We can describe the dynamic control problem as:

- We have a desired high fidelity trajectory to follow $\mathbf{x}(t), \mathbf{\dot{x}}(t), \mathbf{\ddot{x}}(t)$

Control system computes (usually in a fast loop) actuator commands (torques) to follow that trajectory

The purpose of a controller is frequently:

1) **Stabilize** a system about a particular desired configuration $\mathbf{x}_d$.
   Of special interest is the ability to stabilize around unstable equilibrium.

2) **Track** a known desired trajectory with high fidelity.
We have talked informally about some control schemes, and you have seen, especially linear control in 2004.

Today we will see a brief summary.

We commonly distinguish between two types of control methods:

- Feedforward
- Feedback

**Feedforward**

Suppose we have a system with this general equation of motion:

\[ M(\theta) \cdot \ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = z \]

If the equation were to be perfect (no unmodeled dynamics, no noise, no errors in estimating parameters,...) we could directly compute the desired torques for the desired trajectory.

This implements \( z(t) = M(\theta) \cdot \ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) \)

> It is also commonly known as INVERSE DYNAMICS. Note that it is much easier than INVERSE KINEMATICS.

> Feedforward control is PROACTIVE. It anticipates the needs, so it is very fast and has no theoretical time lag.

> Also called OPEN-LOOP control.
Unfortunately, due to imperfections in the model, the output of the manipulator will quickly diverge from intended behavior since errors will accumulate.

\[ \text{Work well for short periods of time.} \]

**FEEDBACK**

It relies on driving the controller by correcting differences between intended and observed behavior.

Error: \( \theta_d - \theta_{\text{REAL}} \)

Avoids diverging from the intended behavior by closing the loop:

Higher error \( \uparrow \) \( \Rightarrow \) Higher torque \( \uparrow \) \( \Rightarrow \) More correction

It is reactive in nature. Inevitably it leads to delays in tracking a trajectory, since it can only react to an error after it has happened (causality)

**FEEDFORWARD + FEEDBACK** For higher performance
- We are going to look at different classical techniques for feedback control.

- The main issue with feedback control is that the set of dynamic equations are non-linear and highly coupled.

- We will see three options:
  
  - **PID**
  - **LINEARIZE** equations of motion + Linear Feedback Design
  - **FEEDBACK** linearization.

**PID**

We forget that the system is non-linear, and add a linear controller nevertheless:

\[ M(t) = K_p \cdot \Delta \theta(t) + K_d \cdot d\Delta \theta(t) + K_i \int_0^t \Delta \theta(t) dt \]

\[ \text{PROPORTIONAL DERIVATIVE INTEGRAL} \]

- We choose the parameters \( K_p, K_d, K_i \) experimentally rather than looking at the system.

- There are algorithms that will do it for you (MATLAB's Control System Toolbox). They measure the response to excitations and then find good values for \( K_p, K_d, K_i \).

**LINEARIZATION**

- We have a non-linear system:

\[ f(x, \dot{x}, \ddot{x}, M) = 0 \]

we linearize it around a configuration of interest \( x_0 \).
The system then becomes a mass-spring-damper system which is linear, and to which you can apply linear feedback design.

\[ m \ddot{x} + b \dot{x} + kx = u \]

Now you can use classical analysis like Laplace transform, pole placing, root locus, ... .

The basic idea is that, although my linearized system might have undesired properties in terms of stability, natural resonance, over/under/critically damping, ... I can artificially change them by making my control law look like:

\[ u = -K_p \cdot x - K_d \cdot \dot{x} \] (PD controller)

Now the closed loop system is

\[ m \ddot{x} + (b + K_d) \dot{x} + (k + K_p) x = 0 \]

and have freedom to design \( K_d \) and \( K_p \) so that, for example, the system is stable and critically damped.

(We can add an integral term to remove steady-state errors)
A similar structure can be replicated for systems with multiple DOFs. Unfortunately, if they are coupled, the design of linear feedback controllers will be more difficult.

We usually note them in "state" space as:

\[
\dot{x} = Ax + Bu
\]

- \( x \) is \( n \)-dimensional
- \( A \) is \( nxn \) matrix
- \( B \) is \( nxr \) matrix
- \( \dot{x} \) is \( n \)-dimensional
- \( u \) is \( r \)-dimensional
- \( A \) is \( nxn \) matrix
- \( B \) is \( nxr \) matrix

For example:

\[
m\ddot{x} + b\dot{x} + Kx = f \quad \rightarrow \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}
\]

\[
\frac{d}{dt}(x) = \frac{dx}{dt} = \begin{pmatrix} \frac{b}{m} & -\frac{K}{m} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} \frac{m}{m} \\ 0 \end{pmatrix} \begin{pmatrix} f \end{pmatrix}
\]

\[
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\frac{b}{m} & -\frac{K}{m} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{pmatrix} \frac{m}{m} \\ 0 \end{pmatrix} \begin{pmatrix} f \end{pmatrix}
\]

This notation generalizes to \( n \)-dimensional linear systems.

A linear feedback law has the form:

\[
u(t) = -Kx = \begin{pmatrix} k_1 & k_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}
\]

accounts both for proportional and derivative terms.

\( K \) is called the state feedback gain.

Then:

\[
\dot{x} = Ax + Bu = (A - BK)x
\]

This matrix determines the properties of the controller.
There are algorithmic rules to design \( K \), but it is a complex problem that requires finding a tradeoff between controller performance (state error) and controller effort (actuator effort).

A widely used solution to find \( K \) is \textit{we optimization}:

**LINEAR QUADRATIC REGULATOR**

- Assume the goal is to stabilize around desired state \( x = 0 \)

![State Error](image)
![Controller Effort](image)

- We define a metric to minimize:

\[
V = \int_{0}^{t_{\text{final}}} (x^2(t) + r \cdot u^2(t)) \, dt
\]

or for multi-dimensional systems:

\[
V = \int_{0}^{t_{\text{final}}} (x^T(t) \cdot Q \cdot x(t) + u^T(t) \cdot R \cdot u(t)) \, dt
\]

- \( Q \) and \( R \) are positive definite matrices.

\( Q \) and \( R \) weight to balance state error and controller effort.

\( \uparrow \) you can give more weight to certain dimensions.

- LQR finds the feedback law \( u = -Kx(t) \) that minimizes \( V \)

- Efficient solutions based on the "Riccati Equation"

- MATLAB \texttt{"lqr(A,B,Q,R)"} outputs \( K \).
FEEDBACK LINEARIZATION

→ Linearization of the equation of motion is only valid for systems that are not too dynamic (otherwise, the coriolis and centrifugal terms become prominent) and for small deviations about the linearized state (otherwise, the dependence of gravity and inertia matrix with state becomes prominent).

→ Feedback linearization uses, instead, knowledge of the state to compensate instantaneously for non-linearities:

![Feedback Linearization Diagram]

called COMPUTED TORQUE CONTROL

1) Gravity compensation:

![Gravity Compensation Diagram]

Originally the system was: \( M(\theta) \cdot \ddot{\theta} + C(\theta, \dot{\theta}) \dot{\theta} + G(\theta) = 2 \)

Now the controller sees: \( M(\theta) \cdot \ddot{\theta} + C(\theta, \dot{\theta}) \dot{\theta} = z_m \)

This is the system that the PD has to deal with.
2) Dynamic Compensation:

\[ M(\theta) \ddot{\theta} + C(\theta, \dot{\theta}) \dot{\theta} + G(\theta) = M(\theta) \ddot{\theta} + \ddot{\theta} + \ddot{\theta} + \ddot{\theta} \]

\[ M(\theta) \ddot{\theta} = \dddot{\theta} \]

The system is much simpler but still non-linear (due to the dependence of \( M(\theta) \) with \( \theta \)) and coupled.

3) Full Feedback Linearization:

\[ M(\theta) \ddot{\theta} + C(\theta, \dot{\theta}) \dot{\theta} + G(\theta) = M(\theta) \ddot{\theta} + \ddot{\theta} + \ddot{\theta} + \ddot{\theta} \]

Now \[ M(\theta) \ddot{\theta} + C(\theta, \dot{\theta}) \dot{\theta} + \ddot{\theta} = M(\theta) \ddot{\theta} + \ddot{\theta} + \ddot{\theta} + \ddot{\theta} \]

\[ \ddot{\theta} = \dddot{\theta} \]

\[ \rightarrow \text{System of decoupled linear double integrators with inertia} \ 1 \]

\[ \rightarrow \text{Simple for the PD controller to stabilize and control} \]