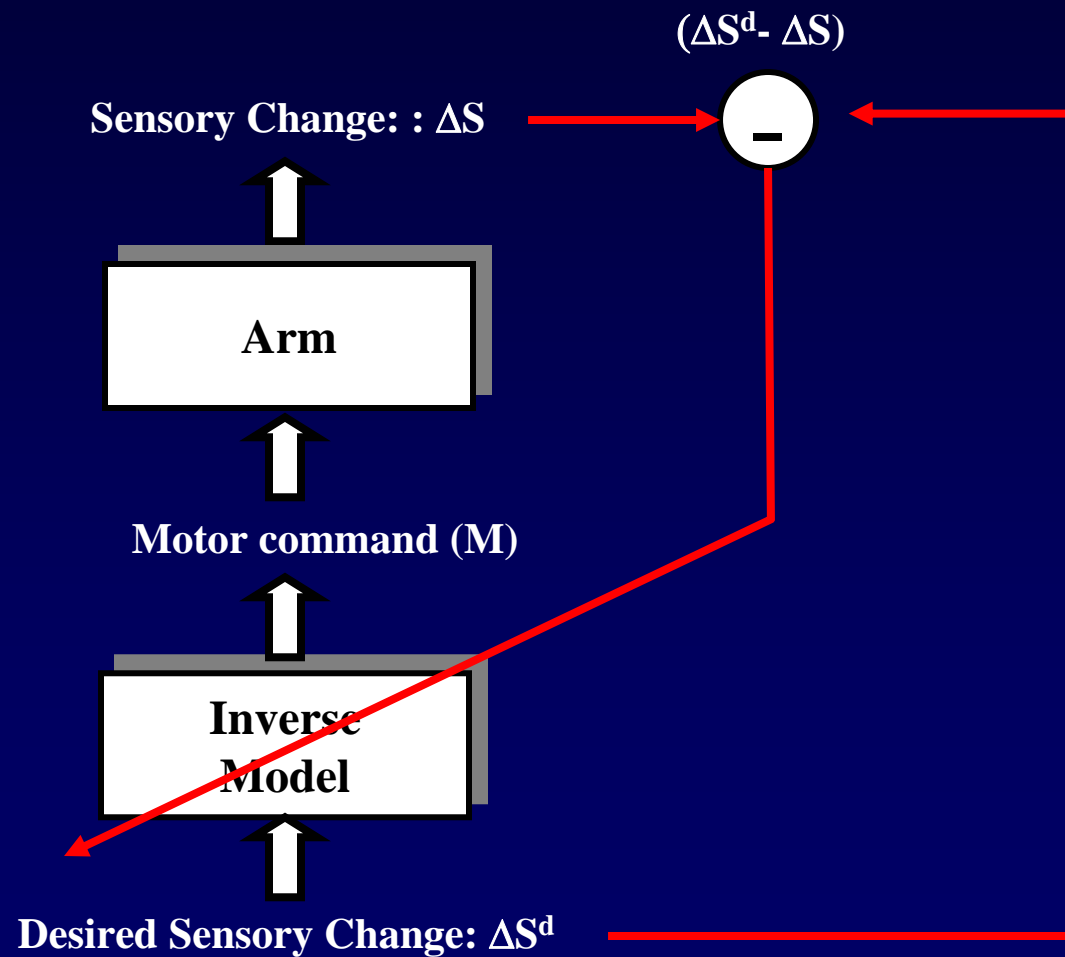


Forward Modeling

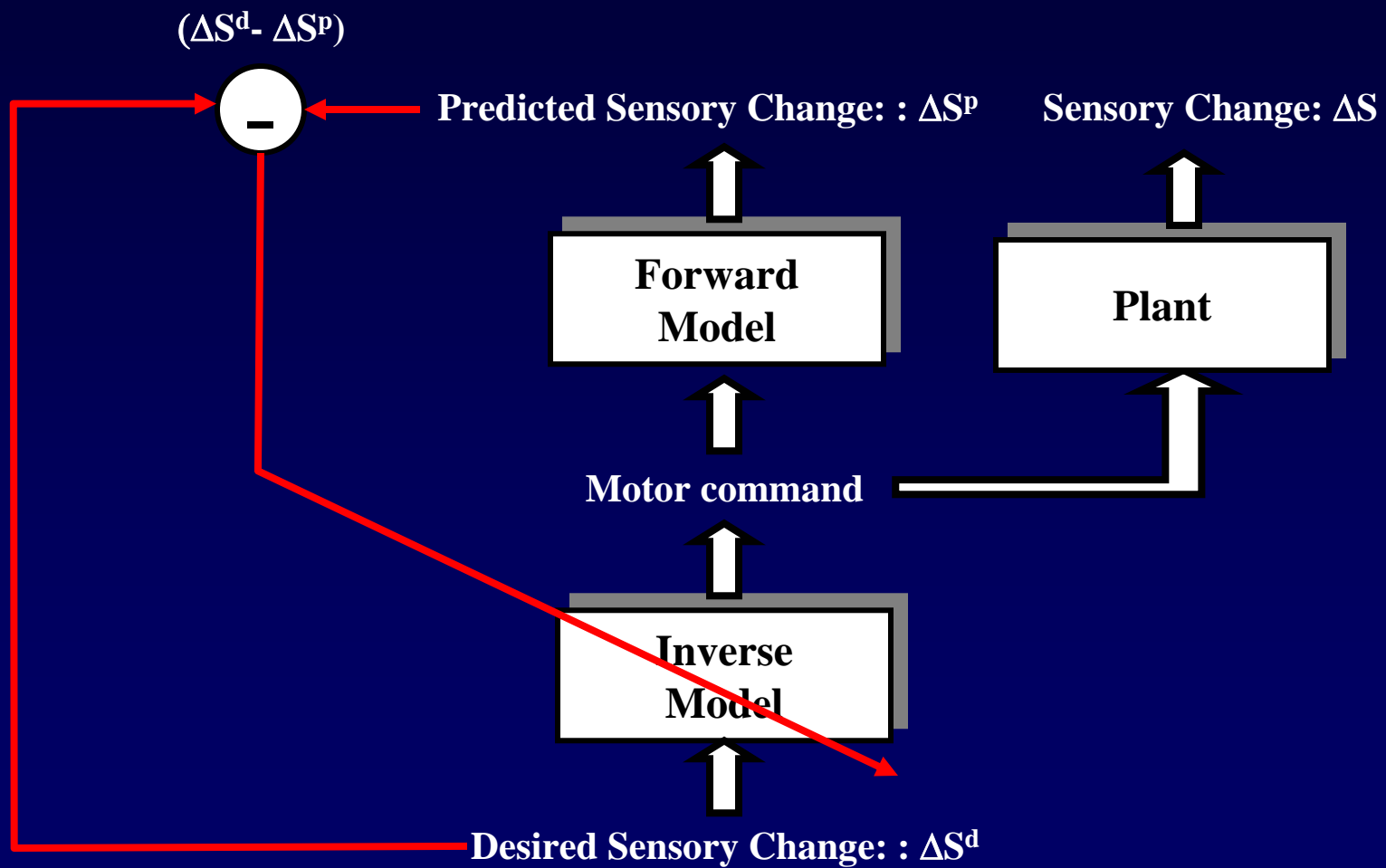
$$\begin{aligned}\delta w_{ij} &= -\alpha \frac{\partial E}{\partial w_{ij}} \\ &= -\alpha \frac{\partial E}{\partial M} \frac{\partial M}{\partial w_{ij}}\end{aligned}$$

Major problem:
How do we compute

$$\frac{\partial E}{\partial M} ?$$



Forward Modeling



Forward Modeling

We can't compute

$$\delta w_{ij} = -\alpha \frac{\partial E}{\partial M} \frac{\partial M}{\partial w_{ij}}$$

but we can use instead,

$$\delta w_{ij} = -\alpha \frac{\partial E^P}{\partial M} \frac{\partial M}{\partial w_{ij}}$$
$$E^P = \frac{1}{2} (\Delta S^d - \Delta S^P)^2$$

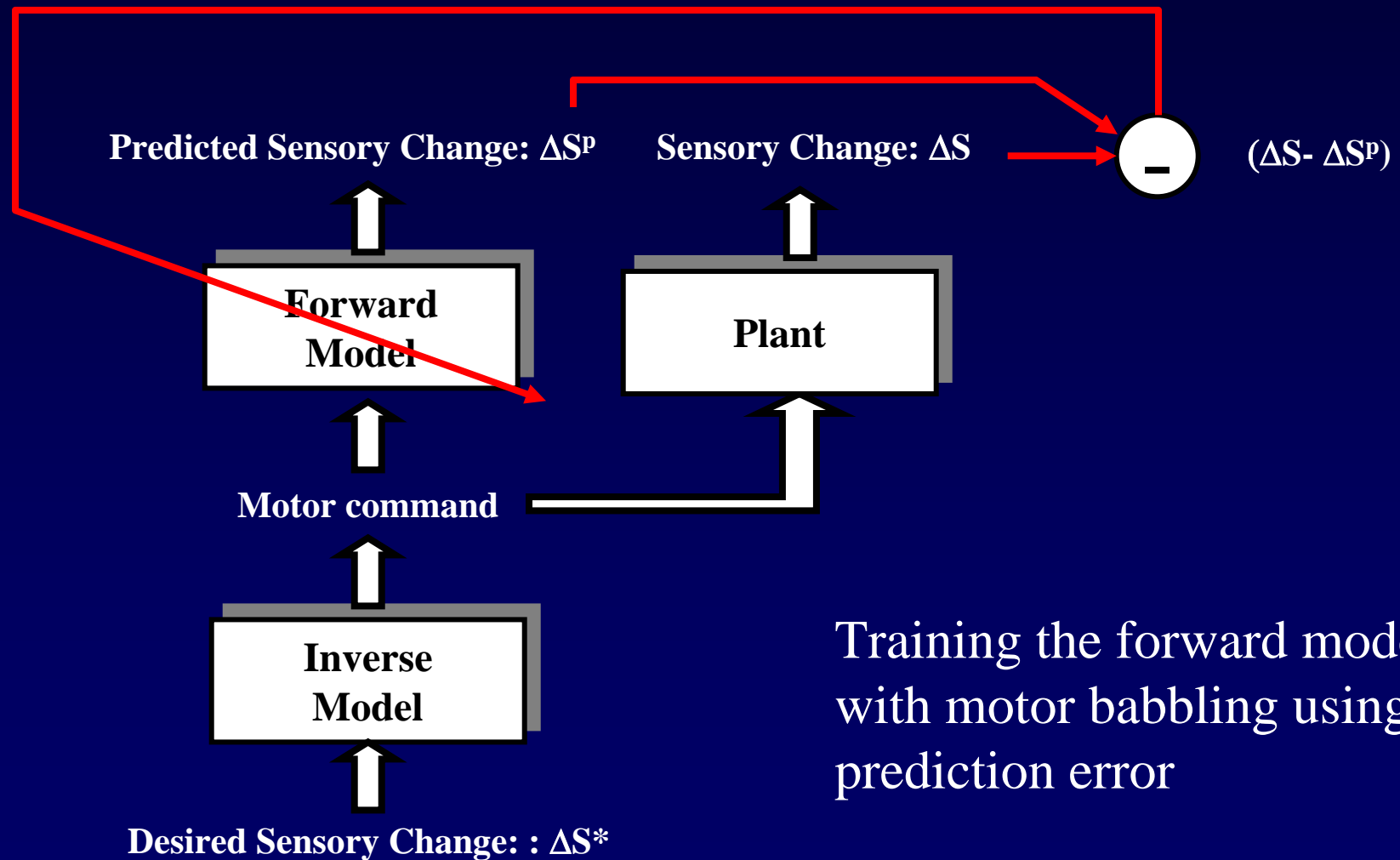
Forward Modeling

This works as long as:

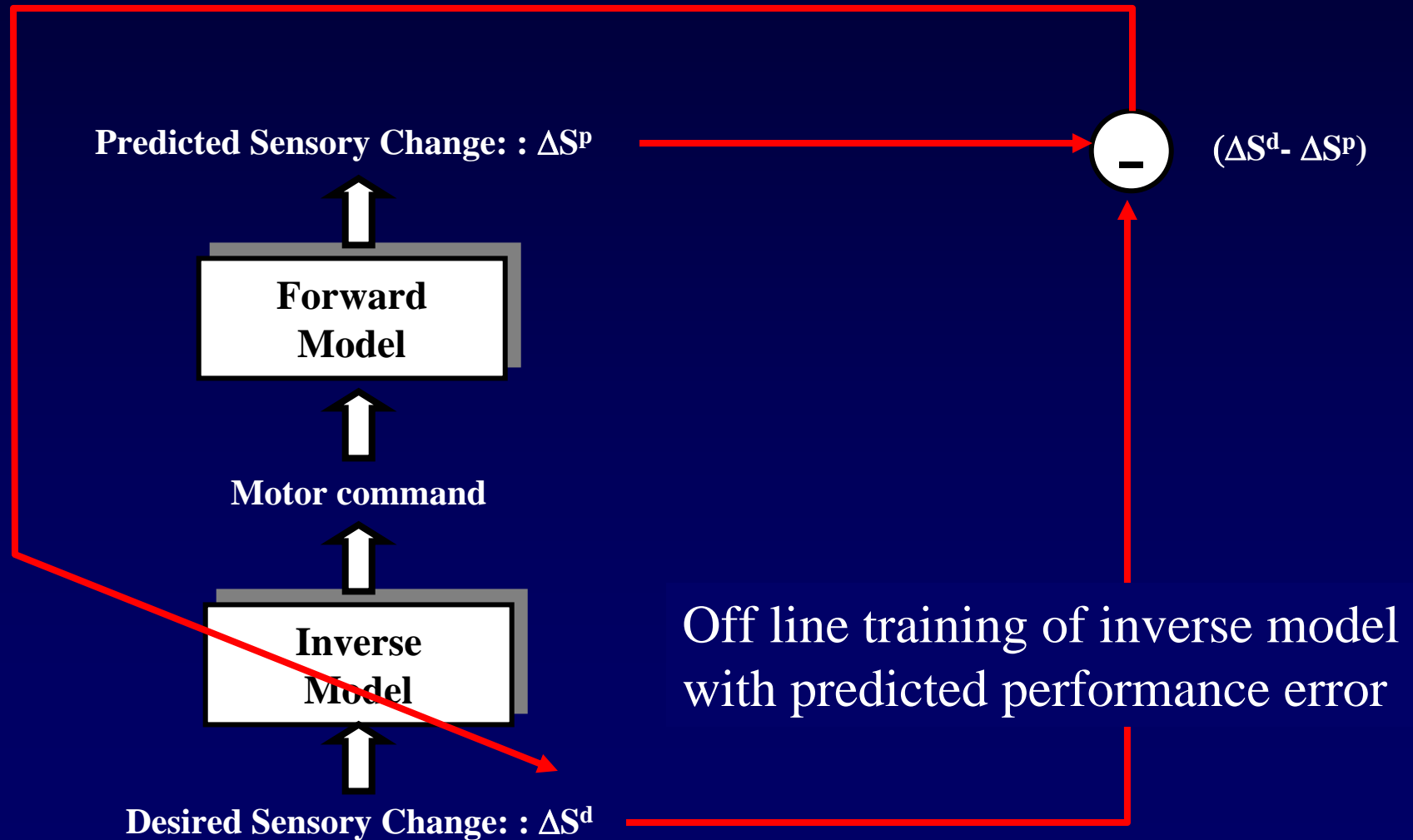
$$\frac{\partial E^P}{\partial M} \frac{\partial E}{\partial M} > 0$$

This means that the forward model only needs to be approximately correct

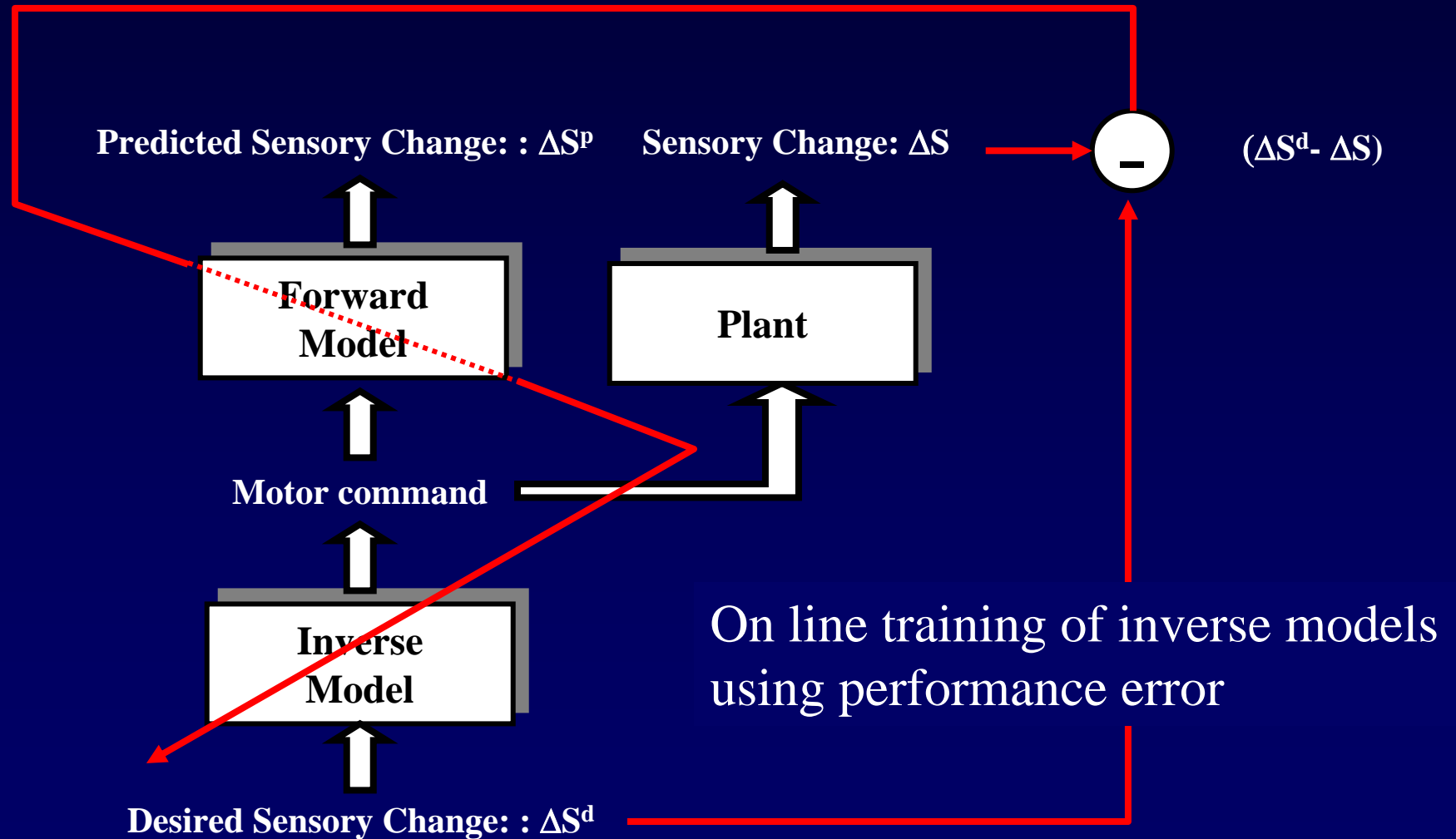
Forward Modeling



Forward Modeling



Forward Modeling



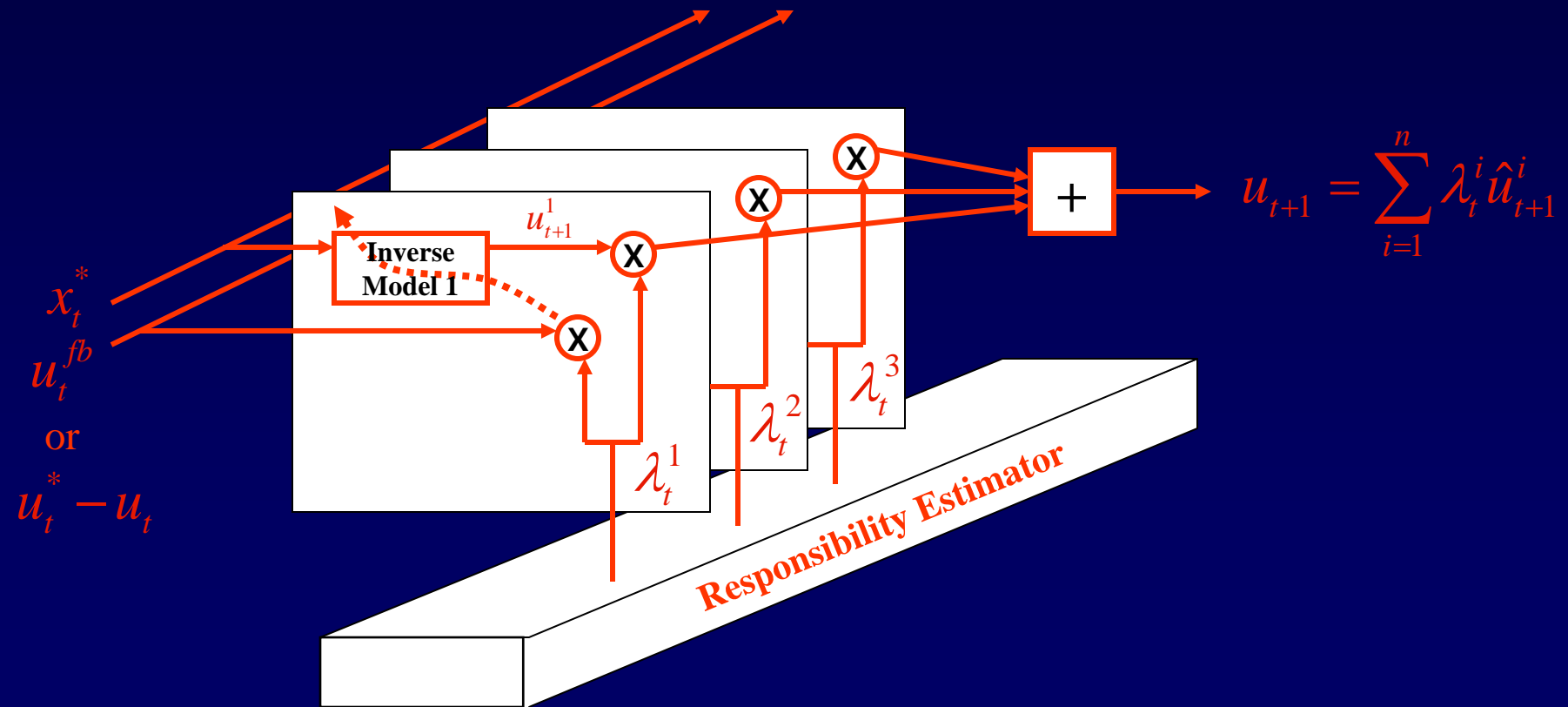
Forward Modeling

- Forward models can be used to predict the consequences of motor commands
- The prediction can be used to drive a linear inverse model (e.g. Jacobian) since they are not subject to long delays
- The prediction can be subtracted from current sensory input to compute the prediction error.
- Errors can be used to improve prediction.

Modular Approach

- Sensorimotor transformations are context dependent, e.g., the force required to move an object depends on the mass of the objects.
- A central controller for all contexts might not be feasible and would take too many resources
- Alternative: use a family of controllers and mix them smoothly across contexts.

Modular Approach



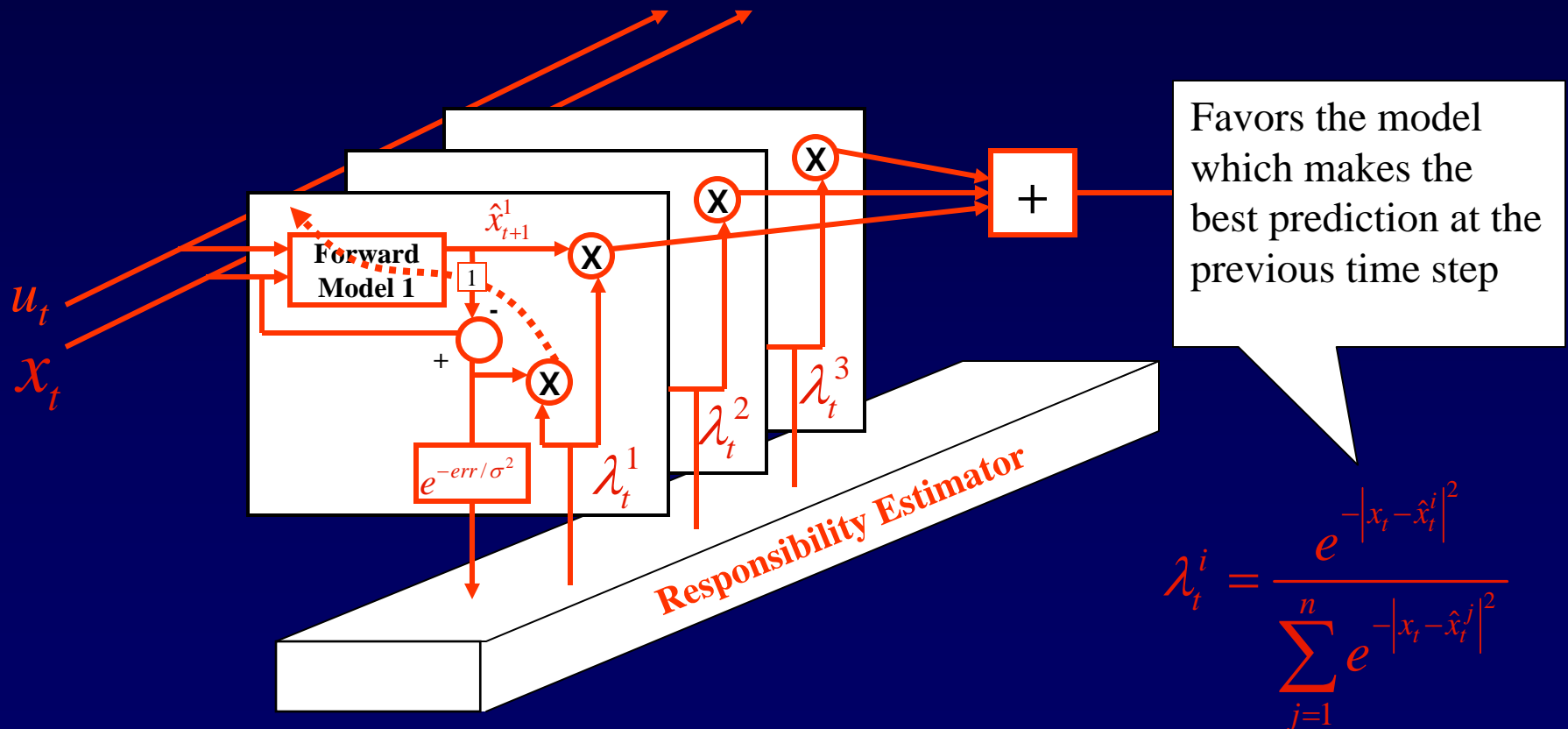
Modular Approach

- The weights of the inverse model are adjusted according to:

$$\Delta \alpha_t^i = \varepsilon \lambda_t^i \frac{du_t^i}{d\alpha_t^i} (u_t^* - u_t^i)$$

Modular Approach

- To compute the responsibility, use the forward models



Modular Approach

- The weights of the forward model are adjusted according to:

$$\Delta w_t^i = \varepsilon \lambda_t^i \frac{d\hat{x}_t^i}{dw_t^i} (x_t - \hat{x}_t^i)$$

Modular Approach

- We can also add a responsibility predictor.

$$\hat{\lambda}_t^i = \eta(\gamma_t^i, y_t)$$

$$\Delta \gamma_t^i = \varepsilon \frac{d\hat{\lambda}_t^i}{d\gamma_t^i} (\lambda_t^i - \hat{\lambda}_t^i)$$

where y_t are the sensory cues used for the predictions.

Modular Approach

- The overall responsibility is computed according to:

$$\lambda_t^i = \frac{\hat{\lambda}_t^i e^{-|x_t - \hat{x}_t^i|^2}}{\sum_{j=1}^n \hat{\lambda}_t^j e^{-|x_t - \hat{x}_t^j|^2}}$$

$e^{-|x_t - \hat{x}_t^i|^2}$ plays the role of a likelihood function,
while $\hat{\lambda}_t^i$ is the prior.

Controlling a trajectory

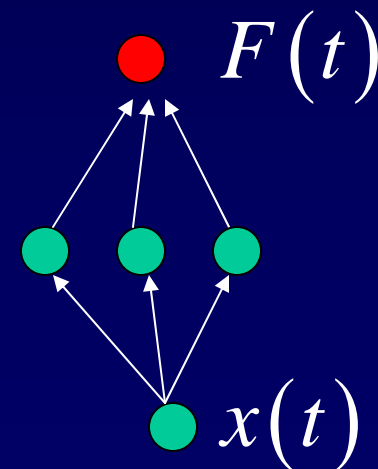
- The inverse problem (*inverse dynamics*)
- Ex: Controlling the trajectory of a spaceship

Forward Model

$$x(t) = \int_0^t \left(\int_0^t a(\tau) d\tau \right) d\tau$$

Inverse Model

$$F(t) = ma(t) = m\ddot{x}(t)$$



Controlling a trajectory

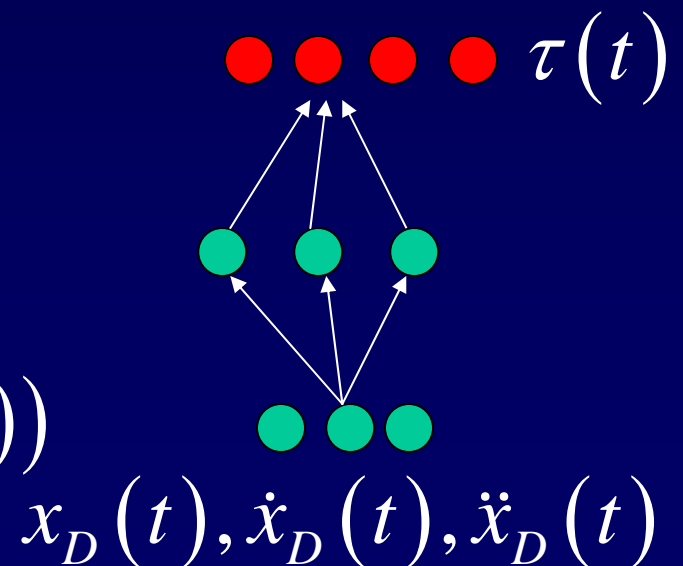
- The inverse problem (*inverse dynamics*)
- Ex: a multi-joint arm ($\tau(t)$: joint torques)

Forward Model

$$(x(t), \dot{x}(t), \ddot{x}(t)) = H(\tau(t))$$

Inverse Model

$$\tau(t) = H^{-1}(x_D(t), \dot{x}_D(t), \ddot{x}_D(t))$$



Controlling a trajectory

- The inverse problem (*inverse dynamics*)
 - Feedback models
 - Feedforward models
 - Equilibrium point models

Controlling a trajectory

- The inverse problem: feedback models
- Generate force according to the difference between desired position and actual position (easy to compute in linear systems).
- Unstable because of sensory delays
- Works better if the “actual position” is internally estimated by a forward model
- Adapts on the fly to change of context
- Popular model of the oculomotor system

Controlling a trajectory

- The inverse problem: feedforward models
- Estimating torque and its differentials from a desired trajectory is a nonlinear mapping
- Use a basis function network to implement the mapping
- Subject to the curse of dimensionality
- Unable to adapt to change of context (unless you add context units...)

Controlling a trajectory

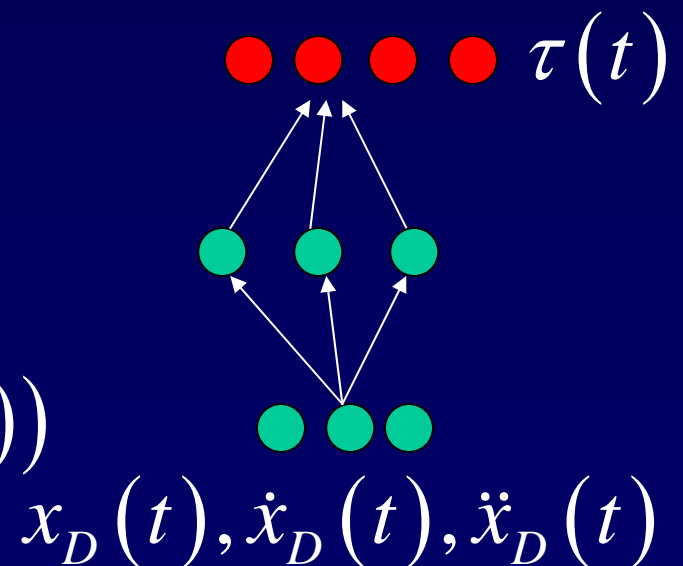
- The inverse problem (*inverse dynamics*)
- Ex: a multi-joint arm ($\tau(t)$: joint torques)

Forward Model

$$(x(t), \dot{x}(t), \ddot{x}(t)) = H(\tau(t))$$

Inverse Model

$$\tau(t) = H^{-1}(x_D(t), \dot{x}_D(t), \ddot{x}_D(t))$$



Controlling a trajectory

- Equilibrium models
- Basic idea: the motor system only specifies the muscle length that maintain the arm at the desired location (in other words, it only specifies the joint coordinates).

Controlling a trajectory

- Equilibrium models
- Monkeys can reach accurately without proprioceptive or visual feedback
- If the arm is moved to the end point right at the onset of a movement, it goes back to the starting point and resume its trajectory. This implies that trajectory are specified by moving the equilibrium point

Controlling a trajectory

- Equilibrium models
- How do you control trajectories? The motor system may specify the trajectory of the equilibrium point (virtual trajectory). Unless muscles are very stiff, the actual trajectory will end up being very different. This makes it difficult to control trajectories very precisely.

Controlling a trajectory

- There is an infinite number of trajectories between two points. How do we choose one? Do you have to specify one?
 - Minimum Jerk (Flash & Hogan)
 - Maximum accuracy (Harris & Wolpert)
 - Optimal control (Todorov & Jordan)

Controlling a trajectory

- Minimum Jerk

$$C = \int_{t_0}^{t_f} \left(\frac{d^3\theta}{dt^3} \right)^2 dt$$

- Goal: find $\theta(t)$ minimizing C . The solution can be found using calculus of variations.

Controlling a trajectory

- Minimum Jerk
- It's unclear how Jerk is computed in the brain.
- No principled explanation for why the brain would minimize such a quantity.

Controlling a trajectory

- Maximum accuracy
- Hypothesis: trajectory are optimized to maximize accuracy
- Assumption: motor commands are corrupted by noise with standard deviation proportional to mean (this is different from Poisson noise!!)
- Question: for a fixed duration and amplitude of a movement, what's the optimal control signal?

Controlling a trajectory

- Maximum accuracy
- Large control signals lead to fast but inaccurate movements
- Small control signals lead to accurate but slow movements.
- Goal: select the control signal that leads to maximum accuracy for a given duration.

Controlling a trajectory

- Maximum accuracy

$$x_{t+1} = \mathbf{A}x_t + \mathbf{B}(u_t + w_t)$$

$$x_{t+1} = \mathbf{A}^t x_0 + \sum_{i=1}^{t-1} \mathbf{A}^{t-1-i} \mathbf{B}(u_i + w_i)$$

- w_t : white noise with mean zero and variance ku_t^2

$$E[x_t] = \mathbf{A}^t x_0 + \sum_{i=1}^{t-1} \mathbf{A}^{t-1-i} \mathbf{B}u_i$$

$$Cov[x_t] = k \sum_{i=0}^{t-1} (\mathbf{A}^{t-1-i} \mathbf{B})(\mathbf{A}^{t-1-i} \mathbf{B})^T u_i^2$$

Controlling a trajectory

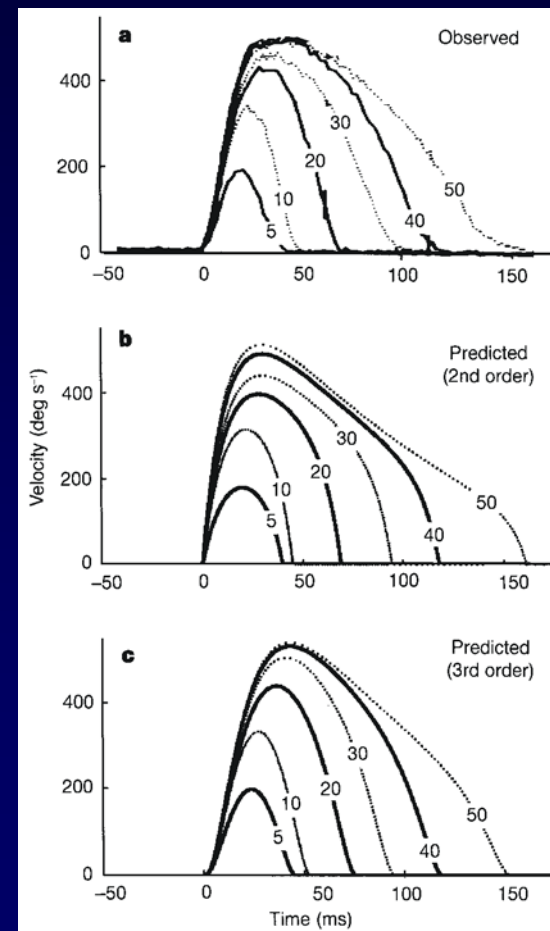
- Maximum accuracy
- Goal: minimize $\text{cov}[x_t]$ under the constraint that $E[x_t]$ is equal the desired location for several time steps after the end of the movement. Quadratic problem.

$$E[x_t] = \mathbf{A}^t x_0 + \sum_{i=1}^{t-1} \mathbf{A}^{t-1-i} \mathbf{B} u_i$$

$$\text{Cov}[x_t] = k \sum_{i=0}^{t-1} (\mathbf{A}^{t-1-i} \mathbf{B}) (\mathbf{A}^{t-1-i} \mathbf{B})^T u_i^2$$

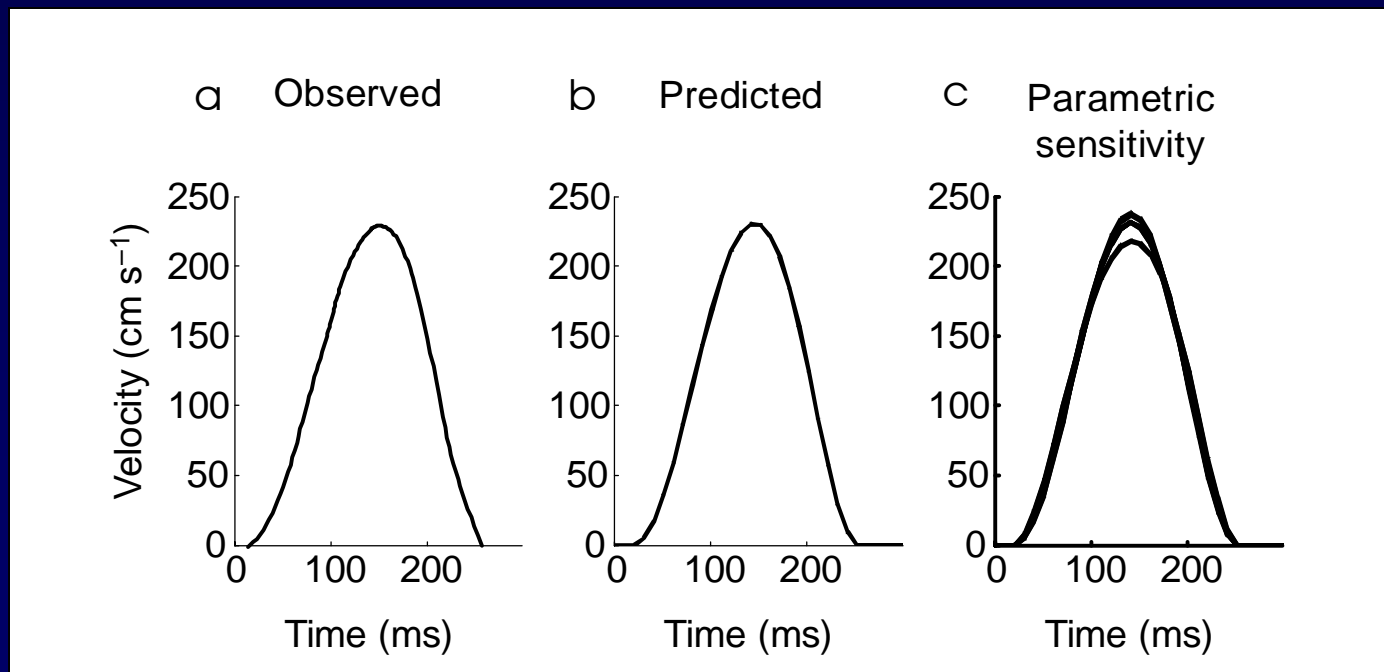
Controlling a trajectory

- Maximum accuracy
- Eye Movements



Controlling a trajectory

- Maximum accuracy
- Arm Movements



Controlling a trajectory

- Optimal motor control (Todorov-Jordan)
- Experts show a lot of variance in their movements but high accuracy on end points
- Indeed, there are directions in motor space that induce no variance in end points, because of the large number of degrees of freedom

Controlling a trajectory

- Optimal motor control (Todorov-Jordan)
- Choose trajectory with maximum accuracy and minimum effort

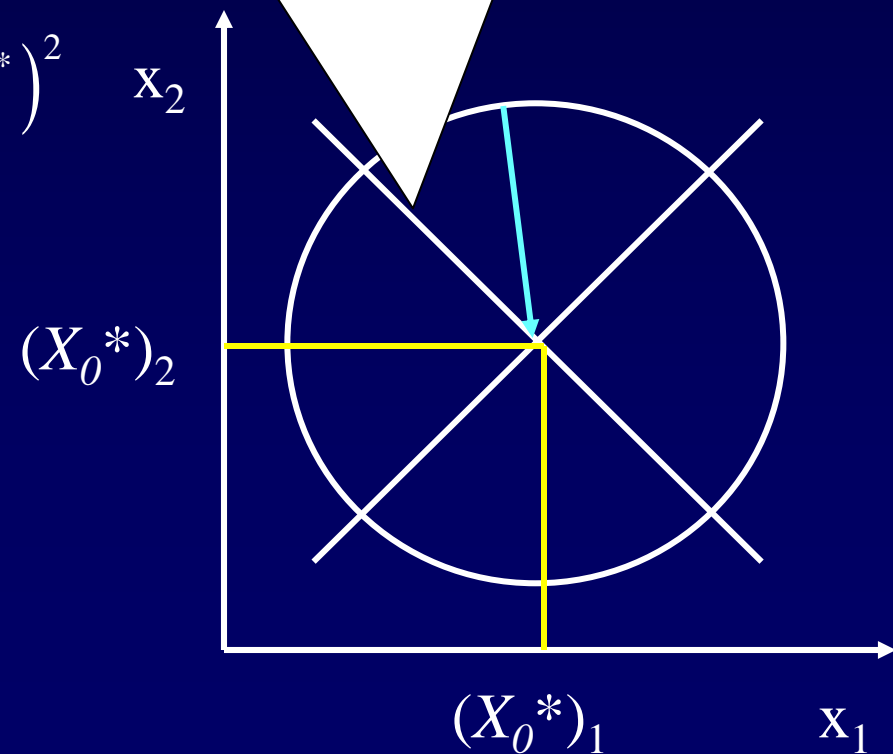
Controlling a trajectory

$$x_i^{\text{final}} = ax_i + u_i + u_i \sigma \varepsilon_i, \quad i \in \{1, 2\}$$

$$\mathbf{u} = \arg \min E_\varepsilon \left(x_1^{\text{final}} + x_2^{\text{final}} - X_0^* \right)^2$$

1st solution: bring \mathbf{X} to a value \mathbf{X}^* such that $x_1^* = (X_0^*)_1$, $x_2^* = (X_0^*)_2$

Line where the constraint $x_1 + x_2 = X_0^*$ is verified

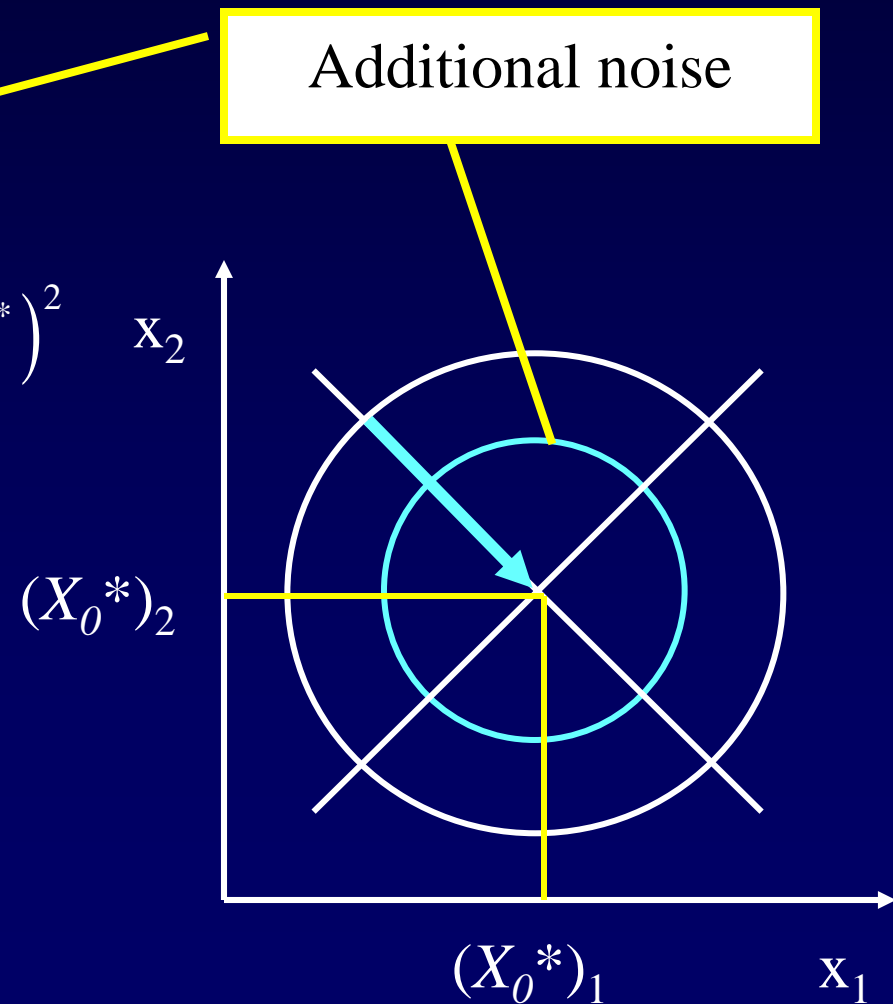


Controlling a trajectory

$$x_i^{\text{final}} = ax_i + u_i + u_i \sigma \varepsilon_i, \quad i \in \{1, 2\}$$

$$\mathbf{u} = \arg \min E_\varepsilon \left(x_1^{\text{final}} + x_2^{\text{final}} - X_0^* \right)^2$$

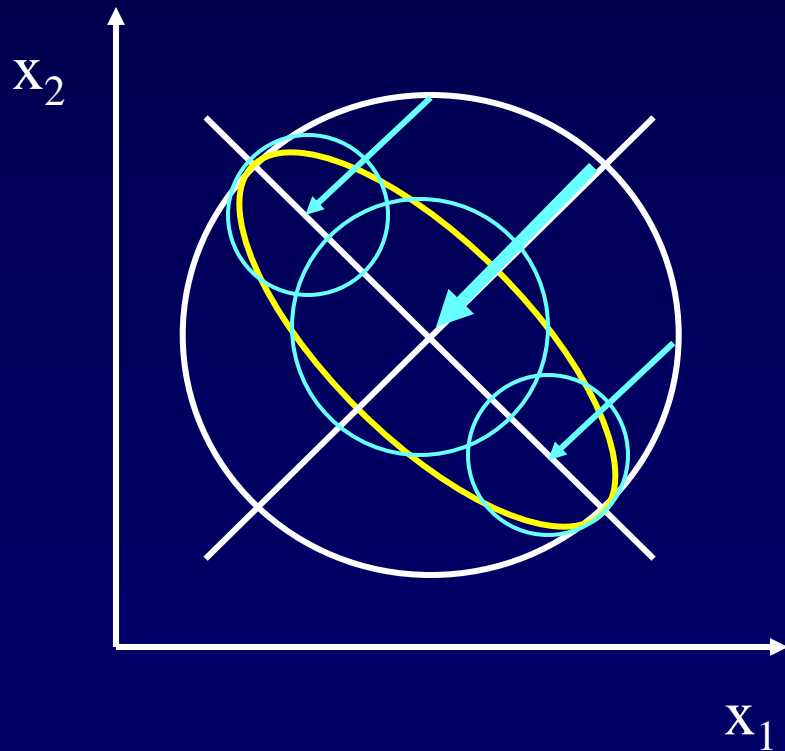
1st solution: bring \mathbf{X} to a value \mathbf{X}^* such that $x_1^* = (X_0^*)_1$, $x_2^* = (X_0^*)_2$



Controlling a trajectory

$$x_i^{\text{final}} = ax_i + u_i + u_i \sigma \varepsilon_i, \quad i \in \{1, 2\}$$

$$\mathbf{u} = \arg \min E_\varepsilon \left(x_1^{\text{final}} + x_2^{\text{final}} - X_0^* \right)^2 + r \left(u_1^2 + u_2^2 \right)$$

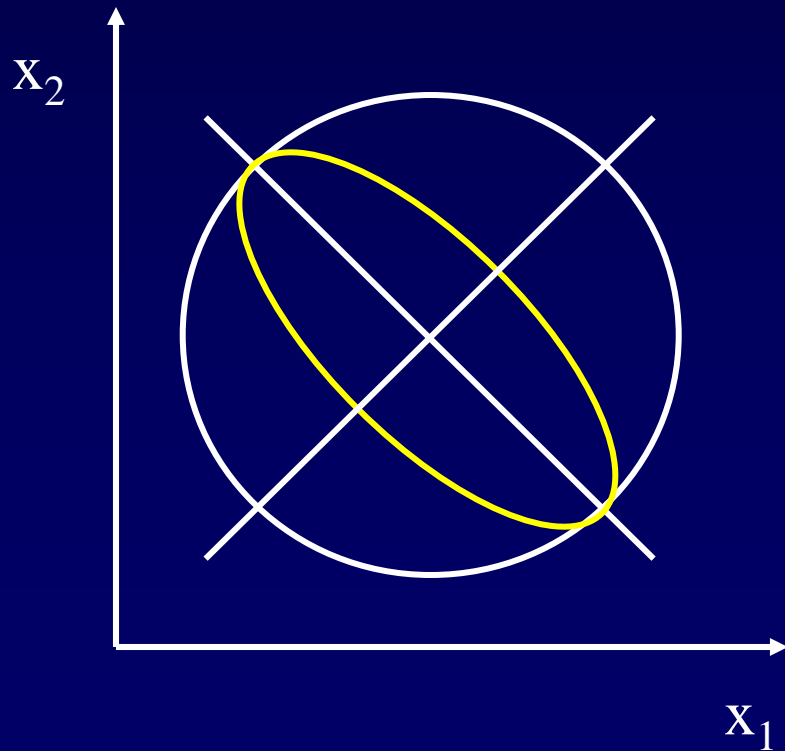


2nd solution: To minimize effort, go to the closest point such that $x_1^* + x_2^* = X_0^*$

Controlling a trajectory

$$x_i^{\text{final}} = ax_i + u_i + u_i \sigma \varepsilon_i, \quad i \in \{1, 2\}$$

$$\mathbf{u} = \arg \min E_\varepsilon \left(x_1^{\text{final}} + x_2^{\text{final}} - X_0^* \right)^2 + r \left(u_1^2 + u_2^2 \right)$$

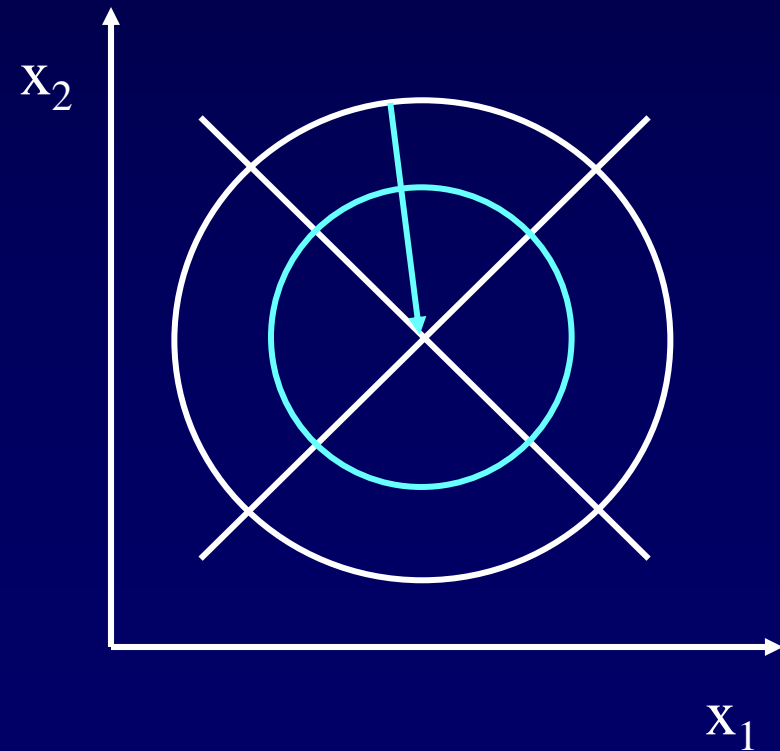
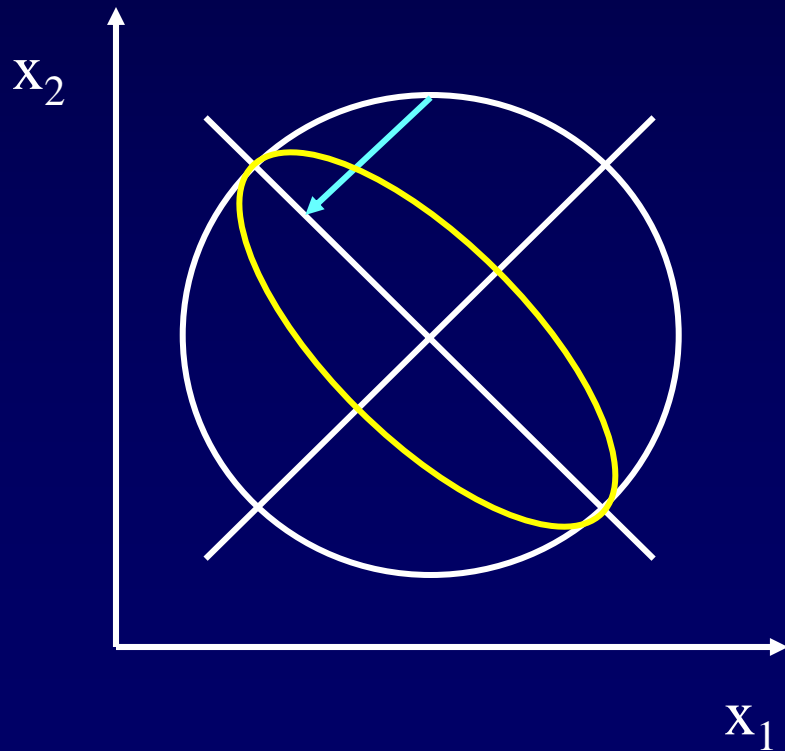


2nd solution: To minimize effort, go to the closest point such that $x_1^* + x_2^* = X_0^*$

Controlling a trajectory

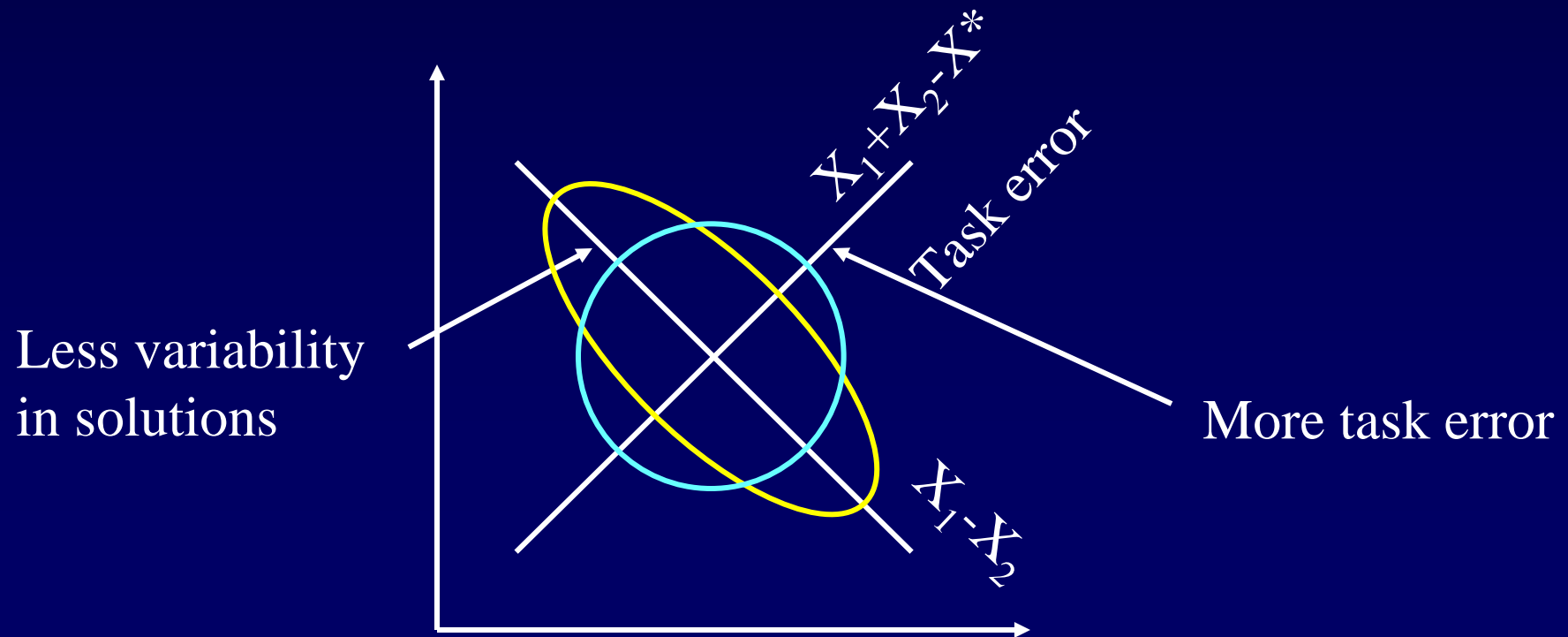
$$x_i^{\text{final}} = ax_i + u_i + u_i \sigma \varepsilon_i, \quad i \in \{1, 2\}$$

$$\mathbf{u} = \arg \min E_\varepsilon \left(x_1^{\text{final}} + x_2^{\text{final}} - X_0^* \right)^2 + r \left(u_1^2 + u_2^2 \right)$$



Controlling a trajectory

- Optimal motor control (Todorov-Jordan)



Open question

- How does the nervous system compute the solutions to those optimization problems?
- Is it done off line (i.e., does the CNS specify a trajectory?) or on line? Attractor nets?