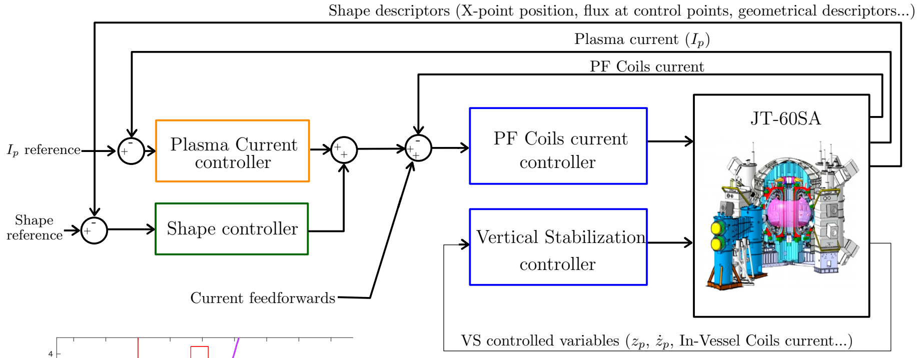
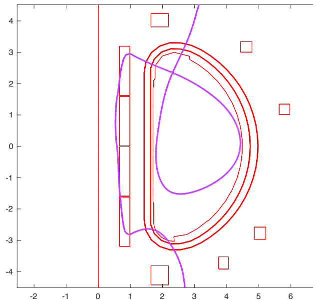
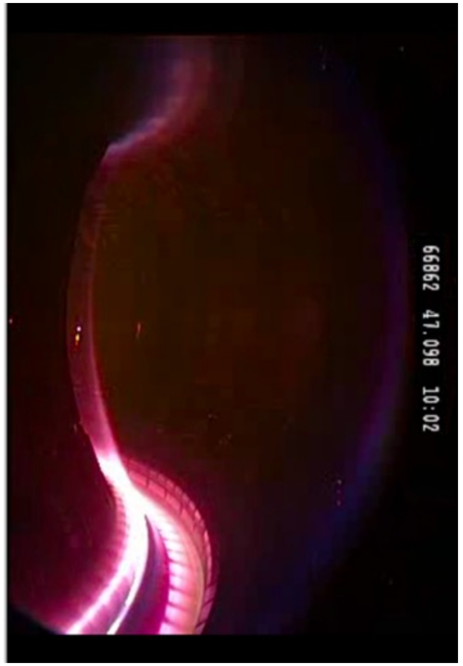
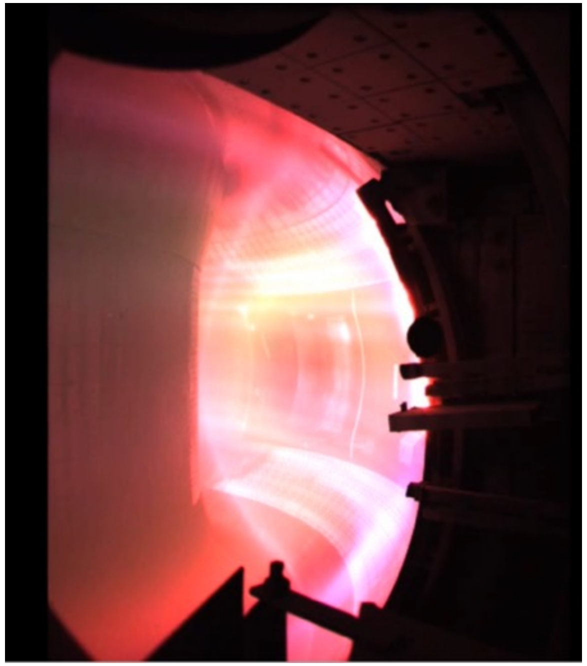
Introduction to basic plasma magnetic control

G. De Tommasi
University of Naples Federico II – CREATE Consortium
detommas@unina.it









This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

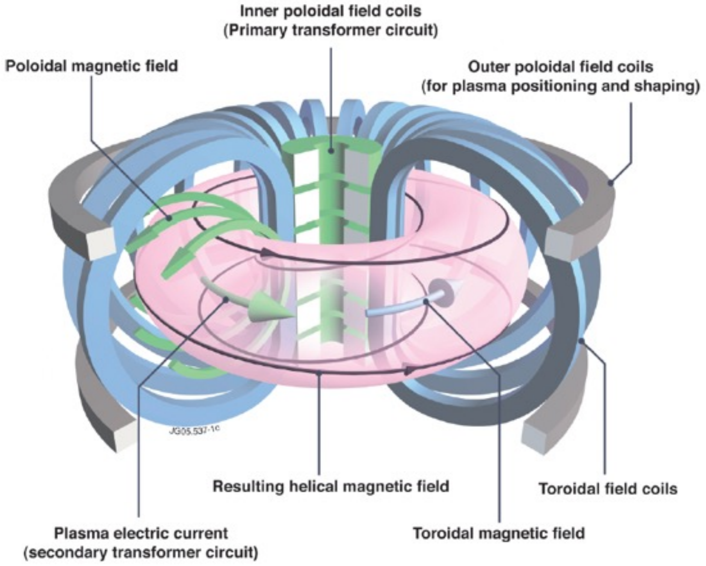
Outline

- **Brief overview of plasma magnetic control in tokamaks**
(see reference paper & book)

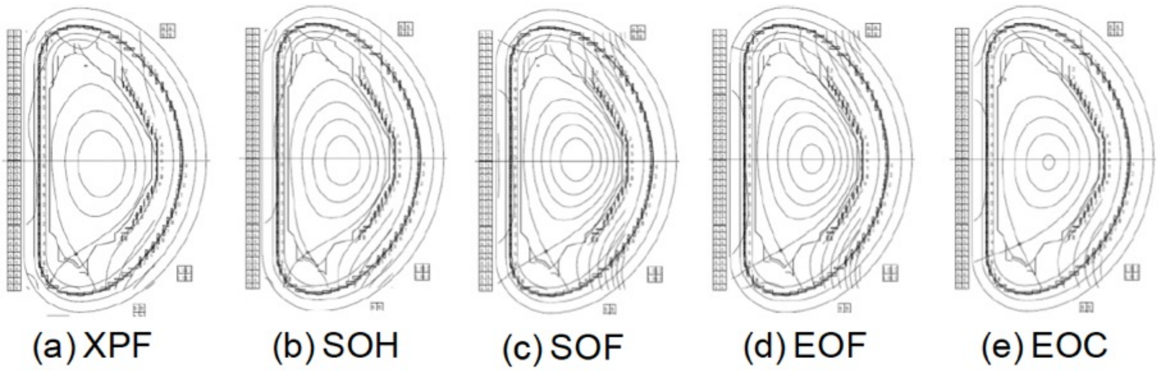
- **Introduction to systems and control theory** (see introductory material)
  - Linear Systems – The State-Space representation
  - Transfer functions & Block diagrams
  - Frequency response G(j**ω**) & its graphical representations
  - The control problem

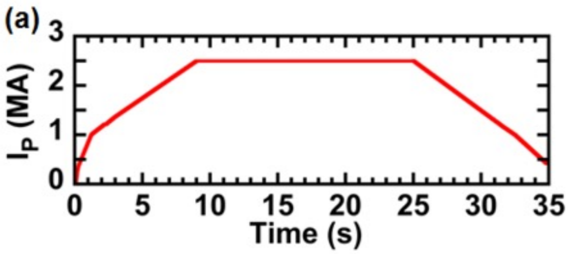
- **Practical #1**
- **Practical #2**
- **Practical #3**

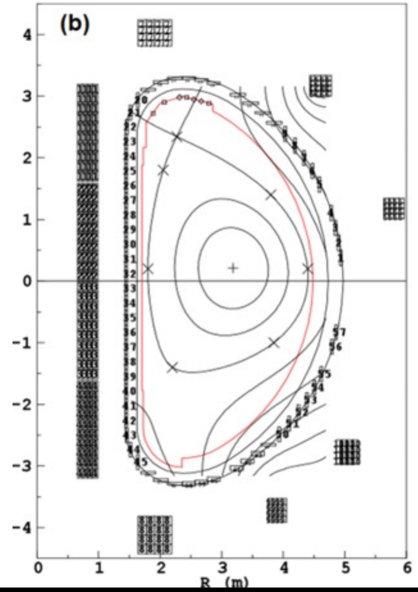
Plasma magnetic (equilibrium) control



Plasma scenario
Enrico di Pietro lecture Sep 6
Emmanuel Joffrin lecture Sep 7





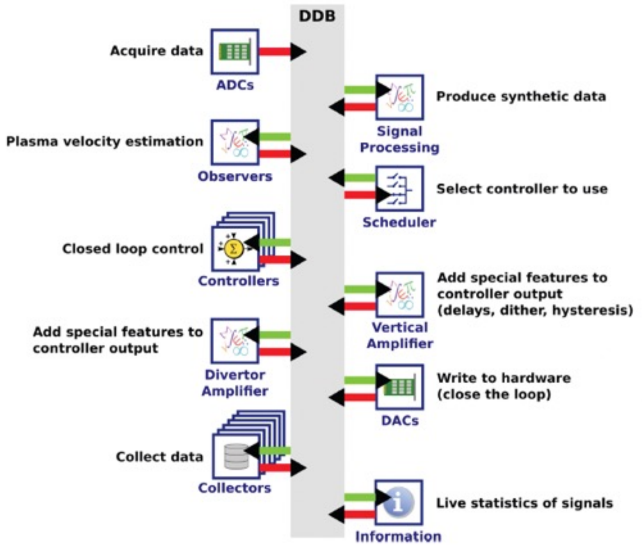


- **Magnetic control** is necessary to **run throughout the desired sequence of equilibria** (aka scenario) **despite model uncertainties and disturbances**
  - Plasma current needs to be controlled…
  - …as well as the shape of the boundary (LCFS)…
  - …as well as the vertical motion of elongated plasmas, since they are vertically unstable…
- Magnetic control of the plasma is obtained by means of the magnetic fields produced by the external active coils

Real actuators are the power supplies, i.e. we can act by applying voltage to the coils

Real sensors are pick-up coils and flux loops (Brian Peterson lecture Sep 6) + data acquisition systems (including integrators),…

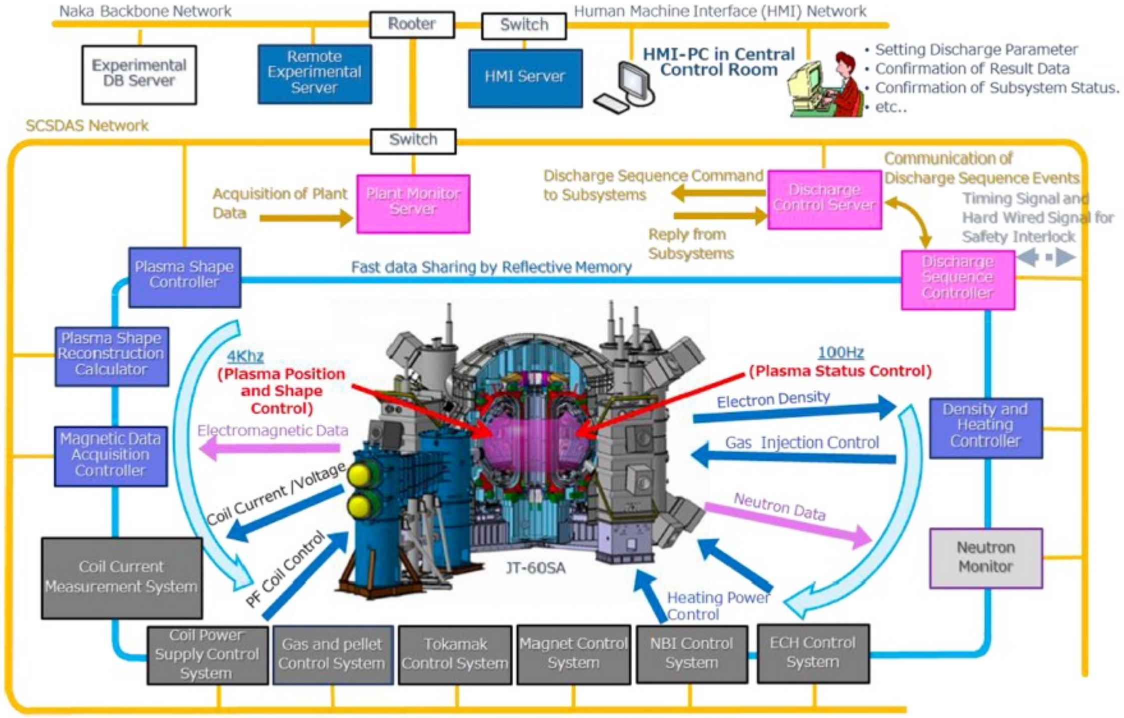
The final objective…build a control system …which is not just the control algorithms, there is more…



JET tokamak
Vertical Stabilization System

H. Urano presentation to TCM36 - 2021

Supervisory Control System and Data Acquisition System (SCSDAS)



Main (plasma magnetic) control problems & model-based design of control algorithms

The plasma (axisymmetric) magnetic control systems includes components to solve the following problems

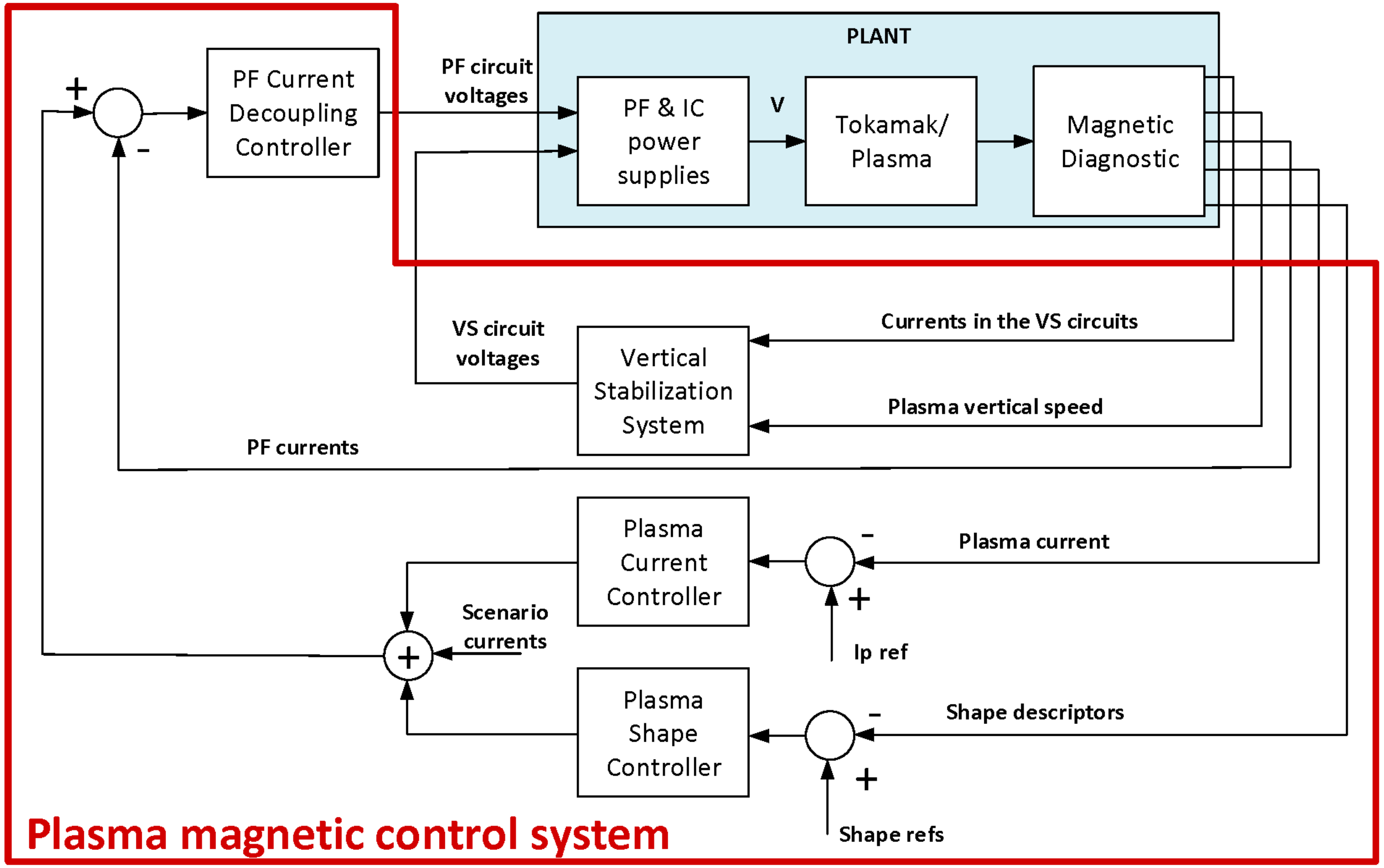
- the plasma current control problem
- the shape and position control problem
- the vertical stabilization problem

Since this is a school, during this practical lessons, I will refer to a ***machine-agnostic*** architecture

The main aim is to introduce you (the students) to **model-based design** of control algorithms…

…since this is JIFS, during the practical sessions, we will *play* with **JT-60SA models**

Reference control architecture (for our lectures)



Four independent controllers

- Current decoupling controller
- Vertical stabilization controller
- Plasma current controller
- Plasma shape controller

The structure and the parameters (gains) of each controller can change according to events generated by an external supervisor (to react to pre-programmed events or exceptions)

Needed since early machine operation, also IC → see Ide presentation 11 Sep

Nonlinear lumped parameters model

In order to adopt a model-based design approach **a model of the plasma and of the current in the surrounding conductive structures (both active and passive) is needed**

- Grad-Shafranov PDE can be solved by using finite-elements methods
- From these numerical approximations it is possible retrieve a **nonlinear lumped parameters** model

$$\frac{\mathrm{d}}{\mathrm{dt}}\Big[\mathcal{M}\big(\mathbf{y}(t), \beta_p(t), l_i(t)\big)\mathbf{I}(t)\Big] + \mathbf{R}\mathbf{I}(t) = \mathbf{U}(t)\,,$$

$$\mathbf{y}(t) = \mathcal{Y}\big(\mathbf{I}(t), \beta_p(t), l_i(t)\big)\,.$$

where:

- $\mathbf{y}(t)$ are the output to be controlled
- $\mathbf{I}(t) = \left[\mathbf{I}_{PF}^T(t)\ \mathbf{I}_e^T(t)\ I_p(t)\right]^T$ is the currents vector, which includes the currents in the active coils $\mathbf{I}_{PF}(t)$, the eddy currents in the passive structures $\mathbf{I}_e(t)$, and the plasma current $I_p(t)$
- $\mathbf{U}(t) = \left[\mathbf{U}_{PF}^T(t)\ \mathbf{0}^T\ 0\right]^T$ is the input voltages vector
- $\mathcal{M}(\cdot)$ is the mutual inductance nonlinear function
- $\mathbf{R}$ is the resistance matrix
- $\mathcal{Y}(\cdot)$ is the output nonlinear function

Plasma linear model (useful to apply model-based design)

Starting from the nonlinear lumped parameters model, following linearized **state-space model** can be obtained:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}\delta\mathbf{x}(t) + \mathbf{B}\delta\mathbf{u}(t) + \mathbf{E}\delta\dot{\mathbf{w}}(t), \quad (1)$$

$$\delta\mathbf{y}(t) = \mathbf{C}\ \delta\mathbf{I}_{PF}(t) + \mathbf{F}\delta\mathbf{w}(t), \quad (2)$$

where:

- **A**, **B**, **E**, **C** and **F** are the model matrices
- $\delta\mathbf{x}(t) = \left[\delta\mathbf{I}_{PF}^T(t)\ \delta\mathbf{I}_e^T(t)\ \delta I_p(t)\right]^T$ is the state space vector
- $\delta\mathbf{u}(t) = \left[\delta\mathbf{U}_{PF}^T(t)\ \mathbf{0}^T\ 0\right]^T$ are the input voltages variations
- $\delta\mathbf{w}(t) = \left[\delta\beta_p(t)\ \delta l_i(t)\right]^T$ are the β_p and l_i variations
- $\delta\mathbf{y}(t)$ are the output variations

The model (1)–(2) relates the variations of the PF currents to the variations of the outputs around a given equilibrium

System theory jargon

State space model

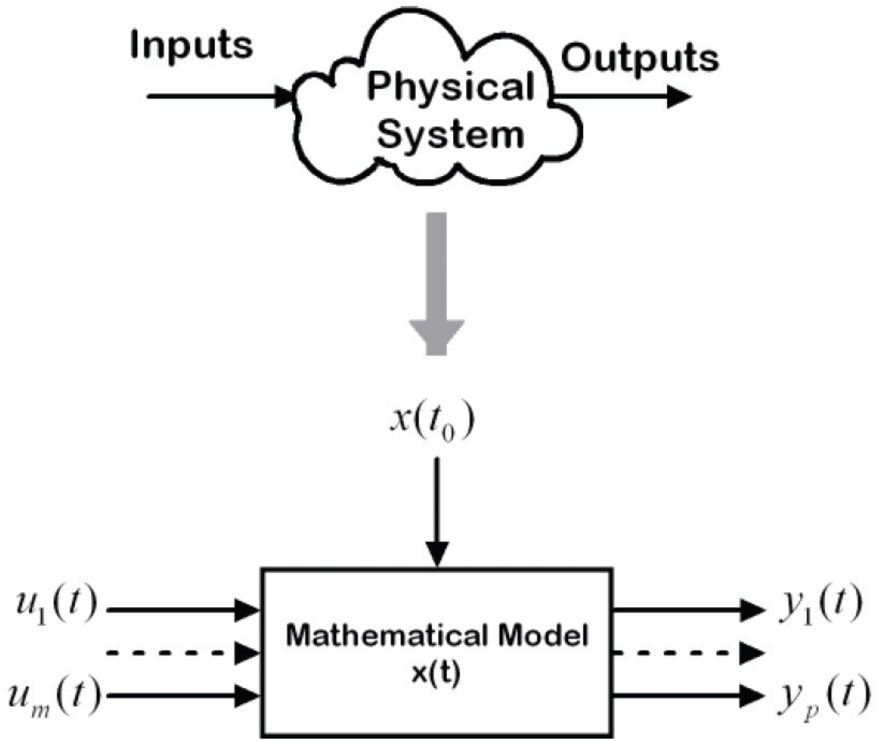
A **finite dimensional** *continuous-time* dynamical system can be described by the following differential equations:

$$\dot{x}(t) = f\big(x(t)\,,u(t)\,,t\,,t_0\big)\,,x(t_0) = x_0 \qquad (3a)$$

$$y(t) = h\big(x(t)\,,u(t)\,,t\,,t_0\big) \qquad (3b)$$

where:

- $x(t) \in \mathbb{R}^n$ is the system **state**
- $x(t_0) \in \mathbb{R}^n$ is the **initial condition**
- $u(t) \in \mathbb{R}^m$ is the **input** vector
- $y(t) \in \mathbb{R}^p$ is the **output** vector
- n is the **order** of the system



Linear time-invariant (LTI) systems (models?)

The state space model for a **linear time-invariant (LTI)** continuous-time system can written as

$$\dot{x}(t) = Ax(t) + Bu(t)\,, \quad x(0) = x_0 \tag{4a}$$

$$y(t) = Cx(t) + Du(t) \tag{4b}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$.

A dynamical system with single-input ($m = 1$) and single-output ($p = 1$) is called *SISO*, otherwise it is called *MIMO*.

Matlab commands

`sys = ss(A,B,C,D)` creates a state space model object.
`y = lsim(sys,u,t)` simulates the time response of the LTI system `sys`.

Equilibria of a nonlinear system

Consider a nonlinear and **time-invariant** system

$$\dot{x}(t) = f\big(x(t)\,, u(t)\big)\,, \quad x(0) = x_0 \tag{5a}$$

$$y(t) = h\big(x(t)\,, u(t)\big) \tag{5b}$$

If the input is constant, i.e. $u(t) = \bar{u}$, then the equilibrium states $x_{e_1}\,, x_{e_2}\,, \ldots\,, x_{e_q}$ of such a system can be computed as solutions of the homogeneous equation

$$f\big(x_e\,, \bar{u}\big) = 0\,,$$

Given an equilibrium state x_{e_i} the correspondent output is given by

$$y_{e_i} = h\big(x_{e_i}\,, \bar{u}\big)\,.$$

Linearization around a given equilibrium

If $x_0 = x_e + \delta x_0$ and $u(t) = \bar{u} + \delta u(t)$, with $\delta x_0\,, \delta u(t)$ **sufficiently small**, then the behaviour of (5) around a given equilibrium point $(\bar{u}\,, x_e)$ is well described by the linear system

$$\delta\dot{x}(t) = \left.\frac{\partial f}{\partial x}\right|_{\substack{x = x_e \\ u = \bar{u}}} \delta x(t) + \left.\frac{\partial f}{\partial u}\right|_{\substack{x = x_e \\ u = \bar{u}}} \delta u(t)\,, \quad \delta x(0) = \delta x_0 \tag{6a}$$

$$\delta y(t) = \left.\frac{\partial h}{\partial x}\right|_{\substack{x = x_e \\ u = \bar{u}}} \delta x(t) + \left.\frac{\partial h}{\partial u}\right|_{\substack{x = x_e \\ u = \bar{u}}} \delta u(t) \tag{6b}$$

The total output can be computed as

$$y(t) = h(x_e\,, \bar{u}) + \delta y(t)\,.$$

Asymptotic stability of LTI systems

Asymptotic stability

This property roughly asserts that every solution of $\dot{x}(t) = Ax(t)$ tends to zero as $t \to \infty$.

Note that for LTI systems the stability property is related to the system and not to a specific equilibrium.

Theorem - System (4) is **asymptotically stable iff** A is Hurwitz, that is if every eigenvalue λ_i of A has strictly negative real part

$$\Re(\lambda_i) < 0 \,, \forall\ \lambda_i \,.$$

Theorem - System (4) is **unstable if** A has at least one eigenvalue $\bar{\lambda}$ with strictly positive real part, that is

$$\exists\ \bar{\lambda}\ \text{s.t.}\ \Re(\bar{\lambda}) > 0 \,.$$

Stability of an equilibrium

For nonlinear system the stability property is related to the specific equilibrium

Theorem - The equilibrium state x_e corresponding to the constant input $\bar{u}$ a nonlinear system (5) is **asymptotically stable if** all the eigenvalues of the correspondent linearized system (6) have strictly negative real part.

Theorem - The equilibrium state x_e corresponding to the constant input $\bar{u}$ a nonlinear system (5) is **unstable if** there exists at least one eigenvalue of the correspondent linearized system (6) which has strictly positive real part.

Transfer function of LTI systems

Given a LTI system (4) the corresponding *transfer matrix* from u to y is defined as

$$G(s) = C(sI - A)^{-1}B + D\,, \tag{7}$$

where $s \in \mathbb{C}$. If we denote with $U(s)$ and $Y(s)$ the Laplace transforms of $u(t)$ and $y(t)$, then it is

$$Y(s) = G(s)U(s)\,,$$

when the initial condition of system (4) is $x(0) = 0$.
For SISO system (7) is called *transfer function* and it is equal to the Laplace transform of the **impulsive response** of system (4) with zero initial condition.

Matlab commands

`sys = tf(num,den)` creates a transfer function object.

Analysis in the Laplace domain

Given the transfer function $G(s)$ and the Laplace transform of the input $U(s)$ the time response of the system can be computed as the inverse transform of $G(s)U(s)$, without solving differential equations.

As an example, the **step response** of a system can be computed as:

$$y(t) = \mathcal{L}^{-1}\left[G(s)\frac{1}{s}\right].$$

Matlab commands

`[y,t] = step(sys)` computes the step response of the LTI system `sys`.

`[y,t] = impulse(sys)` computes the impulse response of the LTI system `sys`.

Zeros and poles of a transfer function

Given a SISO LTI system , its transfer function is a rational function of s

$$G(s) = \frac{N(s)}{D(s)} = \rho \frac{\Pi_i (s - z_i)}{\Pi_j (s - p_j)} \,,$$

where $N(s)$ and $D(s)$ are polynomial in s, with $\deg\big(N(s)\big) \leq \deg\big(D(s)\big)$.
We call

- p_j **poles** of $G(s) \to$ roots of $D(s)$
- z_i **zeros** of $G(s) \to$ roots of $N(s)$

Matlab commands

`sys = zpk(z,p,k)` creates a zeros-poles-gain object.
`p = eig(sys)` or `p = pole(sys)` return the poles of the LTI system `sys`.
`z = zero(sys)` returns the zeros of the LTI system `sys`.

Poles and eigenvalues of LTI systems

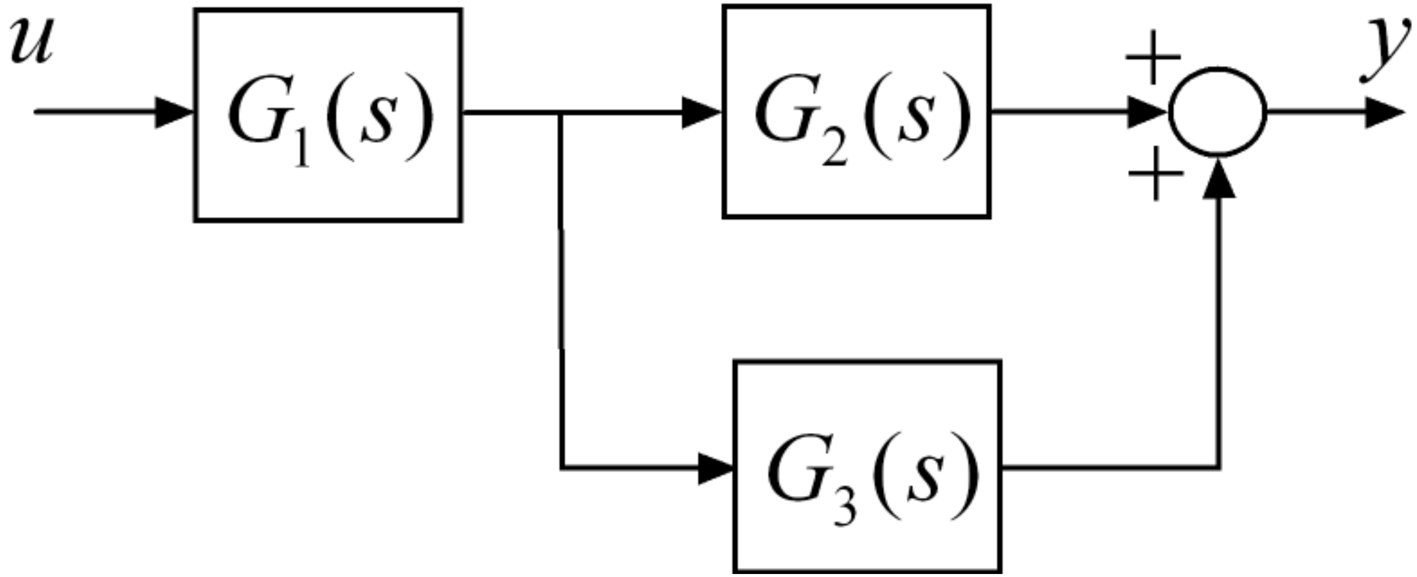
Each pole of G(s) is an eigenvalue of the system matrix A, while the converse is not necessarily true

If all the poles of G(s) have strictly negative real part – i.e. they are located in the left half of the *s*-plane (LHP) – the SISO system is said to be

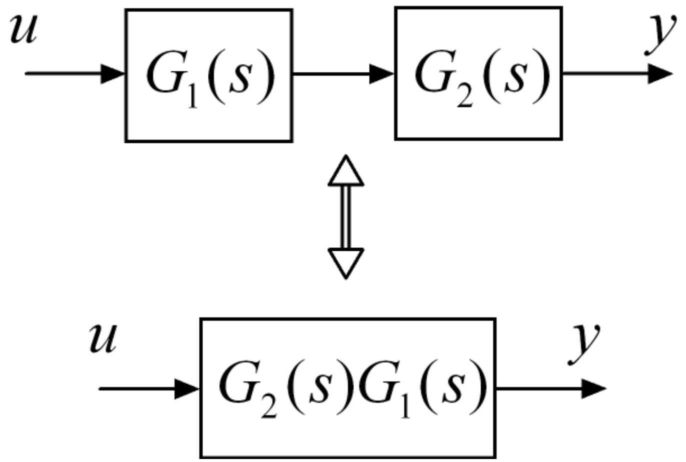
Bounded–Input Bounded–Output stable (BIBO)

A system is BIBO stable if bounded input to the system results in a bounded output over the time interval $[0, +\infty)$

When dealing with transfer functions, it is usual to resort to **block diagrams** which permit to graphically represent the interconnections between systems in a convenient way

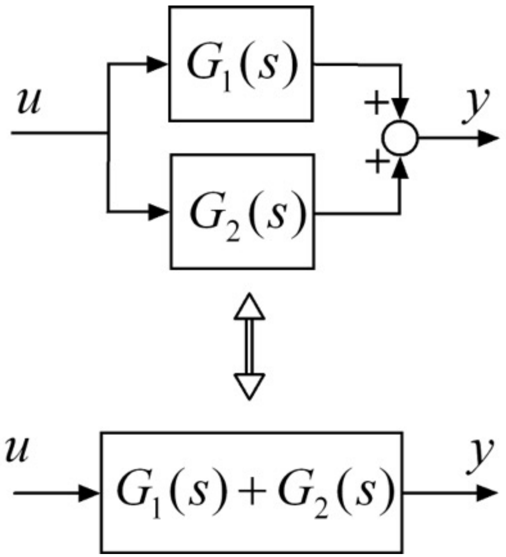


Series & parallel connections



Matlab commands

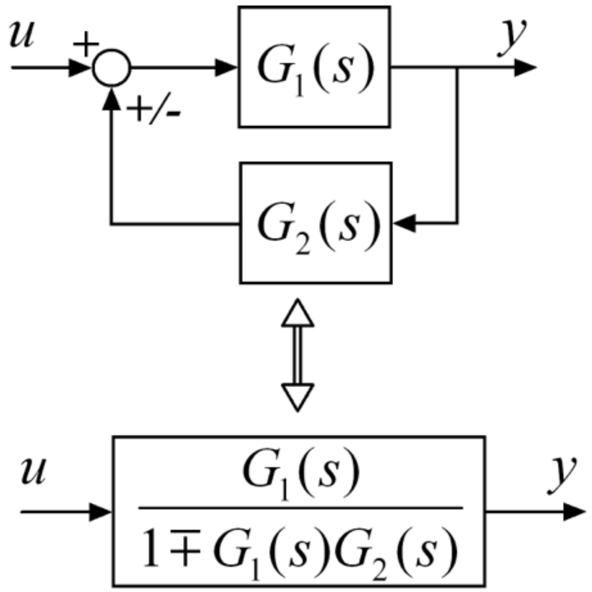
`sys = series(sys1,sys2)` or `sys = sys2*sys1` make the series interconnection between `sys1` and `sys2`.



Matlab commands

`sys = parallel(sys1,sys2)` or `sys = sys1+sys2` make the parallel interconnection between `sys1` and `sys2`.

The feedback connection



Matlab commands

`sys = feedback(sys1,sys2,[+1])` makes the feedback interconnection between `sys1` and `sys2`. Negative feedback is the default. If the third parameter is equal to `+1` positive feedback is applied.

Given **two asymptotically** stable LTI systems $G_1(s)$ and $G_2(s)$

- the **series** connection $G_2(s)G_1(s)$ **is asymptotically stable**
- the **parallel** connection $G_1(s) + G_2(s)$ **is asymptotically stable**
- the **feedback** connection $G_1(s)/(1 \pm G_1(s)G_2(s))$ **is not necessarily stable**

THE CURSE OF FEEDBACK!

The frequency response

Given a LTI system the complex function

$$G(j\omega) = C\left(j\omega I - A\right)^{-1}B + D\,,$$

with $\omega \in \mathbb{R}^+$ is called *frequency response* of the system.

$G(j\omega)$ permits to evaluate the system steady-state response to a sinusoidal input. In particular if

$$u(t) = A\sin(\bar{\omega}t + \varphi)\,,$$

then the steady-state response of a LTI system is given by

$$y(t) = \left|G(j\bar{\omega})\right|A\sin\Big(\bar{\omega}t + \varphi + \angle G(j\bar{\omega})\Big)\,.$$

Graphical representation of G(jω) – The Bode plot

$$G(s) = 10\frac{1+s}{s\left(\frac{s^2}{400} + 2\frac{0.3}{20}s + 1\right)} = 10\frac{1+s}{s(0.0025s^2 + 0.03s + 1)}$$

Given a LTI system $G(s)$ the Bode diagrams plot

- the magnitude of $G(j\omega)$ (in dB, $\left|G(j\omega)\right|_{dB} = 20\log_{10}\left|G(j\omega)\right|$)
- and the phase of $G(j\omega)$ (in degree)

as a function of ω (in rad/s) in a semi-log scale (base 10).

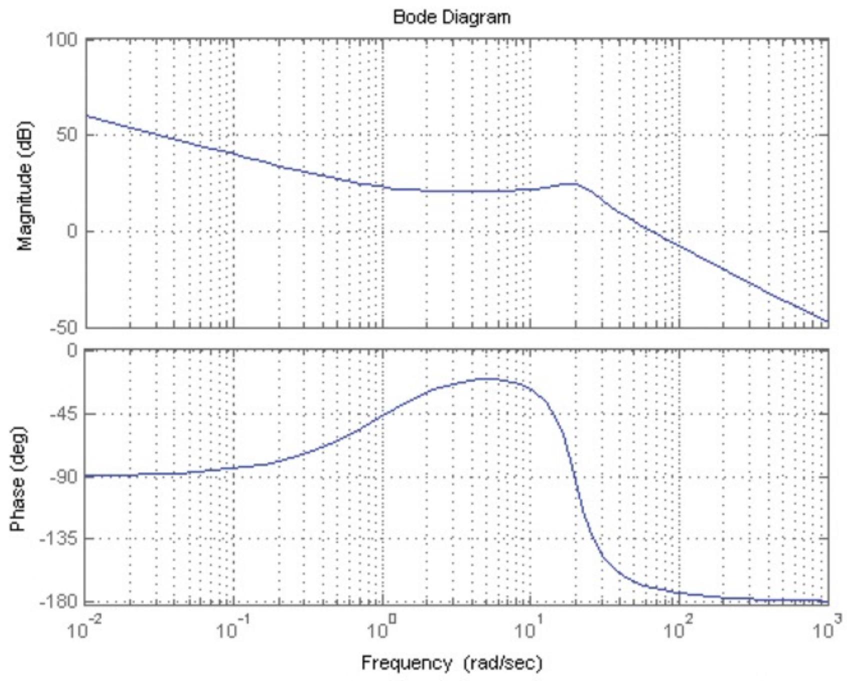
Bode plots are used for both analysis and synthesis of control systems.

Matlab commands

`bode(sys)` plots the the Bode diagrams of the LTI system `sys`.
`bodemag(sys)` plots the Bode magnitude diagram of the LTI system `sys`.

Matlab commands

```
s = tf('s');
sys = 10*(1+s)/(s*(s^2/400+0.6*s/20+1));
bode(sys);
grid
```

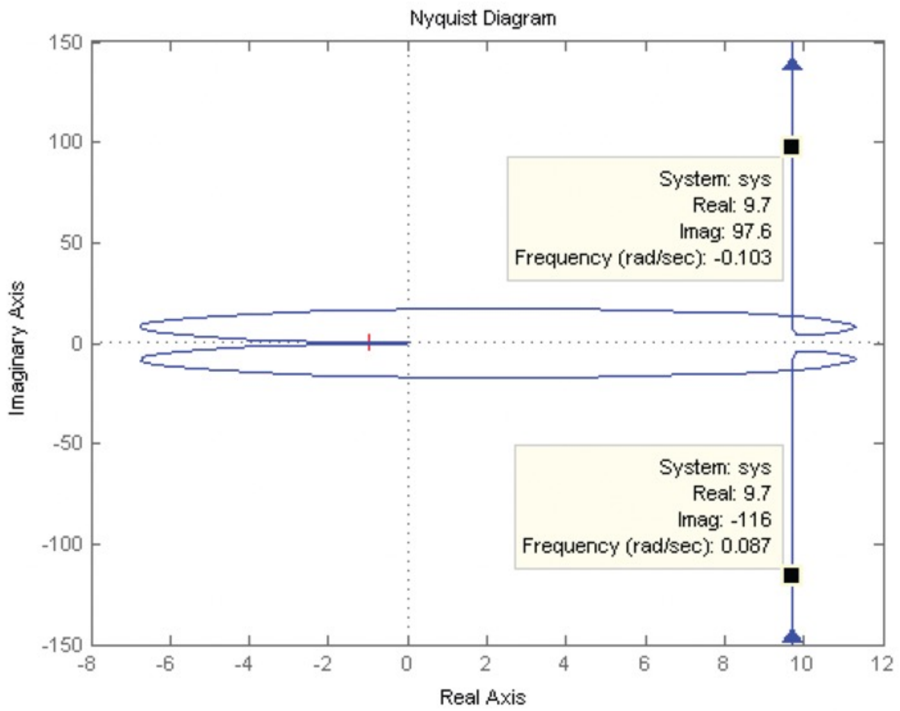


Graphical representation of $G(j\omega)$ – The Nyquist plot

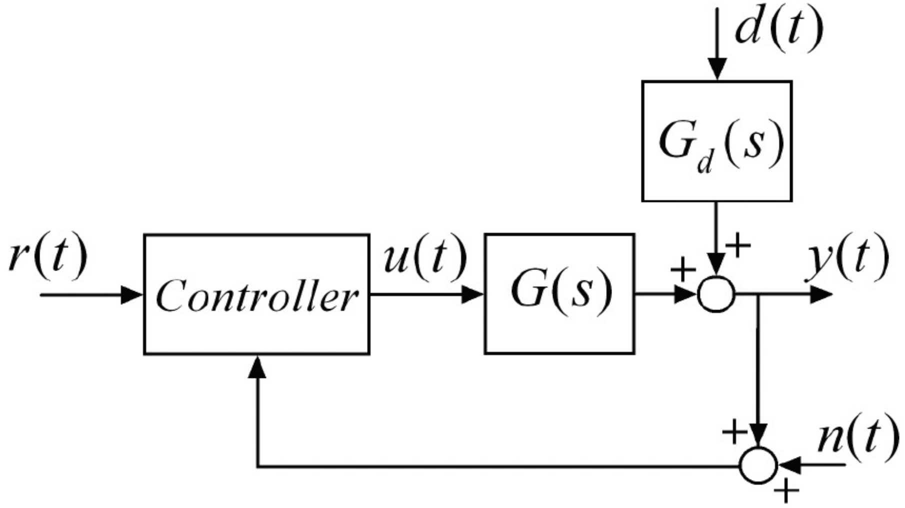
- The Nyquist plot is a polar plot of the frequency response $G(j\omega)$ on the complex plane
- This plot combines the two Bode plots - magnitude and phase - on a single graph, with frequency $\boldsymbol{\omega}$ as a parameter along the curve, ranging in $(-\infty, +\infty)$

- **Nyquist plots are useful to check stability of closed-loop systems**

$$G(s) = 10\frac{1+s}{s\left(\frac{s^2}{400} + 2\frac{0.3}{20}s + 1\right)}$$



The control *game*



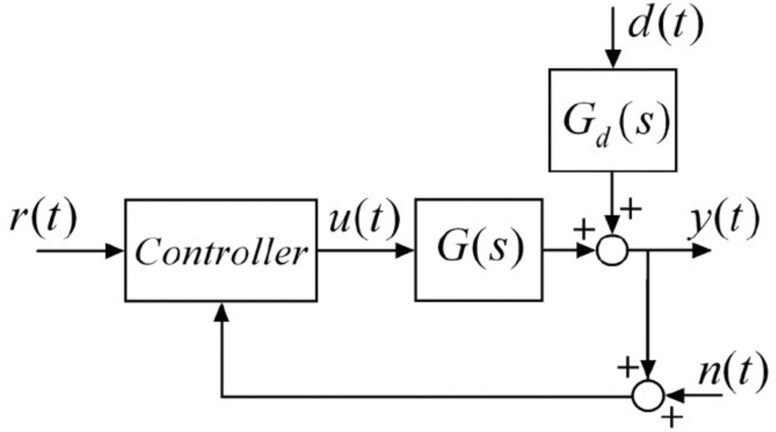
The objective of a control system is to make the output of a plant *y(t)* behaving in a desired way, by manipulating the plant input *u(t)*

A good controller should manipulate *u(t)* so as to

- counteract the effect of disturbances *d(t)* and measurement noise *n(t)* (**regulator problem**)
- keep the output close to a given reference input *r(t)* (**servo problem**)
- compensate for model uncertainties (**robustness against uncertainties**)

The objective is always to keep the control error *e(t) = r(t)-y(t)* as small as possible

Why (negative) feedback?

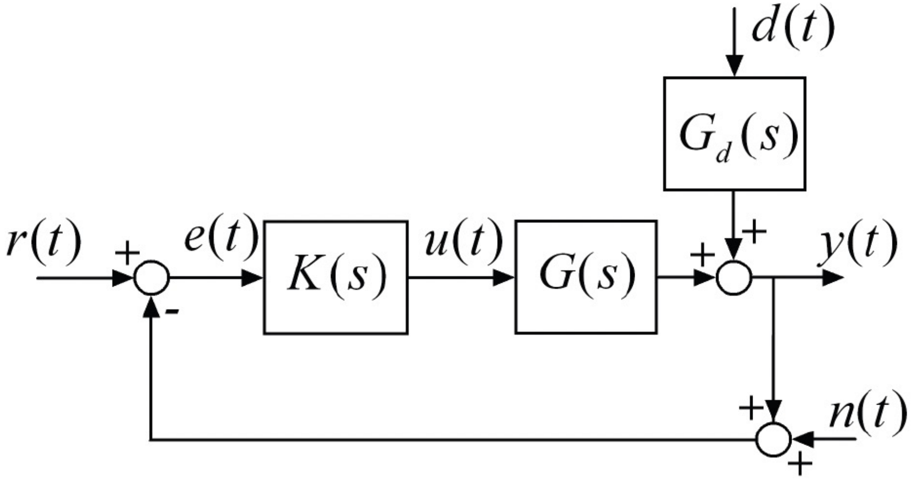


The main sources of difficulty in achieving good control performance are:

1. the plant model *G(s)* and the disturbance model $G_d(s)$ are affected by uncertainty and/or may change with time
2. the disturbances cannot be measured
3. the plant can be unstable

- It turns out that open-loop approaches are not (robust) enough (**we will experience this during Practical #1**)
- A feedback approach can guarantee the desired degree of robustness
- However, design a feedback control system is not straightforward since **instability is around the corner!**

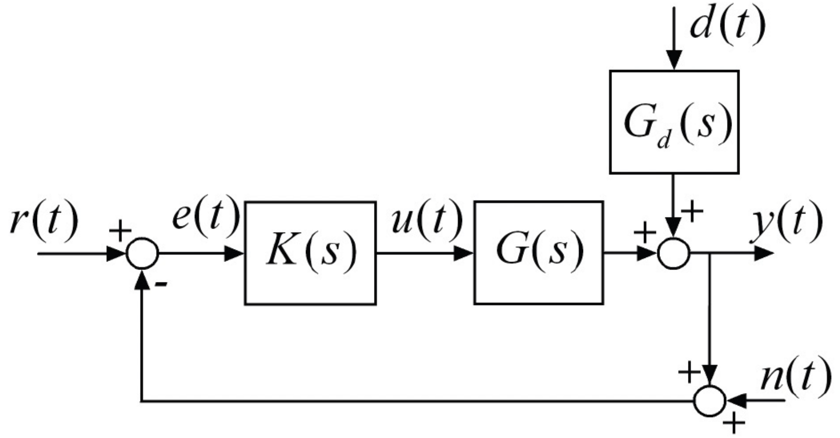
One degree-of-freedom feedback controller



The input to the plant is given by

$$U(s) = K(s)[R(s) - Y(s) - N(s)]$$

The objective of control is to design a controller $K(s)$ such that the control error $e(t) = r(t) - y(t)$ remains small despite the presence of disturbances $d(t)$ and noise $n(t)$

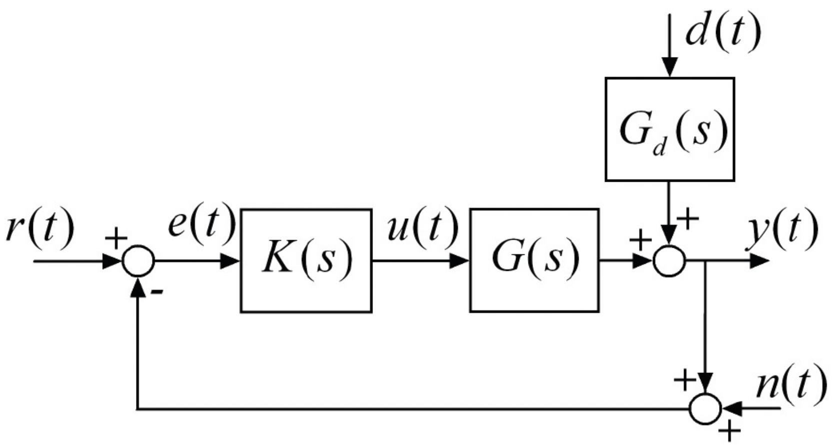


- $L(s) = G(s)K(s)$ is called **loop transfer function**
- $S(s) = \left(I + L(s)\right)^{-1}$ is called **sensitivity function**
- $T(s) = \left(I + L(s)\right)^{-1} L(s)$ is called **complementary sensitivity function**

It is straightforward to note that

$$T(s) + S(s) = I\,.$$

Feedback control - Terminology and notation



- $L(s) = G(s)K(s)$ is called **loop transfer function**
- $S(s) = \left(I + L(s)\right)^{-1}$ is called **sensitivity function**
- $T(s) = \left(I + L(s)\right)^{-1}L(s)$ is called **complementary sensitivity function**

It is straightforward to note that

$$T(s) + S(s) = I\,.$$

Exploiting the composition rules for block diagrams, it turns out that

$$Y(s) = T \cdot R(s) + SG_d \cdot D(s) - T \cdot N(s) \qquad (8a)$$
$$E(s) = -S \cdot R(s) + SG_d \cdot D(s) - T \cdot N(s) \qquad (8b)$$
$$U(s) = KS \cdot R(s) - K(s)S(s)G_d \cdot D(s) - KS \cdot N(s) \qquad (8c)$$

The control dilemma

Let consider

$$Y(s) = T \cdot R(s) + SG_d \cdot D(s) - T \cdot N(s)\,.$$

- In order to reduce the effect of the disturbance $d(t)$ on the output $y(t)$, the sensitivity function $S(s)$ should be made small (particularly in the *low frequency* range)
- In order to reduce the effect of the measurement noise $n(t)$ on the output $y(t)$, the complementary sensitivity function $T(s)$ should be made small (particularly in the *high frequency* range)

However, for all frequencies it is

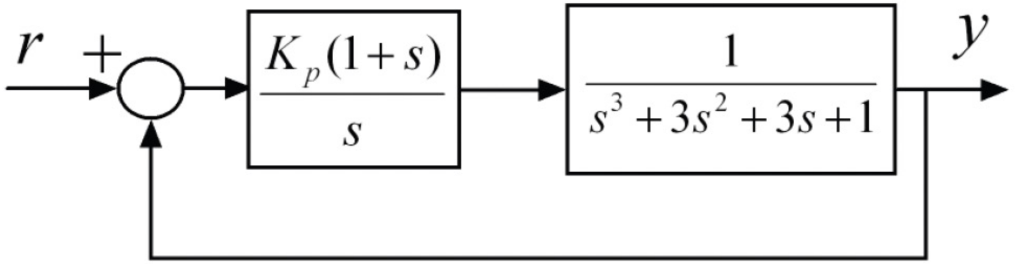
$$T + S = I\,.$$

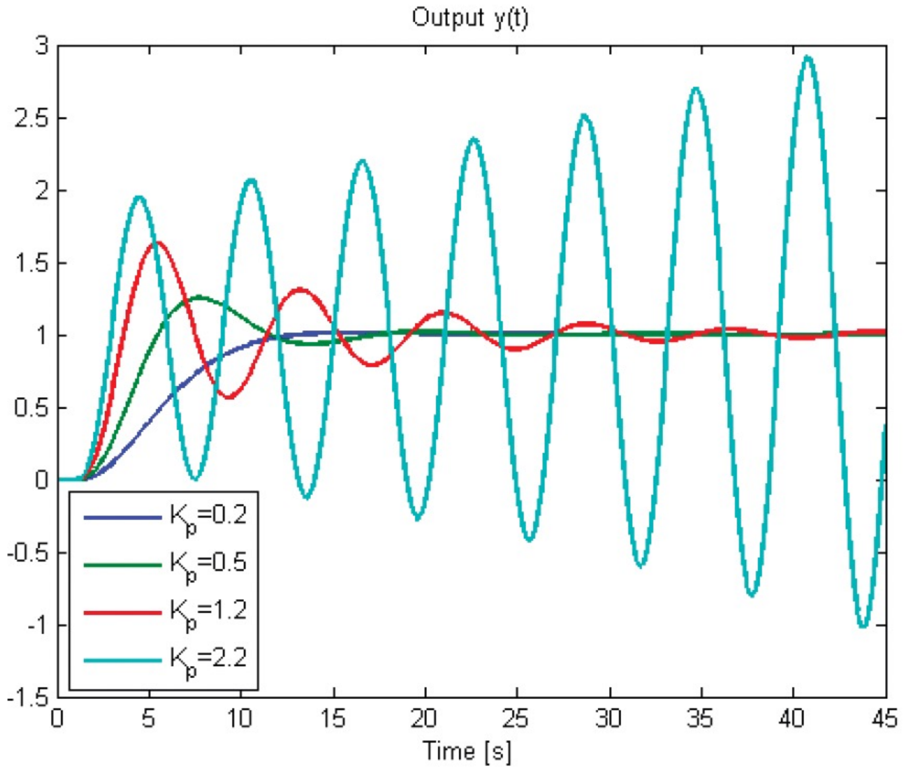
Thus a trade-off solution must be achieved.

Feedback may cause instability

One of the issues in designing feedback controllers is stability

If the feedback gain is too large, then the controller may overreact and the closed-loop system becomes unstable



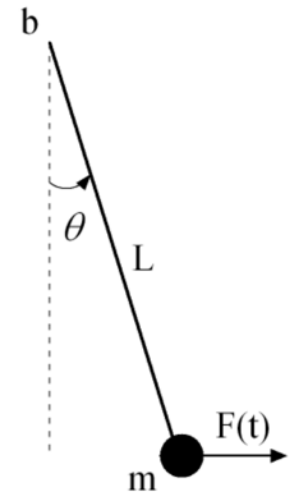


Practical #1 – Experience the basic concepts

- A **Matlab LiveScript** to play with a simple example: the control of the pendulum position
- We will see:
  - the effect of model uncertainties and disturbances
  - why open-loop control is not a good idea
  - the benefit of closed-loop control
  - the drawback of a poorly designed feedback control loop

Why closed-loop control?

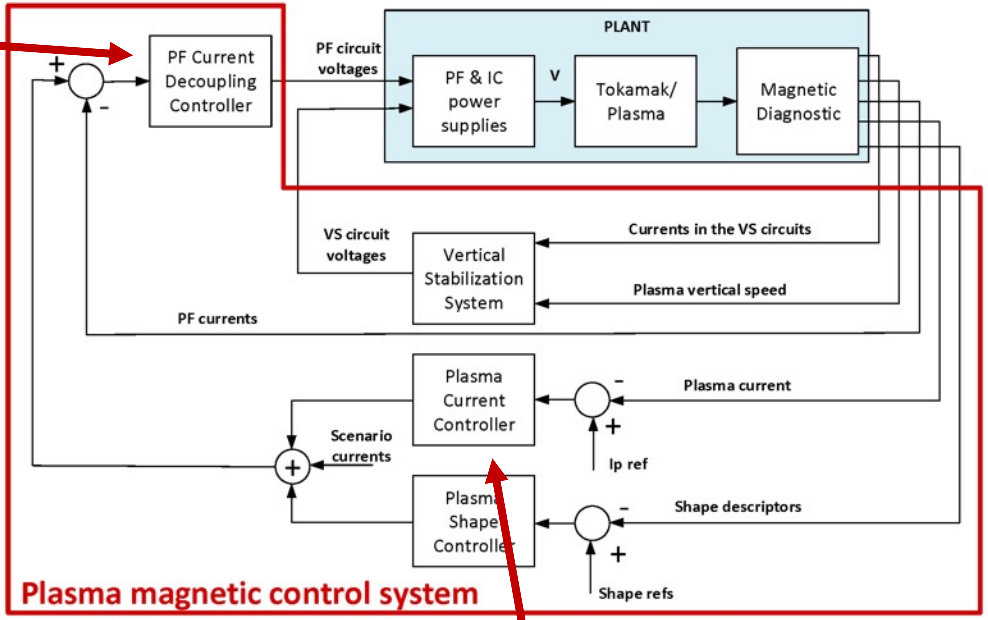
A simple example: control of a pendulum



- m is the pendulum mass
- L is the pendulum length
- b is the rotational friction
- θ is the pendulum rotation (wrt the vertical axis)

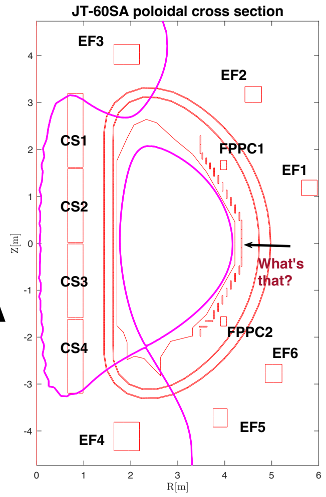
Practical #2 – Design a MIMO control
The case of a PFC decoupling controller

- Model-based design of the **PFC Decoupling Controller**
- With a **Matlab LiveScript** we will:
  - design a set of SISO PI to track currents in the PF circuits
  - design a MIMO controller starting from a plasmaless model of JT-60SA
  - compare the different solutions
  - evaluate the effect of the plasma on the designed system



We will also add the **Plasma Current Controller**

Since this is the JIFS…
…we will play with JT-60SA models



Practical #3 – Let's try to stabilize the plasma!

- Design a robust **VS** for JT-60SA
- Again we will start from a **Matlab LiveScript**
  - this will be less easy, but we will do it ;)