REFRESHING CONCEPTS

In this lecture we revisit some necessary concepts, which we have learnt in previous key courses.

- State space models for continuous time systems.
- LTI systems, Linear time variant systems, and nonlinear systems.
- Approximate discrete time models. Euler approximation, for non LTI cases.
- Classic Observers (continuous time, for LTI systems.)
- Coordinate frames. Transformations. Rotation matrix in 2D (today) and in 3D (later).

REFRESHING CONCEPTS

WHY?

State space models: because we use those.

Nonlinear systems: usually that is the case (our case of study is NL).

Approximate **discrete time** models: we operate in a discrete time fashion using digital computers, for control, estimation, etc.,

Classic Observers: to appreciate the need of the more powerful approaches which we will discuss later.

Coordinate frames. Transformations. Rotation matrix: we use those, in our case of study.

STATE SPACE REPRESENTATION

We usually need to have approximate models of the physical processes we want to understand/control/estimate.

And those models are usually (and conveniently) expressed in a state space representation.

(We are not saying it is the only way, but we should know that it is a convenient and popular way.)

State space Representation

Sometimes, we get that state space model directly, based on our understanding the real process. In other cases, we may have obtained the model in a different representation (e.g. an ODE) for then converting it to a state space model.

We have experience in doing it, from MMAN3200 (and from MTRN3020)

STATE SPACE REPRESENTATION

The general expression, for a continuous time (i.e. analog) system is:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right)$$

$$\mathbf{y}(t) = \mathbf{h}\left(\mathbf{x}(t), \mathbf{u}(t)\right)$$

in which:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ .. \\ x_n(t) \end{bmatrix}, \quad \dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ .. \\ \dot{x}_n(t) \end{bmatrix}, \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ .. \\ u_r(t) \end{bmatrix} \quad \mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ .. \\ y_p(t) \end{bmatrix}$$

$$\mathbf{x}(t) \in \mathrm{R}^n; \quad \dot{\mathbf{x}}(t) \in \mathrm{R}^n; \quad \mathbf{u}(t) \in \mathrm{R}^r; \quad \mathbf{y}(t) \in \mathrm{R}^p;$$

$$\mathbf{f}(\ ) : \mathrm{R}^{n+r} \rightarrow \mathrm{R}^n \quad ; \quad \mathbf{h}(\ ) : \mathrm{R}^{n+r} \rightarrow \mathrm{R}^p$$

STATE SPACE REPRESENTATION

As we know that all those involved variables are usually changing through the time, we may omit explicitly indicating "***(t)***" in our expressions:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

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

in which:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ .. \\ \dot{x}_n \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ .. \\ u_r \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ .. \\ y_p \end{bmatrix}$$

$$\mathbf{x} \in R^n; \quad \dot{\mathbf{x}} \in R^n; \quad \mathbf{u} \in R^r; \quad \mathbf{y} \in R^p;$$

State space Representation

For a LTI system, we get a particular family of models

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

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

Which results in:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u}$$

$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathbf{D} \cdot \mathbf{u}$$

(usually called “standard form”)

State space Representation

For a Time varying (or time variant) system, we get:

$$\dot{\mathbf{x}} = \mathbf{A}(t) \cdot \mathbf{x} + \mathbf{B}(t) \cdot \mathbf{u}$$
$$\mathbf{y} = \mathbf{C}(t) \cdot \mathbf{x} + \mathbf{D}(t) \cdot \mathbf{u}$$

(that is already escaping what we know from MMAN3200/MTRN3020.)

Examples of that? (we mention few cases, in class)

STATE SPACE REPRESENTATION

Nonlinear cases: we have the general expression

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}\left(\mathbf{x}, \mathbf{u}\right)$$

$$\mathbf{y} = \mathbf{h}\left(\mathbf{x}, \mathbf{u}\right)$$

In which the involved functions, **f**() and **h**() are nonlinear. In particular **f**() is the concerning one.

(that is well outside of MMAN3200/MTRN3020's concepts)

Examples? (we mention few cases, in class)

STATE SPACE REPRESENTATION

Nonlinear cases: Linear approximation.

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x},\mathbf{u}); \qquad \mathbf{y} = \mathbf{h}(\mathbf{x},\mathbf{u})$$

In certain cases, it is still possible to approximate the model by using a linear approximation, valid in a "region of operation", i.e., a region of the state space in which the approximating linear function is a good enough approximation. Typical case is using a first order Taylor approximation, evaluated at a "point of operation".

(as it have been seen in MMAN3200)

However, in certain cases, there is no "good enough" linearization.
→ Why can that be? Think about an example.

Examples of NL models (operating in a way not linearizable)

- The pendulum operating in a wide angular region, e.g. [-100,+100] degrees. (*)
- The kinematic model of a car, for a car, being used in a non short trip (usual operation of a car). (*)

If we linearized those models, the discrepancy between the real model and approximating linear one would be too large.

(*) we see this example models soon, in this lecture.

State space Representation

Modelling our processes of interest.

We are not discussing it in detail (that is for MMAN3200/MTRN3020)

In MTRN4010, we will model our cases by applying:

* Direct modeling (physical interpretation). We will do it for our kinematic models.

or

* From a given ODE (for certain SISO cases we may include in examples.)

State space Representation

For a system model from an ODE (usually high order, usually nonlinear)

$$\frac{d^n z}{dt^n} = f\left(u(t), \frac{d^{n-1} z}{dt^{n-1}}, ..., \frac{dz}{dt}, z\right)$$

but which does not involve derivatives of the input
(for LTI cases you know how to solve it, from MMAN3200)

We use an easy approach, similar to that of the Control Canonical Form, seen in MMAN3200 , for choosing states.

STATE SPACE REPRESENTATION

$$\frac{d^n z}{dt^n} = f\left(u(t), \frac{d^{n-1}z}{dt^{n-1}}, ..., \frac{dz}{dt}, z\right)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{bmatrix} = \begin{bmatrix} z \\ \frac{dz}{dt} \\ \frac{d^2 z}{dt^2} \\ \frac{d^{n-1} z}{dt^{n-1}} \end{bmatrix}, \qquad \mathbf{u} = u, \qquad \mathbf{y} = z$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \frac{dx_1}{dt} \\ .. \\ .. \\ \frac{dx_{n-1}}{dt} \\ \frac{dx_n}{dt} \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ ... \\ x_n \\ f(u, \mathrm{x}_n, ..., x_1) \end{bmatrix}$$

STATE SPACE REPRESENTATION

Example ODE → State Space.

Pendulum (as seen in MMAN3200).
It is a particular case of:

$$\frac{d^n z}{dt^n} = f\left(u(t), \frac{d^{n-1} z}{dt^{n-1}}, \ldots, \frac{dz}{dt}, z\right)$$

A 2nd order ODE of the type:

$$\ddot{\varphi} = f\left(u, \dot{\varphi}, \varphi\right)$$

… pendulum (example)

$$\ddot{\varphi} = -10 \cdot \sin(\varphi) - 4 \cdot \dot{\varphi} + u(t)$$

For which we may propose these state variables:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varphi \\ \dot{\varphi} \end{bmatrix}$$

Resulting in this state equation:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(x_1, x_2, u) =$$

$$= \begin{bmatrix} f_1(x_1, x_2, u) \\ f_2(x_1, x_2, u) \end{bmatrix} = \begin{bmatrix} x_2 \\ -10 \cdot \sin(x_1) - 4 \cdot x_2 + u(t) \end{bmatrix}$$

Nothing stop you of choosing an alternative state space representation (of the infinite possible ones) . We simply expect you to know , at least, the basic approach mentioned here.

1) Case in which the “native” model is already in state space representation.

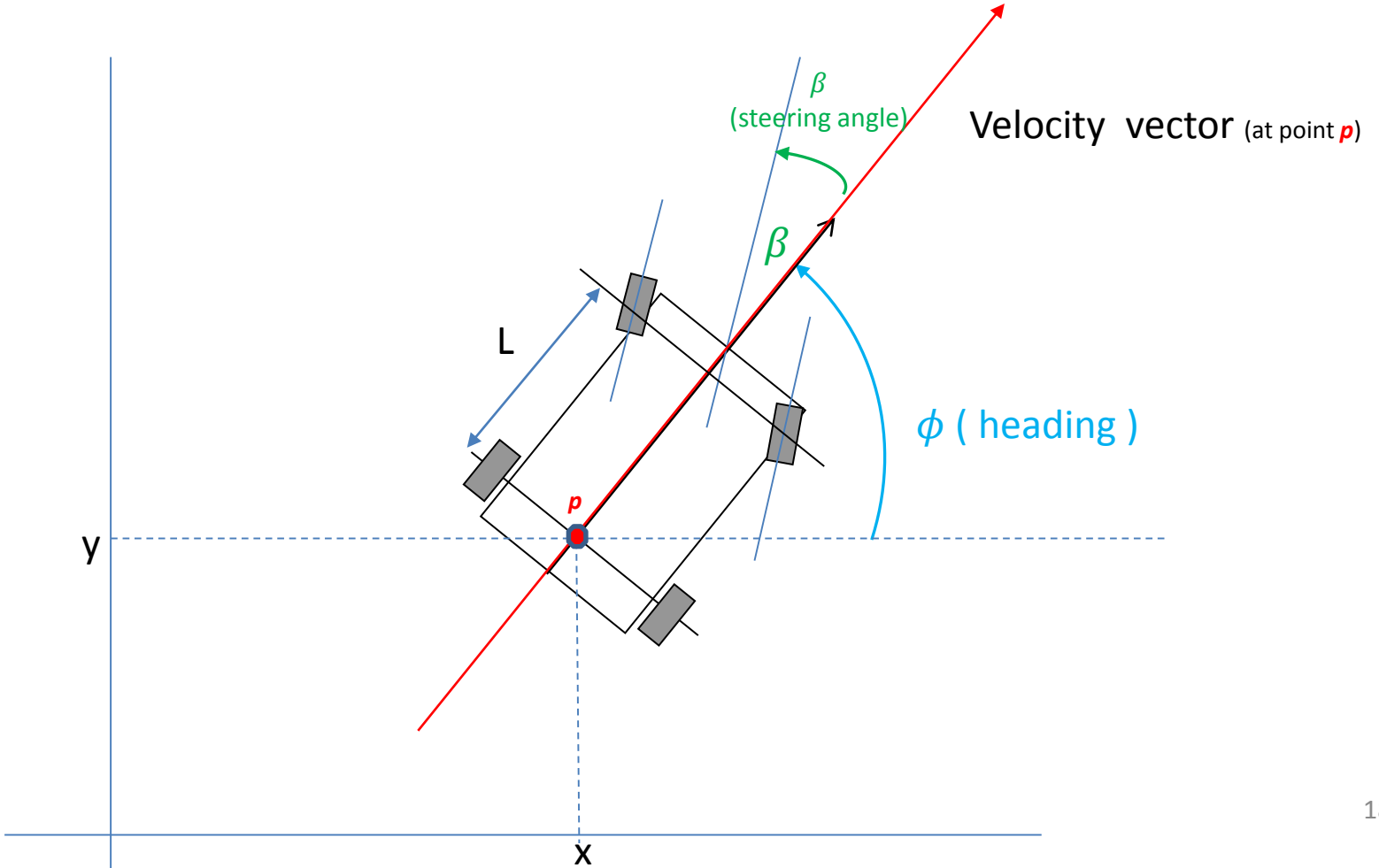
Simplified kinematic model of wheeled platform, in 2D

This model is MIMO and nonlinear
We focus on that matter now.

(How this model is obtained is discussed next week, when we talk about kinematic models.)

Kinematic model of a car

Here, we model the position of certain point of the car, in certain coordinate frame, as function of the steering angle and the instantaneous longitudinal velocity (“speed”)
The model has a 3D state vector (2D position + heading)



Suppose we can operate the values of the steering angle and the longitudinal velocity of the platform (a human driver),

$$\frac{d\mathbf{x}(t)}{dt} = f\left(\mathbf{x}(t), \mathbf{u}(t)\right)$$

$$\Downarrow$$

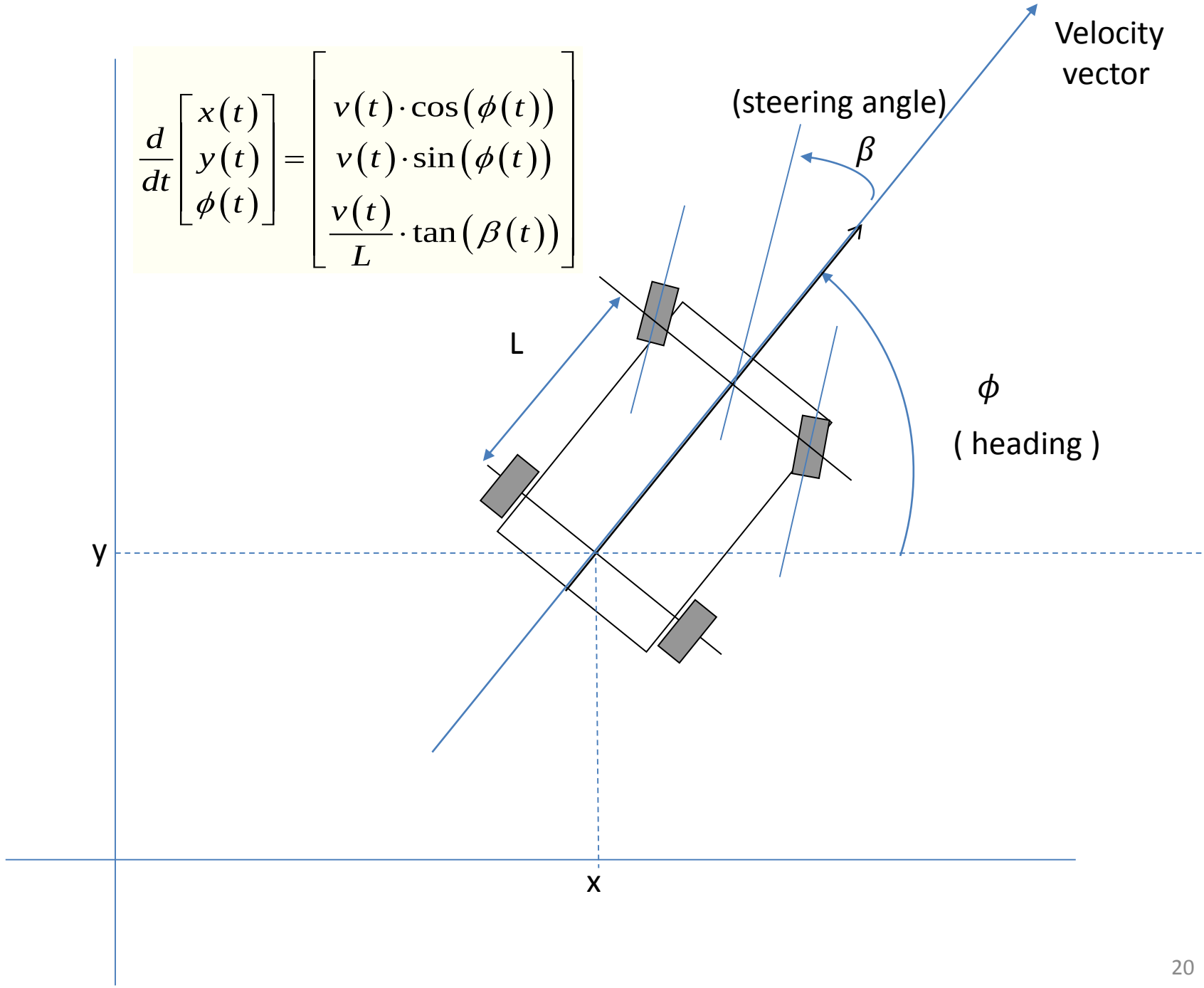
$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \phi(t) \end{bmatrix}, \quad \mathbf{u}(t) = \begin{bmatrix} v(t) \\ \beta(t) \end{bmatrix}$$

$$\frac{d\mathbf{x}(t)}{dt} = \frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \phi(t) \end{bmatrix} = \begin{bmatrix} v(t)\cdot\cos\left(\phi(t)\right) \\ v(t)\cdot\sin\left(\phi(t)\right) \\ \frac{v(t)}{L}\cdot\tan\left(\beta(t)\right) \end{bmatrix}$$

This model is MIMO, in its “native” version.

(In addition, we can see that it is nonlinear.)

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \phi(t) \end{bmatrix} = \begin{bmatrix} v(t)\cdot\cos(\phi(t)) \\ v(t)\cdot\sin(\phi(t)) \\ \frac{v(t)}{L}\cdot\tan(\beta(t)) \end{bmatrix}$$



Discrete time approximation

(of Continuous time models)

You may have seen, for LTI systems, an approach for obtaining a discrete time version of the continuous time model; in MTRN3020.

(no worries if you have not done that course yet!, we cannot (usually) use that here)

Discrete time approximation

One advantage of maintaining the non-linear state space representation, is that we can also approximate those nonlinear continuous time models, to be used in a discrete time fashion, in an almost trivial way.

If we consider a “small” enough “sampling time” for applying the Euler approximation (which is a first-order linear approximation, in time) , we obtain a nonlinear discrete time version of the nonlinear continuous time model.

Discrete time solution, approximated by “Euler method”.

Usually, for nonlinear cases, we do not have a close form expression, consequently we approximate it, for short periods of time.

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right)$$

$$\Downarrow$$

$$\mathbf{x}(t+\Delta t) \cong \mathbf{x}(t) + \Delta t \cdot \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right)$$

By applying this approximation, in a recursive fashion, we can obtain an approximate solution in time, for a set of consecutive times.

(now, we see some simulations, applying this approach, in Matlab.)

Kinematic model of a car (discrete time version)

$$\mathbf{x}(k+1) = \mathbf{f}\left(\mathbf{x}(k), \mathbf{u}(k)\right)$$

$$\Downarrow$$

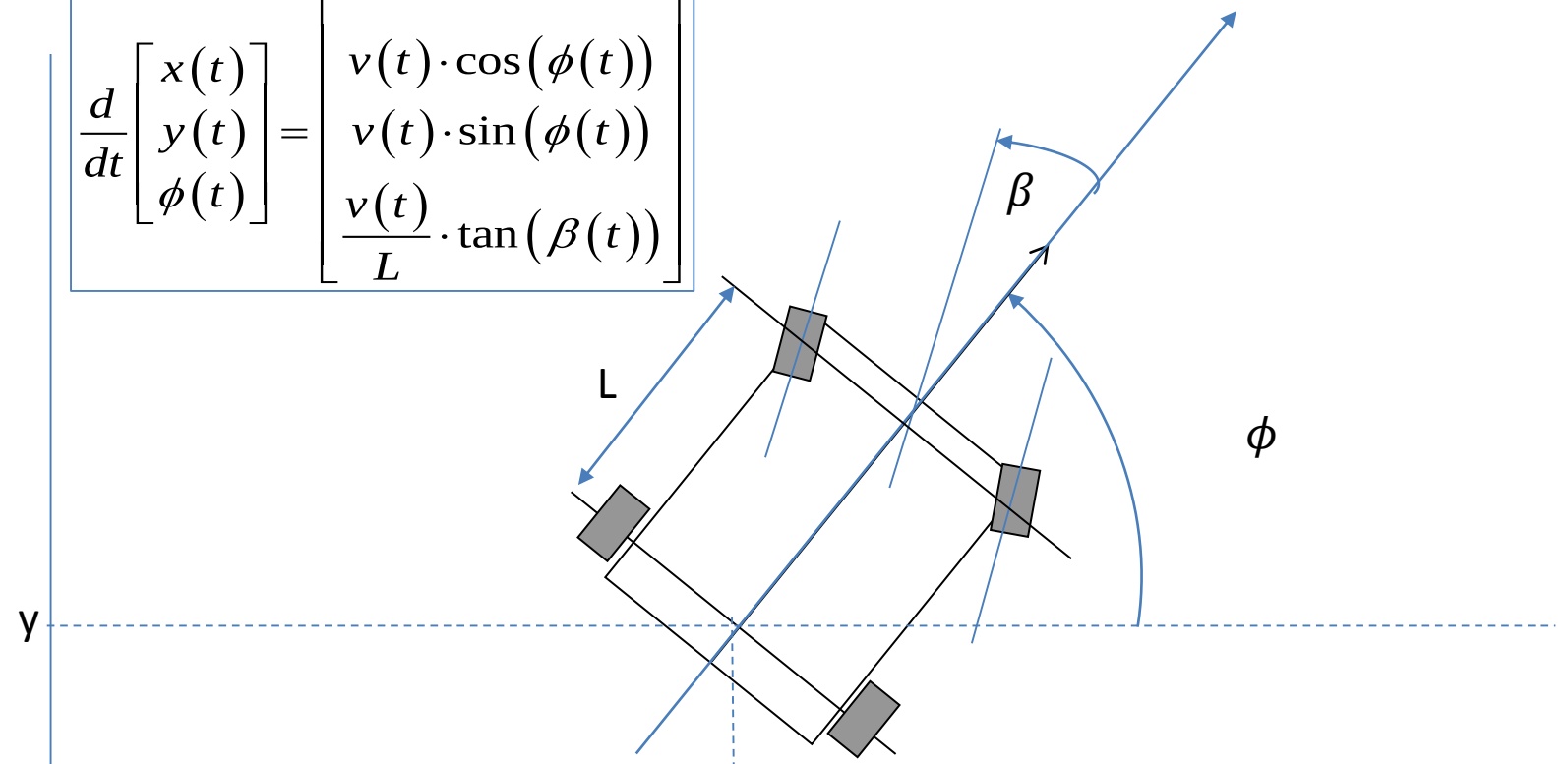
$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} + T \cdot \begin{bmatrix} v(k) \cdot \cos\left(\phi(k)\right) \\ v(k) \cdot \sin\left(\phi(k)\right) \\ \dfrac{v(k)}{L} \cdot \tan\left(\beta(k)\right) \end{bmatrix}$$

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix}, \quad \mathbf{u}(k) = \begin{bmatrix} v(k) \\ \beta(k) \end{bmatrix}$$

In which the index ***k*** implicitly means discrete time $t = k \cdot T$. The input values $v(k), \beta(k)$ are usually sampled by sensors which measure those variables; or they can be actual values applied to the system's actuators, by a driver or a controller, so that we know these values. In addition, we assume that the sampling time, **T**, is short enough, so that the Euler approximation is sufficiently good.

(There are cases in which the model is native in a discrete time context, and no concept of continuous time is associated to it)

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \phi(t) \end{bmatrix} = \begin{bmatrix} v(t)\cdot\cos(\phi(t)) \\ v(t)\cdot\sin(\phi(t)) \\ \frac{v(t)}{L}\cdot\tan(\beta(t)) \end{bmatrix}$$



$$\begin{bmatrix} x((k+1)\cdot\tau) \\ y((k+1)\cdot\tau) \\ \phi((k+1)\cdot\tau) \end{bmatrix} = \begin{bmatrix} x(k\cdot\tau+\tau) \\ y(k\cdot\tau+\tau) \\ \phi(k\cdot\tau+\tau) \end{bmatrix} = \begin{bmatrix} x(k\cdot\tau) \\ y(k\cdot\tau) \\ \phi(k\cdot\tau) \end{bmatrix} + \tau\cdot\begin{bmatrix} v(k\cdot\tau)\cdot\cos(\phi(k\cdot\tau)) \\ v(k\cdot\tau)\cdot\sin(\phi(k\cdot\tau)) \\ \frac{v(k\cdot\tau)}{L}\cdot\tan(\beta(k\cdot\tau)) \end{bmatrix}$$

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} + T \cdot \begin{bmatrix} v(k) \cdot \cos(\phi(k)) \\ v(k) \cdot \sin(\phi(k)) \\ \frac{v(k)}{L} \cdot \tan(\beta(k)) \end{bmatrix}$$

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix}, \quad \mathbf{u}(k) = \begin{bmatrix} v(k) \\ \beta(k) \end{bmatrix}$$

Example: Simple case, written in Matlab code.
We see the model used in a simulation (M file, in Matlab)
We change certain parameter to appreciate the resulting different behaviors.

(We end here reviewing this topic)

COORDINATE FRAMES

Vectors are expressed in coordinate frames. A vector may represent the state of a system, or a simple position in 2D or 3D.

We can “change variables” , by changing the coordinate frame in which we represent that vector.

That is what was discussed about “similarity transforms”, in MMAN3200. In that case we were talking about changing the coordinate frame for expressing the internal state of the system, which resulted in different state and output equations, but always preserved the input/output relationship of the system.
(however, that is not our topic here.)

Here, we are simply talking about expressing positions and orientations, from different perspectives (i.e. using different reference coordinate frames)

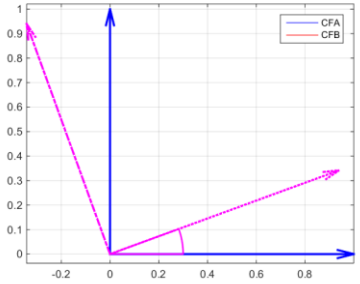
We have a coordinate, frame (CF) in which we express points (vectors), as linear combinations of the set of orthogonal unity vectors which defines that coordinate frame (its basis). We call that CF, "CFA" (coordinate frame A).

Now we propose a new CF, CFB, which is still centered at the origin of CFA, but whose basis is defined by a different set of n orthogonal unity vectors, $\left\{ \mathbf{b}_i \right\}_{i=1}^{n}$, which, if expressed in CFA , are $\left\{ \mathbf{b}_i^a \right\}_{i=1}^{n}$

For expressing a given point **p** in CFB, we simply need to perform the inner product of **p** against each of the vectors in the basis of CFB, and then obtain a vector of those obtained coefficients. We may do it in CFA (or any other CF). So, if we have **p** expressed on CFA , then we do this

$$\mathbf{p}^a \quad \Rightarrow \quad \mathbf{p}^b = \mathbf{R}_b^a \cdot \mathbf{p}^a$$

$$\mathbf{R}_b^a = \left[\mathbf{b}_1^a \quad \mathbf{b}_2^a \quad \dots \quad \mathbf{b}_n^a \right]^T$$



(this matrix multiplication performs these needed n inner products, and generates a column vector with their results, i.e., the vector $\mathbf{p}^b$)

(inner product of column vectors $c = \langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^T \cdot \mathbf{w} = \mathbf{w}^T \cdot \mathbf{v}$)

We can apply similar procedure for converting points expressed in CFB to being expressed in CFA.

We know from MATH, that because our basis are composed by orthogonal unity vectors, that certain simplifications are valid.

$$\mathbf{p}^a \quad \Rightarrow \quad \mathbf{p}^b = \mathbf{R}_b^a \cdot \mathbf{p}^a$$

$$\mathbf{R}_b^a = \begin{bmatrix} \mathbf{b}_1^a & \mathbf{b}_2^a & \dots & \mathbf{b}_n^a \end{bmatrix}^T$$

$$\mathbf{p}^b = \mathbf{R}_b^a \cdot \mathbf{p}^a$$

$$\Downarrow$$

$$\mathbf{p}^a = \mathbf{R}_a^b \cdot \mathbf{p}^b = \left(\mathbf{R}_b^a\right)^{-1} \cdot \mathbf{p}^b = \left(\mathbf{R}_b^a\right)^T \cdot \mathbf{p}^b$$

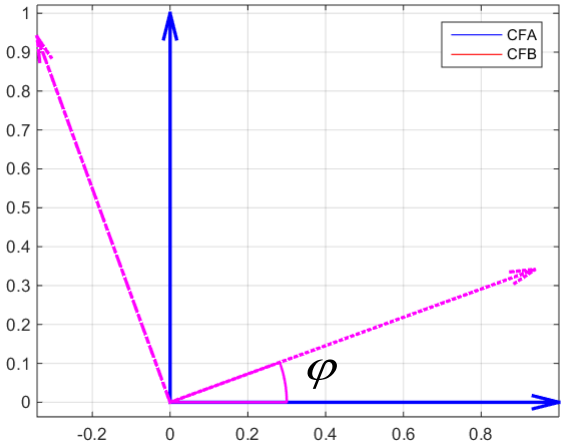
Now we see it in a 2D case.
We have from before: $$\mathbf{p}^a = \left(\mathbf{R}_b^a\right)^T \cdot \mathbf{p}^b$$

We express the basis of CFB (the two magenta vectors) in CFA.

$$\mathbf{b}_1^a = \begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix};\quad \mathbf{b}_2^a = \begin{bmatrix}\cos(\varphi+\pi/2)\\ \sin(\varphi+\pi/2)\end{bmatrix} = \begin{bmatrix}-\sin(\varphi)\\ \cos(\varphi)\end{bmatrix}$$

$$\mathbf{R}_a^b = \left(\mathbf{R}_b^a\right)^T = \begin{bmatrix}\mathbf{b}_1^a & \mathbf{b}_2^a\end{bmatrix} = \left[\begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix}\begin{bmatrix}-\sin(\varphi)\\ \cos(\varphi)\end{bmatrix}\right]$$

$$= \begin{bmatrix}\cos(\varphi) & -\sin(\varphi)\\ \sin(\varphi) & \cos(\varphi)\end{bmatrix} = \mathbf{R}_\varphi$$

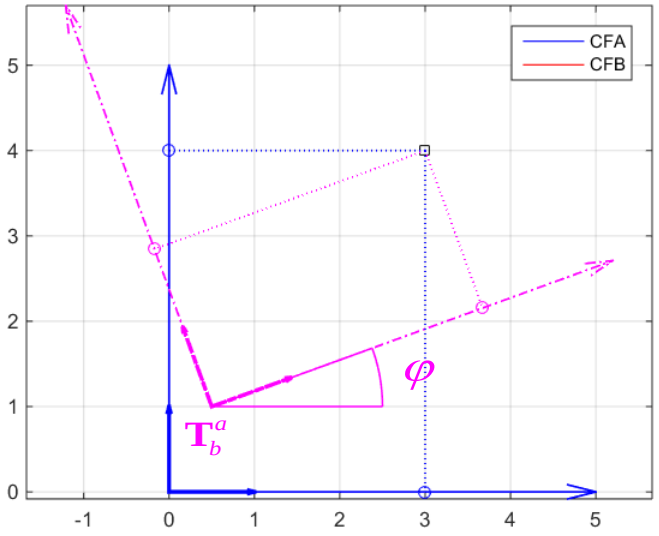


In which we defined $\mathbf{R}_\varphi$, which is known as "rotation matrix" (for a CCW angle φ , in this case.)

$$\mathbf{p}^a = \mathbf{R}_\varphi \cdot \mathbf{p}^b$$

For the inverse process: $$\mathbf{p}^b = \mathbf{R}_{-\varphi} \cdot \mathbf{p}^a \qquad \left(\mathbf{R}_\varphi^{-1} = \mathbf{R}_\varphi^T = \mathbf{R}_{-\varphi}\right)$$

What would happen if we also have a translation?



A point p, seen from CFB can be expressed in CFA as follows:

$$\mathbf{p}^a = \mathbf{R}_\varphi \cdot \mathbf{p}^b + \mathbf{T}_b^a$$

So that the point "seen" from CFB (expressed in CFB) is rotated by the angle $\boldsymbol{\varphi}$, and then translated (shifted) by $\mathbf{T}_b^a$

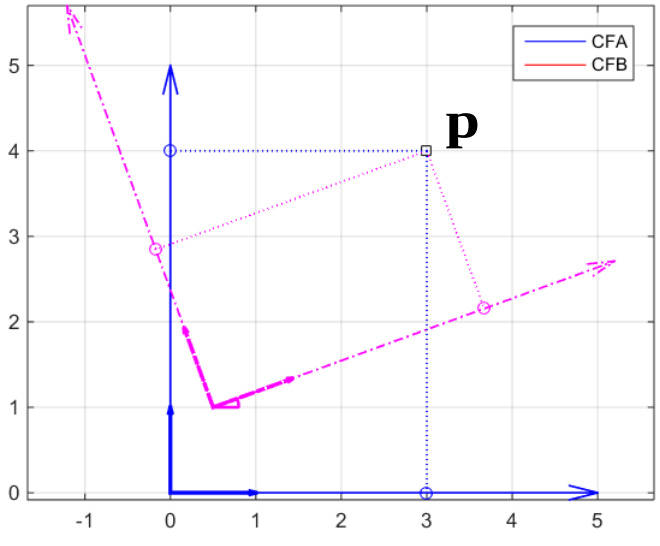
($\mathbf{T}_b^a$ is the origin of CFB expressed in CFA.)

And, also, from CFA and then expressed in CFB

$$\mathbf{p}^a = \mathbf{R}_\varphi \cdot \mathbf{p}^b + \mathbf{T}_b^a$$

$$\Downarrow$$

$$\mathbf{p}^b = \mathbf{R}_\varphi^{-1} \cdot \left(\mathbf{p}^a - \mathbf{T}_b^a \right)$$

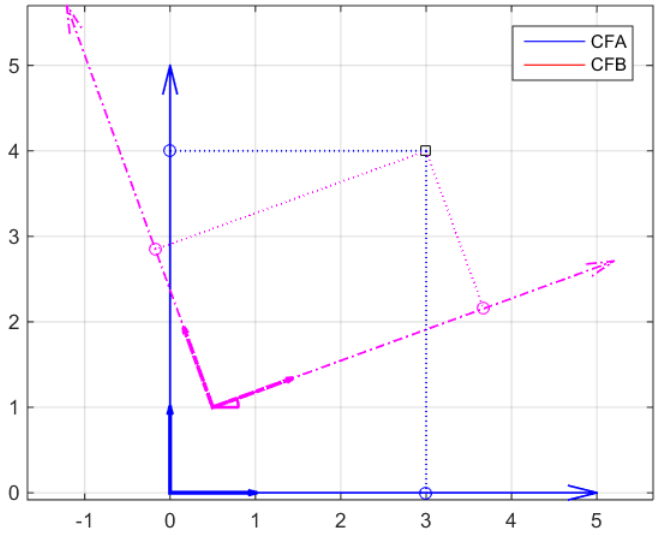


Because $\mathbf{R}_\varphi$ is a 2D rotation matrix , it satisfies that:

$$\mathbf{R}_\varphi^{-1} = \mathbf{R}_\varphi^T = \mathbf{R}_{-\varphi}$$

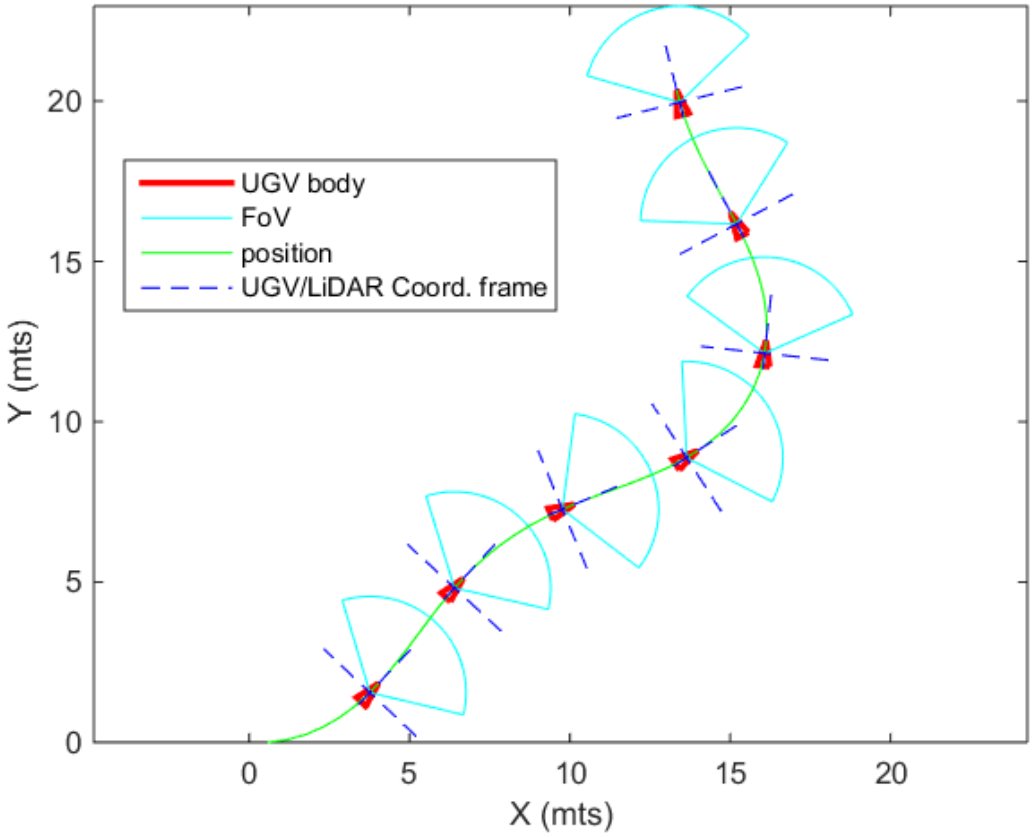
(in 3D and higher dimensional cases, the simple change of sign in the angle is not valid), only the transpose rule Is valid)
(in those cases, the rotation will be defined by a set of angles, and some additional considerations will be necessary)

Why are “Translation and Rotation”: relevant to us?



Because “CFB” may be, in some cases,
the coordinate frame of a sensor (LiDAR or camera or radar), mounted on a platform.
Measurements provided by that sensor will be expressed (“natively”) in its local coordinate frames.

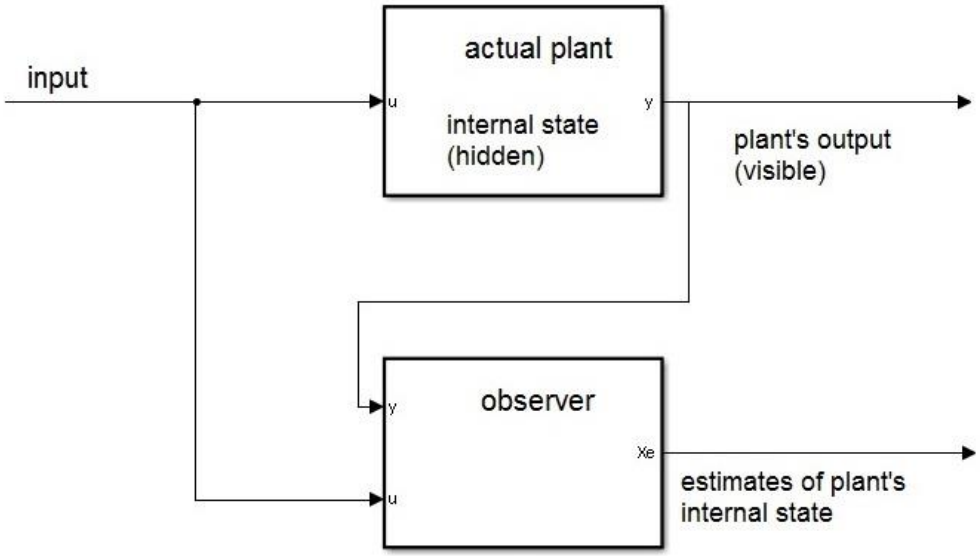
Why are “Translation and Rotation”: relevant to us?



(This topic ends here.)

OBSERVERS

(if we have time, we include it in our review)



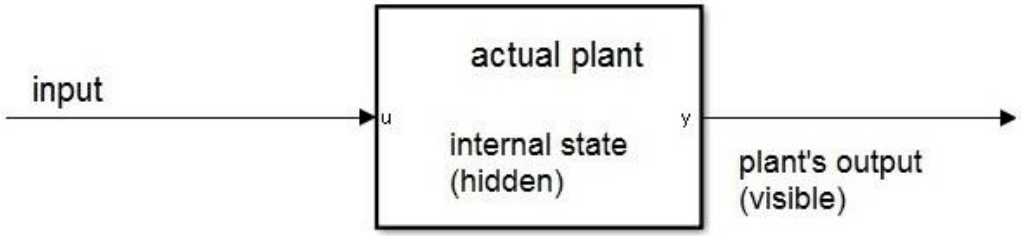
OBSERVERS

We have this situation:

We need to know the current internal state of a system.
i.e. ,the values of **X**(t) at time t (now)

But **X**(t) is hidden to us.

We have access to the plant's outputs, **y**(t), always, because we measure those, via sensors.
We also know the plant' inputs, **u**(t) (because we apply those, or because we measure them)

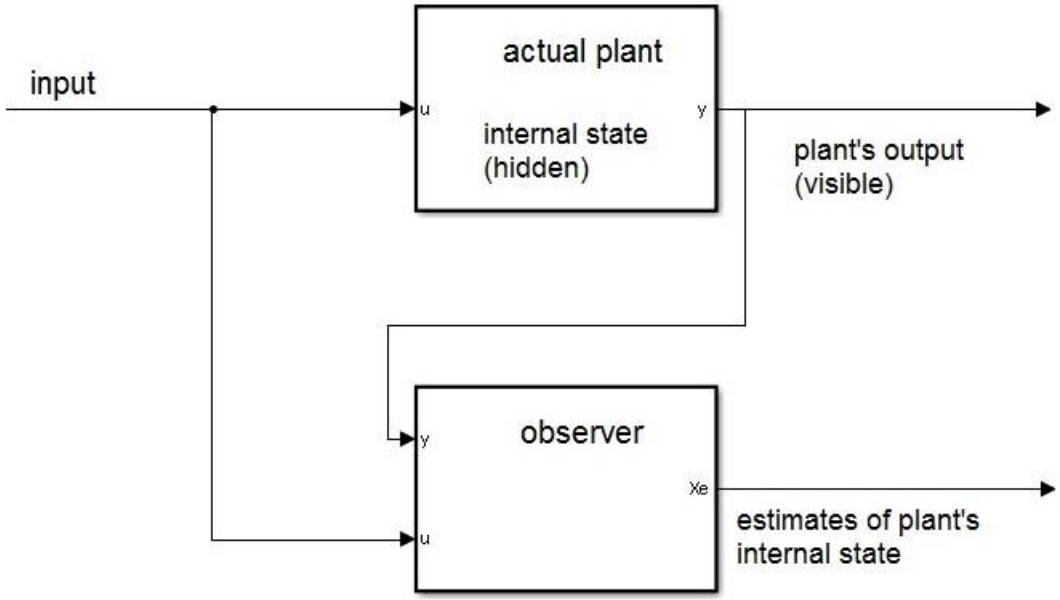


Can we estimate **X**(t)? Continuously, so it is available for Control or other purposes?

OBSERVERS

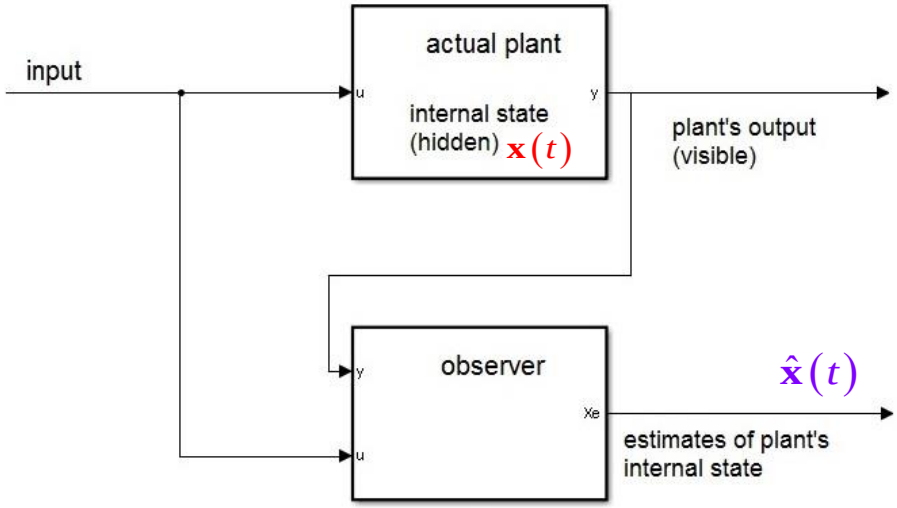
Can we estimate X(t)? Continuously, so it is available?

Answer: **usually yes** (it depends on certain conditions),
by using an "**observer"**



OBSERVERS

In MMAN3200 we saw a classic observer, for LTI systems, in continuous time domain.
Its design and implementation were based on the parameters of the “nominal” model, i.e., the matrixes A,B,C,D. And we also needed to specify the dynamics of the (expected) convergence.



Observer:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}\cdot\hat{\mathbf{x}}(t) + \mathbf{B}\cdot\mathbf{u}(t) + \text{correction}(t)$$

$$\hat{\mathbf{y}}(t) = \mathbf{C}\cdot\hat{\mathbf{x}}(t) + \mathbf{D}\cdot\mathbf{u}(t)$$

$$\text{correction}(t) = \mathbf{L}\cdot\left(\mathbf{y}^{*}(t) - \hat{\mathbf{y}}(t)\right)$$

$\hat{\mathbf{x}}(t)$: estimated state

$\mathbf{y}^{*}(t)$: measured output

$\mathbf{L}$: observer gain

$\mathbf{L} = ?$ Our design provided the parameter L.

(L is function of A,C and the desired eigenvalues of error's dynamics.)

The observer module, which operates in real-time, is usually implemented in software, in a controller computer

(a discrete time version of it, can even be implemented in an Arduino, for certain feasible cases)

Observer:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}\cdot\hat{\mathbf{x}}(t) + \mathbf{B}\cdot\mathbf{u}(t) + \text{correction}(t)$$

$$\hat{\mathbf{y}}(t) = \mathbf{C}\cdot\hat{\mathbf{x}}(t) + \mathbf{D}\cdot\mathbf{u}(t)$$

$$\text{correction}(t) = \mathbf{L}\cdot\left(\mathbf{y}^{*}(t) - \hat{\mathbf{y}}(t)\right)$$

$$\mathbf{L} = ?$$

The design of L is based on obtaining $\mathbf{L}$ such as the eigenvalues of the matrix $(\mathbf{A} - \mathbf{L}\cdot\mathbf{C})$ are the desired ones .

Those eigenvalues describe the dynamics of the error , $\mathbf{e}(t) = \hat{\mathbf{x}}(t) - \mathbf{x}(t)$, which is, nominally,

$$\dot{\mathbf{e}}(t) = (\mathbf{A} - \mathbf{L}\cdot\mathbf{C})\cdot\mathbf{e}(t)$$

There are cases for which there is no solution (no matrix L would satisfy our requirements).

For having solution, the system must be "observable"

(Observability, Observability matrix, etc, as seen in MMAN3200)

However, that classic observer missed some relevant matters/issues which usually happen in real life:

1) Our nominal model may not be LTI (very usual)
2) The actual plant's model may not be well known (so we have a "nominal" one.)
3) The measurements of the plant's outputs are usually polluted with noise.
4) The measurements of the outputs may be not continuously available (e.g. sporadic sensor unavailability).
5) The measurements of the outputs may be asynchronous, or multi-rate.
6) Our knowledge about the input values may be polluted with uncertainty/noise.
7) We may have access to other sources of information, which are sporadically available (and which are worth to be used)
8) We may be required to produce, in addition to the values of the estimates, some details bout the quality of those (confidence)

We will not treat these issues now, but we will present a powerful approach later during the term.

Review completed.

We have some tutorial problems using some of the reviewed concepts.

If you have doubts, this week would be the time to clarify matters.