Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Kétszabadságfokú robusztus szabályozó tervezése és validálása egzakt linearizálható mechatronikai rendszerhez

Design and validation of a two-degrees-of-freedom controller for mechatronic systems

Tudományos Diákköri Konferencia dolgozat

Szerző: Finta Barnabás

Konzulens: Dr. Kiss Bálint

2020.10.28.

Acknowledgement

The research was supported by the EFOP-3.6.2-16-2017-00013 project – financed by the Ministry of Human Capacities of Hungary.

I would like to thank my report supervisor Dr. Bálint Kiss for his valuable time, helpful advice and unceasing confidence in me during the writing of this report. I am also thankful for the opportunity given to me to visit the department's laboratory to implement the work presented below, despite these trying times caused by the pandemic.

Table of contents

Abstract

Exact (feedback) linearization is one of the most effective methods to control nonlinear systems. It consists of a static or eventually dynamic state feedback and a coordinate transformation such that the resulting closed-loop dynamics is linear and hence suitable to apply classical, linear design methods in control theory such as pole placement.
However, exact linearization requires the knowledge of the mathematical model of the system in its entire operating range and the value of the model parameters with high precision which may change during operation. Without this knowledge, neither the performance nor the stability of the closed-loop dynamics is guaranteed. Moreover, the controlled physical systems are subject to external disturbances (e.g. friction and load variation in mechanical systems) and noisy measurements. We study a cascade control architecture and a design method such that the outer controller ensures robustness of the exactly linearized dynamics against parameter uncertainty.
To the extent necessary for the presentation of the design method, the report summarizes first some elements of the theory of nonlinear and robust control and introduces all notations used in the sequel. This is followed by the presentation of the proposed controller architecture and the design workflow. The method allows the design of both serial and two-degrees-of-freedom (2DOF) robustifying controllers, the latter being also suitable to improve disturbance rejection.
The applicability of the method is demonstrated for mechanical systems. Simulations show performance and stability robustness, and thanks to the real-time implementation of the controller for a 2DOF robotic arm in the laboratory of the IIT department, the advantages of the presented methods are also validated by experiments on a real physical system, where the uncertain parameters were chosen considering payloads with various mass, the rest of the parameter values were identified using measurement data.

1 Introduction

In many cases, the control of nonlinear systems uses linearization around an operating point and synthetizing a controller for the arising linear model, approximating the nonlinear behavior in a neighborhood of the operating point (using on the Hartman-Grobman theorem). This approach is simple and works well in the proximity of the operating point [1], however the tracking of a complex reference signal is usually problematic in this case. To solve this problem, other control techniques for nonlinear systems emerged. It was shown that for some classes of nonlinear systems, there exists a state feedback and nonlinear change of coordinates which if applied to the plant result in a closed-loop with linear dynamics.

This approach provides great benefits as the transformed system is linear in the entire operating range (as opposed to the neighborhood of an operating point), however the knowledge of the adequately precise mathematical model of the controlled system is needed. This means the knowledge of the plant parameters with sufficient precision. These parameters can change during operation, for example because of the degradation of mechanical parts, changes in friction, by handling a different payload etc. Also, considering mechanical systems, the presence of friction usually makes matters worse, the modelling of such effects is cumbersome, where even more parameters (now to also model friction) are arising, which are also hard to identify. This is the reason why, with the exception of linear friction, these effects are omitted in a typical control synthesis process.

These considerations mean that by applying the state feedbacks and coordinate transformations using the nominal parameter values for a nonlinear plant with different, uncertain parameters, the resulting closed loop system may become quasi-linear or it is a different linear system than the nominal one. This results in decreased reference tracking performance, and the system may also become unstable. This thesis is concerned with a method to robustly control such nonlinear systems, despite these parameter uncertainties.

The second chapter summarizes the necessary theoretical background. If the reader is familiar with the concepts of nonlinear control and robust control theory this chapter may be skipped. Naturally given the limits of the thesis, this summary only scratches the surface of the vast and exciting theory of nonlinear and robust control, this means that for example the proofs are omitted. However, it hopefully condenses this information to the minimal necessary level, which makes the examples and overall concept of the control architecture clear.

The third chapter presents the proposed controller solution, which merges the two state-of-the-art methods of robust and nonlinear control. The linearizing and tracking feedbacks are applied to the plant and the cascaded outer loop containing the robust controller ensures robust stability and performance. After looking at the general concept of synthetizing such a controller, several examples are presented to illustrate the effectiveness of the method. All of them are validated by simulations, and for one particular example, for an identified two-degrees-of-freedom robotic arm the controller is also implemented on a real physical system. The results show the effectiveness of the method for these mechatronic systems.

In the next chapter, let us review first the theoretical concepts used in the sequel.

2 Theoretical Background

This chapter overviews the concept necessary to introduce the proposed control architecture. The reader who is already familiar may only need to check the notations introduced.

Let us start with the nonlinear control theory, where the majority of the concepts and examples are from the books [2] and [3] and also from the course material [4]. The field of controlling nonlinear systems (including mechanical systems such as robots, cranes, ground and aerial vehicles), is often referred to as geometric control. The name refers to the fact that the configuration manifold (and the state space) of mechanical systems containing rotational degrees of freedom are given by a "curvy" space as opposed to a "flat" (Euclidean) space. Take for example a simple pendulum. Its configuration can be exactly described with one variable, the angle of the pendulum's rod in relation to a default (in this case a horizontal) position. It is clear that this variable we use to describe the configuration with is evolving on a circle S^1 and not on the real line $\mathbb{R}^1$, e.g. after one full rotation the value 2π radians equals 0 radians from the point of view of the system.

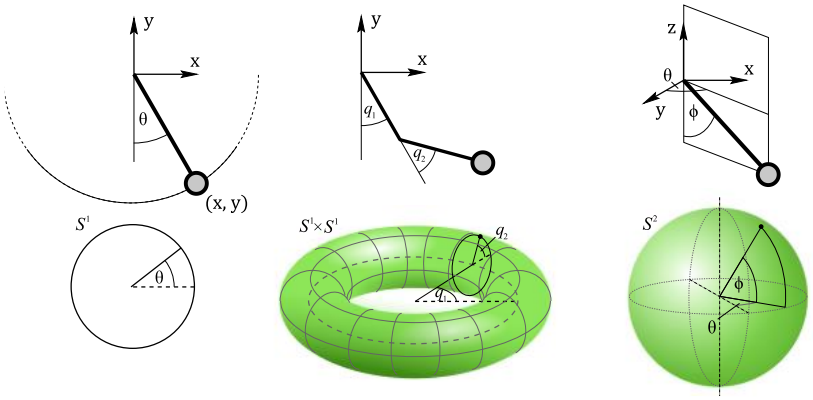

Fig 1. Configuration spaces of a pendulum, double pendulum and spherical pendulum

This fact motivated the concepts of geometric control to which the field of differential geometry gave the mathematical background. The next subsection gives a brief overview of the language of differential geometry for better understanding of the underlying concepts of geometric control.

2.1 Differential Geometry

It is mentioned earlier that mechanical systems with rotational degrees of freedom are evolving on curvy spaces, that is, their configuration space is a curved and not a flat one. The main purpose of differential geometry is to equip these configuration sets with a differentiable structures, locally equivalent to vector spaces, called manifolds.

2.1.1 Manifolds and diffeomorphisms

Manifold: a n-dimensional manifold is a space that is everywhere locally diffeomorphic to $\mathbb{R}^n$.

So anywhere we are on a manifold, the point p we stand at and its neighborhood U is diffeomorphic to the n-dimensional flat space or $\mathbb{R}^n$.

Diffeomorphism: A diffeomorphism is a C^r-differentiable (smooth), one-to-one and onto (bijective) relation with a smooth inverse.

$$\varphi(p) = x, \quad p \in U \subset M^n, \quad x^T = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \in \mathbb{R}^n \tag{2.1}$$

A diffeomorphism is thus a smooth, invertible mapping between two sets. So a manifold looks like a locally flat space, i.e. if we "zoom in" enough (which means that the flat space itself is also a manifold). Given a diffeomorphism ϕ that maps an open subset U of the manifold M to the flat space, the pair (ϕ,U) is a chart. This means that we may need multiple charts to cover the whole manifold if it is curved.

Atlas: Given a manifold M there exist a collection of charts $\{(\phi,U)\}$ that cover the whole manifold. This collection is called an atlas.

There can be overlaps between the open sets that cover the manifold, however they must be compatible. Given two neighborhoods, U_i and U_j, that are not disjoint and a point p in their intersection, the charts $\{(\phi_i,U_i)\}$ and $\{(\phi_j,U_j)\}$ are compatible if by applying the diffeomorphism ϕ_j^{-1} that takes a point in $\mathbb{R}^n$ and maps it to p, then applying ϕ_i to the result, we get a diffeomorphism that maps a point from $\mathbb{R}^n$ to $\mathbb{R}^n$.

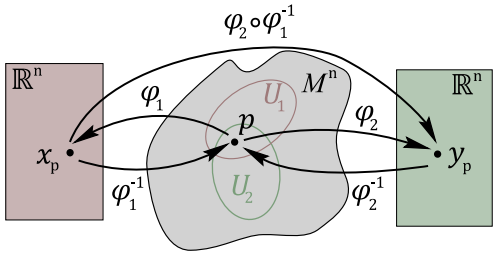

Fig 2. Compatibility of two charts covering the same point of an n-dimensional manifold

We concluded that a manifold locally looks like a flat space where differentiation is possible. An example of manifolds, widely used in mechanics is the set of 3D orientation matrices. The collection of all these matrices give us the group of special orthogonal 3x3 matrices, or SO(3). It can be proved that SO(3) is indeed a manifold, where each point on the manifold is a rotation matrix.

To summarize the subject of manifolds we give the formal definition:

M is an n-dimensional C^r-differentiable manifold if:

1. For $\forall p \in M$ there exists a neighborhood U of p in M and a diffeomorphism $\varphi: U \to \mathbb{R}^n$, where the pair (φ,U) is called a local coordinate chart.
2. $\bigcup_i U_i = M$, the collection $\{(\varphi_i,U_i)\}$ is called an atlas.
3. Compatibility: Given $U_i \cap U_j \neq \varnothing$, then $\varphi_j \circ \varphi_i^{-1}: \mathbb{R}^n \to \mathbb{R}^n$ must be a C^r-diffeomorphism.

We can define differentiable maps between manifolds, even if they have a different structure and dimension.

2.1.2 Curves and Tangent spaces

We can define curves on manifolds. These are functions that take a scalar variable (e.g. time) and give us a point on the manifold. If we consider the set where we get from our scalar variable, the set of real numbers as a manifold, then we can say that a curve is a mapping between two manifolds. First let us look at a function that takes a point on the manifold and maps it to the real line.

To define differentiability of functions acting on manifolds we need to somehow transform them into mappings from a vector space to another vector space, where differentiability can easily be defined.

A function $f:M\to\mathbb{R}$ is differentiable for all local charts (φ,U) if the mapping $f\circ\varphi^{-1}:\mathbb{R}^n\to\mathbb{R}$ is differentiable. We say that $f\circ\varphi^{-1}$ is the representation of f in local coordinates, denoted by $f(x_1 \quad x_2 \quad \cdots \quad x_n)$.

If we take an *m*-dimensional manifold *M*, and an *n*-dimensional manifold *N*, and have a mapping *F* from *M* to *N*, then $F:M\to N$ is differentiable if $\psi\circ F\circ\varphi^{-1}:\mathbb{R}^m\to\mathbb{R}^n$ is differentiable. Alternatively, $F:M\to N$ is differentiable if for every differentiable function *g* on *N*, the mapping $g\circ F:M\to\mathbb{R}$ is differentiable.

A curve γ on a manifold M is a map $\gamma:[0,\ t]\to M,\quad [0,\ t]=I\subset\mathbb{R}$. It is a differentiable curve at point *p*, if the mapping $\varphi\circ\gamma:I\subset\mathbb{R}\to\mathbb{R}^n$ at *p* is differentiable.

These curves can be thought of as trajectories that are functions of time, and velocities can be defined. The vector representing the velocity is tangent to the trajectory.

Tangent space at p. Consider all possible differentiable curves on a manifold *M*, which go through the point $p\in M$ at time 0. The space spanned by all possible velocity vectors is the tangent space at the point *p*, denoted by T_pM.

If we think about manifolds as surfaces, then it is easy to check if a given vector at a point is tangent to the surface. Let us assume that the surface is defined by a function $f(x)=const.$ and we are given a possible tangent vector *v*. If the dot product of the surface normal at point *p* and vector *v* is zero, then vector *v* is tangent to the given surface at point *p*, hence

$$\nabla f(x)|_p\cdot v=0 \tag{2.2}$$

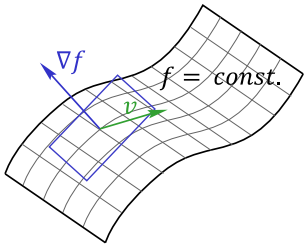

Fig. 3. The connection of the tangent plane and surface normal

Tangent bundle. The tangent bundle of a manifold *M* is denoted by *TM* and given by $\bigcup_{p\in M}T_pM=\{(p,v)\mid p\in M, v\in T_pM\}$.

Tangent space is applicable to a certain point of the manifold. The tangent bundle is the collection of all the tangent spaces at each point of the manifold.

Let us take for example a unit sphere as our manifold embedded in 3-dimensional Euclidean space (so we use the classical *x*, *y* and *z* coordinates in this example) and a curve $\gamma(t)=\left[t \quad t/2 \quad \sqrt{1-5t^2/4}\right]^T$ on S^2 at a point $p=[0 \quad 0 \quad 1]^T$. It is trivial that at this point the tangent space contains vectors in the form $[a \quad b \quad 0],\ a,b\in\mathbb{R}$. We also define a diffeomorphism $\varphi(x,y,z)=\left[\frac{2x}{1-y} \quad \frac{2z}{1-y}\right]^T$ which is the so called stereographic projection with the corresponding open set

$U = \{(x,y,z) \in \mathbb{R}^3 \mid f(x,y,z) = x^2 + y^2 + z^2 = 1, y < 1\}$. The stereographic projection takes a point in U and projects a line from E through the given point in U, such that it maps it to the x-z plane. We want to compute the tangent vector of the curve γ at time 0, which is denoted by $\gamma'(0)$. We show two different methods: the more classical approach, where we take advantage of the fact that the manifold is embedded in a higher dimensional space; and the differential geometric approach, where we use the given diffeomorphism.

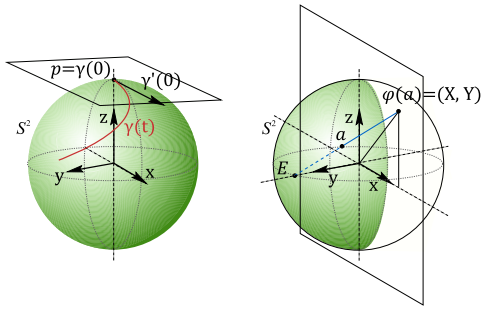

Fig. 4. Curve and tangent vector on a sphere, stereographic projection to the x-z plane

With the classical approach we simply differentiate with respect to t:

$$\gamma'(0) = \begin{bmatrix} 1 & 0.5 & 0 \end{bmatrix}^T \in T_p S^2 \tag{2.3}$$

With the differential geometric approach, the differentiation is carried out in local coordinates, meaning that we stereographically project the curve and then differentiate.

$$\frac{d}{dt}(\varphi \circ \gamma(t)) = \frac{d}{dt}\begin{bmatrix} \frac{2t}{1-t/2} & \frac{\sqrt{1-5t^2/4}}{1-t/2} \end{bmatrix}^T \rightarrow \frac{d}{dt}(\varphi \circ \gamma(0)) = \begin{bmatrix} 2 & 1 \end{bmatrix}^T \in T_p S^2 \tag{2.4}$$

We conclude that the two results are equivalent, they point to the same direction. This demonstrates one of the main features of differential geometry, namely that the calculus is independent of the choice of the coordinate system.

Let us look at a function $f : M \rightarrow \mathbb{R}$, $f \in C^\infty$, where M is a manifold and a vector v that acts on that function. For example the manifold is the surface of the ocean (it is $\mathbb{R}^2$), and f gives us the temperature at any given point on the surface, we assume that at every point the temperature is constant. Vector v is the velocity of a particle. Then v acting on f expresses the relative temperature change the particle experiences as it travels with the velocity specified by v.

$$v(f) = v \cdot f = v_1 \frac{\partial f}{\partial x_1} + \cdots + v_n \frac{\partial f}{\partial x_n} = \nabla f \cdot v \tag{2.5}$$

This equation is known to result the directional derivative of f along v, $v \in T_p M$, $v : C^\infty(M) \rightarrow \mathbb{R}$.

We saw that a vector acting on a scalar function is a derivation. Generally speaking, a derivation D on a manifold M at $p \in M$ is a linear map, $D : C^\infty(M) \rightarrow \mathbb{R}$ satisfying the product rule, namely $D(f \cdot g)|_p = f(p)D(g) + g(p)D(f)$, $\forall f, g \in C^\infty(M)$. Let $D_p(M)$ denote the set of all derivatives at p on M. This implies that if a vector is the element of the tangent space at a point, it is also a derivation at the same point, $v \in T_p M \Rightarrow v \in D_p M$. This means that $T_p M \subset D_p(M)$.

2.1.3 Tangent maps

Consider two smooth manifolds M and N with diffeomorphisms φ and ψ respectively, a vector $v \in T_pM$, and a mapping $F : M \to N$ between the manifolds. Let x denote the local coordinate representation of M in the neighborhood of the point p, and y denote the local coordinate representation of N in the neighborhood of $F(p)$.

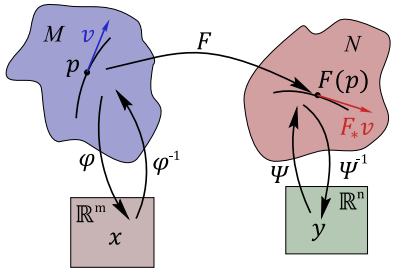

Fig. 5. Mapping between two manifolds

The function F in local coordinates reads $\psi \circ F \circ \varphi^{-1} : \mathbb{R}^m \to \mathbb{R}^n$. For the sake of simplicity F is interpreted as a function in coordinates, such that $y = F(x)$, omitting the notion that F is composed by diffeomorphisms. The tangent vector at point p is given by differentiating a curve in x coordinates. Then the same vector mapped from M to N by F can be computed by:

$$\dot{y} = \left[\frac{\partial F}{\partial x}\right]_p \dot{x}, \quad \dot{x} \in T_pM, \quad \dot{y} \in T_{F(p)}N \tag{2.6}$$

where $\left[\frac{\partial F}{\partial x}\right]_p$ is the Jacobian of F at point p. We also call it the tangent map of F, and the push-forward by F and can also be denoted by F_*. This means that F_* at $\mathrm{p} : T_pM \to T_{F(p)}N$ is a linear mapping, the velocities from one manifold to another always map linearly.

2.1.4 Vector Fields and integral curves

A vector field V on a manifold is a map which at each point on a manifold gives us a vector v in the tangent space of the same point. A vector field on an n-dimensional manifold in local coordinates looks like a vector valued function on $\mathbb{R}^n$. The set of all vector fields on the manifold M is denoted by $\Gamma(M)$. Then the following expression hold true: $V \in \Gamma(M) : M \to TM$ so that $V(p) \in T_pM$.

For example the vector field $v = \begin{bmatrix} y & -x & 0 \end{bmatrix}^T$ on S^2 gives us a vector at each point p, that is tangent to the sphere at that point p.

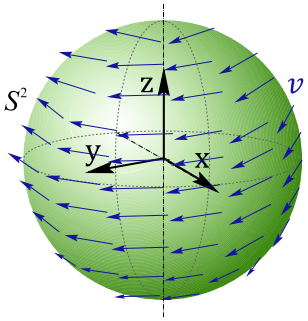

Fig. 6. Vector field defined on a sphere

If we imagine a point particle on the manifold M that flows along a vector field V and we draw its trajectory over time, we get an integral curve of the vector field.

Given $V \in \Gamma(M)$ and $p \in M$, an integral curve of V through p is a curve $\gamma : I \subset \mathbb{R} \to M$ which satisfies $\gamma(0) = p$ and $\frac{d}{dt}\gamma(t) = V(\gamma(t)) = V \circ \gamma(t),\ \forall t \in I$. In local coordinates we can write

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} V_1(x_1, \cdots, x_n) \\ \vdots \\ V_n(x_1, \cdots, x_n) \end{bmatrix},\ \mathrm{x}(0) = p \tag{2.7}$$

which is a system of ordinary differential equations (with an initial condition), its solution for the given interval I is an integral curve.

Flow. Given a vector field $V \in \Gamma(M)$ let $\phi_t^V : M \to M$ define the flow of V. It takes a point p on the manifold and gives back another point, which we get if starting from p we flow along the vector field V for time t.

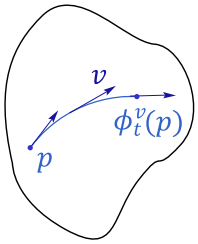

Fig. 7. Flow and tangent vector on a manifold

For example take a vector field on $\mathbb{R}^n$ in the following form:

$$V(x) = Ax,\ \ V(x) \in T\mathbb{R}^n = \mathbb{R}^n,\ \ x \in \mathbb{R}^n,\ \ A \in \mathbb{R}^{n \times n}. \tag{2.8}$$

We can rewrite it as

$$\dot{x} = Ax \tag{2.9}$$

And it is easy to see that the solution to this ODE is

$$x(t) = e^{At} x_0 \tag{2.10}$$

for a given x_0. This implies that the flow map for V is

$$\phi_t^V(x) = e^{At} x \tag{2.11}$$

The flow is dependent on time t and it is possible to partially differentiate the flow w.r.t the time to get back the vector field which generated it:

$$\frac{\partial}{\partial t}\phi_t^V(p) = V(\phi_t^V(p)) \tag{2.12}$$

where the left side of the equation is the Jacobian or push-forward.

2.1.5 Lie derivative and Lie bracket

Let us have a function $h \in C^\infty : M \to \mathbb{R}$ and a vector field $V \in \Gamma(M)$ on an *n*-dimensional manifold *M*. The Lie derivative of *h* along *V* is defined by

$$L_V h = V(h) = V \cdot h = \sum_{i=1}^{n} V_i \cdot \frac{\partial h}{\partial x_i} \tag{2.13}$$

which is the directional derivative of *h* along *V*. The Lie derivative is an operator that acts on *h*, the operator itself is in the form

$$L_V = \sum_{i=1}^{n} V_i \cdot \frac{\partial}{\partial x_i} \tag{2.14}$$

We can apply the Lie derivative multiple times. Given two vector fields $f, g \in \Gamma(M)$ and a function $h \in C^\infty(M)$, one possibility is to take the Lie derivative of *h* along *g* and then apply the Lie derivative along *f* to get

$$L_f\left(L_g h\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(f_i g_j \frac{\partial^2 h}{\partial x_j \partial x_i} + f_i \frac{\partial g_i}{\partial x_i}\frac{\partial h}{\partial x_j} \right) \tag{2.15}$$

By applying it the other way around the results reads

$$L_g\left(L_f h\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(g_i f_j \frac{\partial^2 h}{\partial x_j \partial x_i} + g_i \frac{\partial f_i}{\partial x_i}\frac{\partial h}{\partial x_j} \right) \tag{2.16}$$

The result is not a Lie derivative along some vector field $V \in \Gamma(M)$ in either cases, which implies it is not a derivation either. However, if we use the following expression

$$L_f\left(L_g h\right) - L_g\left(L_f h\right) = \sum_{i=1}^{n}\left(\sum_{j=1}^{n}\left(f_j \frac{\partial g_i}{\partial x_j} - g_j \frac{\partial f_i}{\partial x_j} \right)\right)\frac{\partial h}{\partial x_i} \tag{2.17}$$

the result becomes a derivation of *h* along a new vector field. This operation is called the Lie bracket and is one of the most important concepts used in geometric control. The usual notation of the Lie bracket is

$$[f, g] = L_f\left(L_g\right) - L_g\left(L_f\right). \tag{2.18}$$

Also there is another useful notation when this operator is applied several times, which is

$$ad_f g = [f,g],\ ad_f^2 g = [f,[f,g]],\ ad_f^3 g = [f,[f,[f,g]]],\ \cdots \tag{2.19}$$

and so on. It is easy to see the usefulness of this notation, when several Lie brackets are applied.

The Lie bracket is also interpreted as the measure of commutation. If the two vectors fields commute then the Lie bracket is 0. Commutation is understood with the help of the flows. Starting from a point *x*, we flow along the vector field *f* for some time *t*, then flow along *g* for the same time *t*, then flow backwards along *f* (the same as going backwards in time along *f*) and finally flow backwards along *g* for time *t*, we get a curve by changing the travel time *t*. The tangent vector to this curve at $t = 0$ is the value of the vector field $[f, g]$ at point *x*, which is illustrated in Fig. 8.

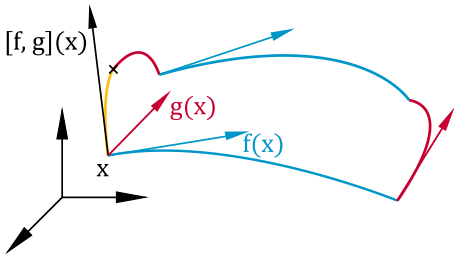

Fig. 8. The geometric interpretation of the Lie bracket

Let us look at an example with vector fields

$$f = \begin{bmatrix} y \\ -x \\ 0 \end{bmatrix},\ g = \begin{bmatrix} 0 \\ z \\ -y \end{bmatrix}, \quad f, g \in \Gamma\left(S^2\right). \tag{2.20}$$

The Lie bracket of f and g is

$$[f, g] = \begin{bmatrix} -z \\ 0 \\ x \end{bmatrix} \in \Gamma\left(S^2\right) \tag{2.21}$$

From a mechanical point of view, if f and g gives us directions in which we can generate velocity, then the newly acquired vector field we got from the Lie bracket means that we can also generate velocity along that direction by appropriate "switching" between the two vector fields.

The set of all possible vector fields $\Gamma(M)$ on a manifold M with the addition of the Lie bracket as an operator is a Lie algebra. The Lie bracket can be thought of as a operation that takes two elements from $\Gamma(M)$ and gives back another element from $\Gamma(M)$.

$$[\,\cdot\,,\,\cdot\,]: \Gamma(M) \times \Gamma(M) \to \Gamma(M) \tag{2.22}$$

The Lie bracket is an operation satisfying the following properties

- Linearity: $[f, [g + h]] = [f, g] + [f, h]$
- Skew symmetry: $[f, g] = -[g, f]$
- Jacobian identity: $[f, [g, h]] + [h, [f, g]] + [g, [h, f]] = 0$

where $f, g, h \in \Gamma(M)$.

2.1.6 Integrability of distributions, Frobenius theorem

We arrived at an important concept of differential geometry linked to the controllability of dynamical systems. Let us introduce it through an example. Consider two vector fields over $\mathbb{R}^3$: $g_1 = [y \quad -x \quad 0]^T$, $g_2 = [0 \quad z \quad -y]^T$, $g_1, g_2 \in \Gamma\left(\mathbb{R}^3\right)$ and the following question: is there an (embedded) surface in $\mathbb{R}^3$ such that the vector fields g_1 and g_2 are always tangent to it? (We define an embedded surface by a function $f: \mathbb{R}^3 \to \mathbb{R}$ where $f(x, y, z) = const.$ is a surface.)

To check if the vector field is tangent to a surface, we take the dot product of the vector field and the normal of the surface. If the result is zero, then they are tangent to each other. So, if there exists such a surface the following conditions have to be satisfied:

$$g_1 \cdot \nabla f = 0 \text{ and } g_2 \cdot \nabla f = 0. \tag{2.23}$$

This is true in Euclidean space, to generalize it for manifolds we use our notion of the Lie derivative.

$$L_{g_1} f = y\frac{\partial f}{\partial x} - x\frac{\partial f}{\partial y} = 0 \qquad L_{g_2} f = z\frac{\partial f}{\partial y} - y\frac{\partial f}{\partial z} = 0 \tag{2.24}$$

The problem boils down to solving a system of partial differential equations. In this case the solution can be easily obtained

$$f(x,y,z) = x^2 + y^2 + z^2 + c \tag{2.25}$$

where c is a constant. This means that there is such a surface where the given two vector fields are always tangent to it and it is a sphere.

To generalize this concept, also known as the integrability of tangent subbundles, let us introduce the notions of distributions first. A distribution Δ on a manifold M is a mapping that selects for each point $p \in M$ a subspace of T_pM. Thus, the distribution can be spanned by a set of vector fields.

$$\Delta = span\{g_1, g_2, \cdots, g_m\} \quad g_i \in \Gamma(M) \quad i = 1,2,\cdots,m \tag{2.26}$$

There are multiple properties that can defined for distributions:

- regular distribution: this means that the distribution has constant dimension for all points of the manifold.
- integrable distribution: there exists a submanifold $N \subset M$ such that $\Delta(p) = T_pN \quad \forall p \in N$.
- Involutive distribution: the distribution is closed for Lie brackets: $[g_i, g_j] \in \Delta \quad \forall g_i, g_j \in \Delta$.

The Frobenius theorem says that a regular smooth distribution is integrable if and only if it is involutive.

Going back to our previous example, this means that the distribution given by the two vector fields is integrable which implies that it is also involutive. This theorem is important because the problem of integrability boils down to checking Lie brackets instead of trying to solve partial differential equations.

2.1.7 Covector fields and codistributions

Let us introduce the concept of covector fields and codistributions, which we also use later on. The Frobenius theorem will also be reformulated again using these notions. A covector can be thought of as a function that can be applied to a vector. Let us say that X is a vector space. This has a dual space X^* which is the set of linear, real valued functions, such that $v^* : X \to \mathbb{R}$, where $v^* \in X^*$. If we apply an element of X^* to a vector v, the notation is given in a scalar product form $\langle v^*, v \rangle$. For example, the dual space of $\mathbb{R}^n$ is denoted as $\mathbb{R}^{n*}$. Note that the dual space is also a vector space. There also exist dual tangent spaces. Given a manifold M with the tangent space T_pM at point $p \in M$, the dual tangent space is denoted by T_p^*M. The vectors of the dual tangent space are called tangent convectors. Suppose that $\omega_1, \cdots, \omega_n$ are all smooth, real valued function of the real variables $x_1, \cdots, x_n \in U \subset \mathbb{R}^n$. A covector field can be defined in local coordinates as

$$\omega(x) = \begin{bmatrix} \omega_1(x_1,\cdots,x_n) & \omega_2(x_1,\cdots,x_n) & \cdots & \omega_3(x_1,\cdots,x_n) \end{bmatrix}. \tag{2.27}$$

If not stated otherwise we consider covector fields to be in row vector form. Covector fields obtained by differentiation from a real-valued functions as

$$d\lambda(x)=\left[\frac{\partial\lambda}{\partial x_1}\quad\frac{\partial\lambda}{\partial x_2}\quad\cdots\quad\frac{\partial\lambda}{\partial x_3}\right]=\frac{\partial\lambda}{\partial x} \tag{2.28}$$

are referred to as exact differentials. Using this notion some previous concepts can be revisited. For example, the Lie derivative is the same as applying an exact differential covector field to a vector field f, since

$$\left\langle d\lambda(x), f(x)\right\rangle=\frac{\partial\lambda}{\partial x}f(x)=\sum_{i=1}^{n}\frac{\partial\lambda}{\partial x_i}f_i(x). \tag{2.29}$$

Going back to the notion of distribution, we see that a dual object, the codistribution can be also defined. Codistributions can be defined as spans of covector fields $\omega_1,\cdots,\omega_d$, where $\omega_i:\mathbb{R}^n\to\mathbb{R}^{n^*}$ such that

$$\Omega=span(\omega_1,\cdots,\omega_d). \tag{2.30}$$

Let us introduce now the annihilation property of a codistribution. We say that the annihilator of a distribution Δ is the codistribution if

$$\Delta^{\perp}=\left\{v^*\in\mathbb{R}^{n^*}:\left\langle v^*,v\right\rangle=0,\ \forall v\in\Delta\right\}. \tag{2.31}$$

Let us now go back to the Frobenius theorem. Consider a distribution Δ in $\mathbb{R}^n$ and let d denote its dimension. Also consider the vector fields $g_1,\cdots,g_d$, such that

$$\Delta=span\{g_1,\cdots,g_d\}. \tag{2.32}$$

Let the annihilator codistribution of Δ be $\Omega=\Delta^{\perp}$ which has dimension $n-d$ and spanned by the covectors $\omega_1,\cdots,\omega_{n-d}$. By virtue of annihilation, the following statement hold true

$$\left\langle\omega_j(x),g_i(s)\right\rangle=0,\quad\forall 1\le i\le d,\ 1\le \text{j}\le \text{n-d}, \tag{2.33}$$

so the covector field $\omega_j(x)G(x)=0$, where $G(x)=\left[g_1(x)\quad\cdots\quad g_d(x)\right]$, which has rank d, since we said that Δ has dimension d. The space of the solutions is spanned by the $n-d$ linearly independent convectors, which implies that $\omega_1(x),\cdots,\omega_{n-d}(x)$ are the basis of the solution space. To arrive at the problem formulated by the Frobenius theorem, we say that we only accept solutions in the form

$$\omega_j=\frac{\partial\lambda_j}{\partial x} \tag{2.34}$$

where $\lambda_1,\cdots,\lambda_{n-d}$ are real-valued smooth functions. This means that we seek the solution of the partial differential equation

$$\frac{\partial\lambda_j}{\partial x}\left[g_1(x)\quad\cdots\quad g_d(x)\right]=\frac{\partial\lambda_j}{\partial x}G(x)=0 \tag{2.35}$$

where we need to find $n-d$ independent solutions, such that the row vectors

$$\frac{\partial\lambda_1}{\partial x},\cdots,\frac{\partial\lambda_{n-d}}{\partial x} \tag{2.36}$$

are linearly independent at all x. This is the same problem as stated in the chapter regarding the Frobenius theorem. The question now becomes: Given a distribution Δ, does it have an annihilator $\Delta^{\perp}$ which is spanned by exact differentials? This annihilator is given by

$$\Delta^{\perp} = span\{d\lambda_1, \cdots, d\lambda_{n-d}\}. \tag{2.37}$$

2.2 Feedback linearization

This section is dedicated to the theory of exact linearization, which transforms a nonlinear system into a linear one by state-feedback and coordinate transformation. We mainly present the approach used in [3], however feedback linearization can be extended to a special class of systems called differentially flat system [5], which is not discussed here.

2.2.1 Linearization of SISO systems

In this subsection we consider an affine nonlinear control systems in the form

$$\dot{x} = f(x) + g(x)u \tag{2.38}$$

where $x \in \mathbb{R}^n$ is the vector of the state variables, u is the input, f is the drift vector field and g is the control vector field. First, we consider input-state linearizable systems. By definition, if a system is input-state linearizable, then there exists a coordinate transformation $z = \Phi(x)$, which has to be a diffeomorphism, and a fedback $u = \alpha(x) + \beta(x)v$ where v will be the new input of the closed-loop system so that we obtain $\dot{z} = Az + bv$ in closed-loop.

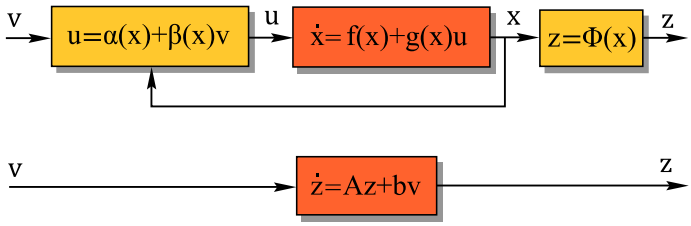

Fig. 9. Exact linearization via feedback with coordinate transformation and the equivalent linear system

By substitution the following expression is obtained:

$$\dot{z} = \left[\frac{\partial \Phi}{\partial x}\right]\left(f + g(\alpha + \beta v)\right) = Az + bv, \quad \forall \mathrm{v}. \tag{2.39}$$

This equation can be broken down to two separate equations in the form

$$\begin{aligned} \left[\frac{\partial \Phi}{\partial x}\right](f + g\alpha) &= Az \\ \left[\frac{\partial \Phi}{\partial x}\right] g &= \frac{1}{\beta} B \end{aligned} \tag{2.40}$$

It is obvious that $\beta \neq 0$, since that would mean that the closed-loop system cannot be controlled, our new input is multiplied by β. Without loss of generality we take matrix A and vector b in the control canonical form, such that all the poles lie at zero.

$$A := \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{2.41}$$

Now, let us look at $\left[\frac{\partial \Phi}{\partial x}\right]$ which is the Jacobian of the diffeomorphism. If we consider the form of the Jacobian matrix we see that

$$\left[\frac{\partial \Phi}{\partial x}\right] = \begin{bmatrix} \frac{\partial \Phi_1}{\partial x_1} & \cdots & \frac{\partial \Phi_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Phi_n}{\partial x_1} & \cdots & \frac{\partial \Phi_n}{\partial x_n} \end{bmatrix} \tag{2.42}$$

where Φ_i and x_i is the *i*-th element of the diffeomorphism and the state vector respectively. Looking at the Jacobian matrix it is easy to see that each row is the gradient of each element of the diffeomorphism Φ. Using this fact our original two equations can be rewritten as

$$\begin{bmatrix} \nabla \Phi_1{}^T \\ \nabla \Phi_2{}^T \\ \vdots \\ \nabla \Phi_n{}^T \end{bmatrix} (f + g\alpha) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \qquad \begin{bmatrix} \nabla \Phi_1{}^T \\ \nabla \Phi_2{}^T \\ \vdots \\ \nabla \Phi_n{}^T \end{bmatrix} g = \frac{1}{\beta} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{2.43}$$

Using the notations introduced earlier and simplifying some terms, these equations can once again be rewritten in the form

$$\begin{aligned} L_f \Phi_1 &= \Phi_2 & L_g \Phi_1 &= 0 \\ L_f \Phi_2 &= \Phi_3 & L_g \Phi_2 &= 0 \\ &\vdots & &\vdots \\ L_f \Phi_n &= -\frac{\alpha}{\beta} & L_g \Phi_n &= \frac{1}{\beta} \end{aligned} \tag{2.44}$$

that are actually $2n$ partial differential equations. The first question that arises: is this system of equations solvable? Many PDEs may not have a solution, so before continuing it is advised to check if it is solvable. Firstly let us look at Φ_1. Take the Lie bracket of f and g,

$$L_{[f,g]} \Phi_1 = L_f L_g \Phi_1 - L_g L_f \Phi_1 = 0 \tag{2.45}$$

since $L_g \Phi_1 = 0$, $L_f \Phi_1 = \Phi_2$ and $L_g \Phi_2 = 0$. It is possible to take Lie brackets and simplifying up to $L_{[ad^{n-2} f, g]} \Phi_1 = 0$. This creates the following system of equations:

$$L_g \Phi_1 = 0, \; L_{[f,g]} \Phi_1 = 0, \; L_{[f,[f,g]]} \Phi_1 = 0, \; \cdots, \; L_{[ad^{n-2} f, g]} \Phi_1 = 0 \tag{2.46}$$

for which we seek a $\Phi_1 = const.$ satisfying all of them, in other words a surface in $\mathbb{R}^n$ such that $\left\{g,\ [f,g],\ \left[f,[f,g]\right],\ \cdots,\ \left[ad_f^{n-2},g\right]\right\}$ is everywhere tangent to that surface. This surface exists if the distribution $\Delta = span\left\{g,\ [f,g],\ \left[f,[f,g]\right],\ \cdots,\ \left[ad_f^{n-2},g\right]\right\}$ is integrable, and by Frobenius theorem it is integrable if and only if it is involutive. The question of the solvability thus boils down to checking Lie brackets and determining whether the arising new vector fields are part of the distribution.

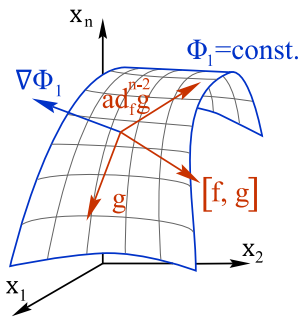

Fig. 10. The connection between an integrable distribution and an embedded, lower dimension surface

This property is a necessary but not sufficient condition for the input-state linearization. Brockett provided a second condition which is analogous to the linear system's controllability matrix's rank condition by which a theorem is formed:

A nonlinear system in the form (2.38) is input-state linearizable if and only if:

1. The distribution $\Delta = span\left\{g,\ [f,g],\ \left[f,[f,g]\right],\ \cdots,\ \left[ad_f^{n-2},g\right]\right\}$ is integrable
2. $span\left\{g,\ [f,g],\ \left[f,[f,g]\right],\ \cdots,\ \left[ad_f^{n-1},g\right]\right\} = \mathbb{R}^n$.

We present a simple example to summarize the above. Consider an inverted pendulum with a motor at the joint.

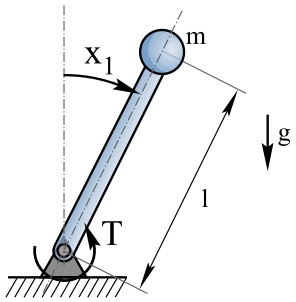

Fig. 11. Inverse pendulum

The rod is massless, has length l, at the end has a point mass m and its angle in relation to the horizontal position is denoted by x_1, which is also the first state variable. The joint has viscous friction σ and the motor has a time constant τ. The dynamics read

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix}=\begin{bmatrix}x_2\\ -\sigma x_2+\frac{g}{l}\sin x_1+x_3\\ -\frac{1}{\tau}x_3\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \frac{1}{\tau}\end{bmatrix}u \tag{2.47}$$

First, we check if the system is input-state linearizable. This is performed by looking at the distribution $\Delta_1 = span\{g,[f,g]\}$ and check if it is involutive according to the Frobenius theorem.

$$\Delta_1 = span\left\{\begin{bmatrix}0 & 0 & \frac{1}{\tau}\end{bmatrix}^T,\begin{bmatrix}0 & -\frac{1}{\tau} & \frac{1}{\tau^2}\end{bmatrix}^T\right\} \tag{2.48}$$

Indeed this distribution is involutive, since $[g,[f,g]]=0\in\Delta_1$. The second condition that needs to be satisfied is

$$\Delta_2 = span\{f,[f,g],[f,[f,g]]\}=\mathbb{R}^3. \tag{2.49}$$

If we compute

$$[f,[f,g]]=\begin{bmatrix}\frac{1}{\tau} & -\left(\frac{b}{\tau}+\frac{1}{\tau}\right) & \frac{1}{\tau^3}\end{bmatrix}^T \tag{2.50}$$

it is easy to see that $[f,[f,g]]$ is linearly independent from f and $[f,g]$, so Δ_2 is indeed equivalent to $\mathbb{R}^3$. This implies that this system is input-state linearizable. Next we construct the diffeomorphism $z=\Phi(x)$ and the functions $\alpha(x)$ and $\beta(x)$. This is achieved by solving the following system of partial differential equations

$$\begin{aligned} &L_f\Phi_1(x)=\Phi_2(x) && L_g\Phi_1(x)=0\\ &L_f\Phi_2(x)=\Phi_3(x) && L_g\Phi_2(x)=0\\ &L_f\Phi_3(x)=-\frac{\alpha(x)}{\beta(x)} && L_g\Phi_3(x)=\frac{1}{\beta(x)}\end{aligned} \tag{2.51}$$

Looking at $L_g\Phi_1(x)=\frac{1}{\tau}\frac{\partial\Phi_1}{\partial x_3}=0$ it becomes clear that $\Phi_1=\Phi_1(x_1,x_2)$ so it does not depend on x_3. Substituting into $L_f\Phi_1(x)=\Phi_2(x)$ we get

$$x_2\frac{\partial\Phi_1}{\partial x_1}+\left(-bx_2+\frac{g}{l}\sin(x_1)+x_3\right)\frac{\partial\Phi_1}{\partial x_2}=\Phi_2 \tag{2.52}$$

which yet again substituted into $L_g\Phi_2(x)=0$ yields us

$$L_g\Phi_2(x)=\frac{1}{\tau}\frac{\partial\Phi_1}{\partial x_2}=0 \tag{2.53}$$

which implies that $\Phi_1=\Phi_1(x_1)$. Since $L_f\Phi_1(x_1)=\Phi_2(x)$ we get

$$\Phi_2(x) = x_2 \frac{\partial \Phi_1}{\partial x_1} \tag{2.54}$$

which substituted into $L_f \Phi_2(x) = \Phi_3(x)$ gives us

$$\Phi_3(x) = x_2\left(x_2 \frac{\partial^2 \Phi_1}{\partial x_1^2}\right) + \left(-bx_2 + \frac{g}{l}\sin x_1 + x_3\right)\frac{\partial \Phi_1}{\partial x_1}. \tag{2.55}$$

Taking into consideration that $L_g \Phi_3(x) = \frac{1}{\tau}\frac{\partial \Phi_3}{\partial x_3} \neq 0$ tells us that $\frac{\partial \Phi_1}{\partial x_1} \neq 0$. This means that we need to choose a Φ_1 that is only dependent on x_1 and not constant. An obvious choice then is $\Phi_1(x) = x_1$. Obtaining the remaining elements of the coordinate transformation is now trivial, hence

$$\Phi(x) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ -\sigma x_2 + \frac{g}{l}\sin x_1 + x_3 \end{bmatrix}. \tag{2.56}$$

The function $\beta(x)$ can be computed by $\frac{1}{L_g \Phi_3(x)}$ and $\alpha(x)$ by $-\beta(x) L_f \Phi_3(x)$ respectively.

$$\beta(x) = \tau \;\; \alpha(x) = \tau\left(-\frac{g}{l}x_2 \cos x_1 + \sigma \frac{g}{l}\sin x_1 - \sigma^2 x_2 + \sigma x_3 + \frac{1}{\tau}x_3\right) \tag{2.57}$$

This concludes our example to construct an input-state linearizing feedback for SISO systems.

There is however the possibility that a system is not input-state linearizable. For this consider a nonlinear system in the form

$$\begin{aligned} \dot{\mathrm{x}} &= \mathrm{f}(\mathrm{x}) + \mathrm{g}(\mathrm{x})\mathrm{u} \\ \mathrm{y} &= \mathrm{h}(\mathrm{x}) \end{aligned} \tag{2.58}$$

where y is the output of the system and the function h describes its relation to the states. Our aim is to find a diffeomorphism $\mathrm{z} = \Phi(\mathrm{x})$ and a control input $\mathrm{u} = \alpha(\mathrm{x}) + \beta(\mathrm{x})\mathrm{v}$ which transforms the states from x into (ξ, η) and the original system into

$$\dot{\xi} = \mathrm{A}\xi + \mathrm{bv} \tag{2.59}$$

$$\dot{\eta} = \mathrm{q}(\xi, \eta) \tag{2.60}$$

where $\mathrm{A} \in \mathbb{R}^{\mathrm{r}\times\mathrm{r}}$, $\mathrm{b} \in \mathbb{R}^{\mathrm{r}}$, $\xi \in \mathbb{R}^{\mathrm{r}}$, $\mathrm{f}_0 : \mathbb{R}^{\mathrm{r}} \times \mathbb{R}^{\mathrm{n}-\mathrm{r}} \to \mathbb{R}^{\mathrm{n}-\mathrm{r}}$ and $\mathrm{r} \leq \mathrm{n}$. Equation (2.59) describes the linear dynamics and equation (2.60) describes the remaining dynamics after feedback linearization. It is easy to see that if $\mathrm{r} = \mathrm{n}$ the system is input-state linearizable, since there won't be any "leftover" nonlinear dynamics. $\frac{\mathrm{d}^{\mathrm{r}}\mathrm{y}}{\mathrm{dt}^{\mathrm{r}}} = \mathrm{v}$. The value of r is called the relative degree. We say that the system (2.38) has a relative degree of $\mathrm{r} \leq \mathrm{n}$ at $\mathrm{x}_0 \in \mathrm{M}$ if

$$\begin{aligned} L_g L_f^i h &= 0 \quad \text{i=1,}\cdots\text{,r-2} \\ L_g L_f^{r-1} h &\neq 0 \quad \forall x \in U(x_0) \subset M \end{aligned} \tag{2.61}$$

This way the system with the new states becomes

$$\begin{aligned} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \vdots \\ \dot{\xi}_{r-1} \\ \dot{\xi}_r \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{r-1} \\ \xi_r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} v \\ \dot{\eta} &= q(\xi, \eta) \end{aligned} \tag{2.62}$$

Some remarks about relative degree:

- Relative degree is the number of differentiations of y until u appears.
- Relative degree depends on $y = h(x)$.
- For a fixed y the relative degree is invariant.
- If the relative degree $r = n$, then the system is input-state linearizable.

This concludes the theory of feedback linearization of SISO systems.

2.2.2 Linearization of MIMO systems

The results discussed in the previous section can be generalized for MIMO systems as well. The dynamics of a MIMO system are described by

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x) u_i \tag{2.63}$$

where we suppose that the vector fields g_i are linearly independent. The outputs are defined as

$$y_i = h_i(x) \quad i = 1, 2, \cdots, m \tag{2.64}$$

where we suppose that the covector fields dh_i are linearly independent.

The relative degree of the outputs, if it exists is the vector $\begin{bmatrix} r_1 & r_1 & \cdots & r_m \end{bmatrix}$ and the following exact differentials (covectors) are independent

$$\begin{aligned} \omega_1 &= \begin{bmatrix} dh_1 & dL_f h_1 & \cdots & dL_f^{r_1 - 1} h_1 \end{bmatrix} \\ \omega_2 &= \begin{bmatrix} dh_2 & dL_f h_2 & \cdots & dL_f^{r_2 - 1} h_2 \end{bmatrix} \\ &\cdots \\ \omega_m &= \begin{bmatrix} dh_m & dL_f h_m & \cdots & dL_f^{r_m - 1} h_m \end{bmatrix} \end{aligned} \tag{2.65}$$

The zero dynamics are also similar to that of the SISO systems. Since the system has m inputs and outputs, there will be m series of integrators. The following expression is the coordinate transformation, where the upper index of the variables indicates which series of integrators (or in other words which input/output) are we considering.

$$\begin{aligned}
&\xi_1^i = \phi_1^i(x) = y_i = h_i(x)\\
&\xi_2^i = \phi_2^i(x) = \frac{dy_i}{dt} = L_f h_i(x)\\
&\vdots\\
&\xi_{r_i}^i = \phi_{r_i}^i(x) = \frac{d^{r_i-1}y_i}{dt^{r_i-1}} = L_f^{r_i-1}h_i(x)
\end{aligned} \tag{2.66}$$

The dynamics corresponding to these variables read

$$\begin{aligned}
&\dot{\xi}_1^i = \xi_2^i\\
&\dot{\xi}_2^i = \xi_3^i\\
&\vdots\\
&\dot{\xi}_{r_i-1}^i = \xi_{r_i}^i\\
&\dot{\xi}_{r_i}^i = L_f^{r_i}h_i(x) + \sum_{j=1}^{m} L_{g_j}L_f^{r_i-1}h_i(x)u_j
\end{aligned} \tag{2.67}$$

Now let $r = \sum_{i=1}^{m} r_i$. It is always possible to find $n - r$ scalar functions $\phi_{r+1}(x), \phi_{r+2}(x), \cdots, \phi_n(x)$, which satisfy

- $L_{g_i}\phi_j(x) = 0 \quad 1 \le \mathrm{i} \le \mathrm{m},\ \mathrm{r}+1 \le \mathrm{j} \le \mathrm{n}$ and
- $\Phi(x) = \left[\phi_1^1(x) \ \cdots \ \phi_{r_1}^1(x) \ \cdots \ \phi_1^m(x) \ \cdots \ \phi_{r_m}^m(x) \ \phi_{r+1}(x) \ \cdots \ \phi_n^1(x)\right]^T$, which is a coordinate transformation being invertible.

As with the SISO case, the state variables for the remaining dynamics which do not belong to the series of integrators are ordered in a vector and denoted by

$$\eta = \left[\phi_{r+1}(x) \quad \phi_{r+2}(x) \quad \cdots \quad \phi_n(x)\right]^T \tag{2.68}$$

Let us introduce now a matrix $A(x)$ multiplying the vector of the inputs in the dynamics. This has elements

$$a_{ij}(\xi,\eta) = L_{g_j}L_f^{r_i-1}h_i\left(\Phi^{-1}(\xi,\eta)\right), \quad 1 \le \mathrm{i}, \mathrm{j} \le \mathrm{m} \tag{2.69}$$

and also introduce a vector with elements

$$b_i(\xi,\eta) = L_f^{r_i}h_i\left(\Phi^{-1}(\xi,\eta)\right) \tag{2.70}$$

Using these notations, the dynamics of the whole system can be rephrased in the form

$$\begin{aligned}
&\dot{\xi}_1^i = \xi_2^i\\
&\vdots\\
&\dot{\xi}_{r_i-1}^i = \xi_{r_i}^i\\
&\dot{\xi}_{r_i}^i = b_i(\xi,\eta) + \sum_{j=1}^{m} a_{ij}(\xi,\eta)u_j\\
&\dot{\eta} = q(\xi,\eta)
\end{aligned} \tag{2.71}$$

If $r_1 + \cdots + r_m = n$, then the system is fully linearizable by state feedback and coordinate transformation and there will be no remaining dynamics.

The zero dynamics in question are obtained if we set the output to zero, such that

$$\dot{\eta} = q(0, \eta) \tag{2.72}$$

The relative degree has the following properties

- $L_{g_j} L_f^k h_i(x) = 0$ for all $1 \le j \le m$, $k < r_i - 1$ and $1 \le i \le m$.
- The following matrix is nonsingular

$$A(x) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1} h_1(x) & \cdots & L_{g_m} L_f^{r_1 - 1} h_1(x) \\ L_{g_1} L_f^{r_2 - 1} h_2(x) & \cdots & L_{g_m} L_f^{r_2 - 1} h_2(x) \\ \vdots & \cdots & \vdots \\ L_{g_1} L_f^{r_m - 1} h_m(x) & \cdots & L_{g_m} L_f^{r_m - 1} h_m(x) \end{bmatrix} \tag{2.73}$$

Suppose that our system has relative degree, which means that $A(x)$ is invertible. The new inputs for the closed loop system equal the input for the chain of integrators.

$$v = b(\xi) + A(\xi) u \tag{2.74}$$

So the input u is easily obtained as

$$u = A^{-1}(\xi)\big(v - b(\xi)\big) \tag{2.75}$$

This means that these kinds of systems can be linearized by a static feedback. One example from this class of systems is a robotic arm with open kinematic chain, which will be discussed later in the following chapter.

Now consider the system

$$\dot{x} = f(x) + g_1(x) u_1 + g_2(x) u_2 \tag{2.76}$$

where

$$\begin{aligned} x &= \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T, \; f(x) = \begin{bmatrix} 0 & x_4 & \lambda x_3 + x_4 & 0 \end{bmatrix}^T \\ g_1(x) &= \begin{bmatrix} 1 & x_3 & 0 & 0 \end{bmatrix}^T, \; g_2(x) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \end{aligned} \tag{2.77}$$

and the outputs are

$$\begin{aligned} y_1 &= h_1(x) = x_1 \\ y_2 &= h_2(x) = x_2 \end{aligned} \tag{2.78}$$

The first step is to determine the relative degree of the system, which is done by the simple calculations of Lie derivatives and the matrix (2.73). This system has no relative degree since

$$L_g h(x) = \begin{bmatrix} 1 & 0 \\ x_3 & 0 \end{bmatrix} \tag{2.79}$$

is singular. This problem occurred because the lowest (first) order derivatives of the outputs are both affected by u_1 and none by u_2. To resolve this one may try to "delay" the appearance of the input u_1, to derivatives of higher order, where, hopefully, u_2 will also appear, making the matrix … nonsingular. In this case this is achieved by attaching an integrator at the first input of the plant, which means adding a new state variable to the closed loop system. So we need to set

$$\begin{aligned} u_1 &= \zeta \\ \dot{\zeta} &= v_1 \end{aligned} \tag{2.80}$$

To have a consistent notation we also set

$$u_2 = v_2 \tag{2.81}$$

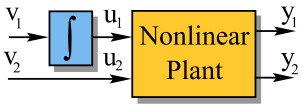

Fig. 12. Dynamic extension

The composition of this auxiliary system with the original system results in the dynamics

$$\tilde{f}(x,\zeta) = \begin{bmatrix} \zeta \\ x_4 + x_3\zeta \\ \lambda x_3 + x_4 \\ 0 \\ 0 \end{bmatrix}, \quad \tilde{g}_1(x,\zeta) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \tilde{g}_2(x,\zeta) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.82}$$

Going back to the calculation of the relative degree to determine matrix (2.68) we get

$$L_{\tilde{g}} h(x,\zeta) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.83}$$

and

$$L_{\tilde{g}} L_{\tilde{f}} h(x,\zeta) = \begin{bmatrix} 1 & 0 \\ x_3 & 1 \end{bmatrix} \tag{2.84}$$

which means that this augmented system has relative degree vector $[2 \quad 2]$. This process is called dynamic extension. Now the exact linearization process can be applied exactly same way as described before, the linearizing feedback however is a dynamic one. Unfortunately there is no proven way to tell which outputs yield themselves to this property.

2.3 Robust control

Robustness plays a major role in real-life engineering applications, since these systems are subject to disturbances, there are noises on the measured signals and the accurate mathematical model of the dynamics may not even be known. These factors can compromise the performance or even the stability of the closed-loop. The field of robust control takes these uncertainties into consideration when designing a controller such that the real system can operate reliably. Luckily, the theory is thoroughly developed to analyze systems subject to uncertainty and to perform the controller synthesis [6].

This chapter summarizes some concepts used in robust control. It is important to note that the robust control theory presented here considers linear uncertain systems. This suits our needs since we aim to control nonlinear system which are already linearized by a state feedback with nominal parameter values that are different from the actual parameter values of the system, hence the uncertainty.

2.3.1 Norms

This subsection reviews the norms of signals and systems. One definition of a norm is the *p*-norm. It gives a measure of a vector in a finite dimensional vector space *X* (which could be $\mathbb{R}^n$ or $\mathbb{C}^n$). The norm of $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$ is given by

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, \quad 1 \le \mathrm{p} < \infty \tag{2.85}$$

If $p = 2$ we get the well-known Euclidean norm. In the case of $p = \infty$ the definition of the norm becomes

$$\|x\|_p = \sup_{t \in \mathbb{R}} |x(t)|. \tag{2.86}$$

Signals in continuous time can be considered as continuous functions of time, denoted by $x(t)$. These type of functions live on an infinite dimensional vector space. The *p* norm in this case reads

$$\|x\|_p = \left(\int_{-\infty}^{\infty} |x(t)|^p \, dt \right)^{\frac{1}{p}} \tag{2.87}$$

This holds true if $1 \le p < \infty$. In the case of $p = \infty$ the expression is modified to

$$\|x\|_\infty = \sup_{t \in \mathbb{R}} \|x(t)\|_\infty \ . \tag{2.88}$$

Normed spaces consisting of signals with finite norms are denoted by $\mathcal{L}^p$. For different values of *p* we obtain different properties as measures. With $p = 1$ the notion $\|x\|_1$ gives us the integral of the absolute value of $x(t)$, for $p = 2$ $\|x\|_2$ is also known as the energy of the signal. The ∞-norm, $\|x\|_\infty$ gives us the peak value of $x(t)$, and it is bounded if $x(t) \in \mathcal{L}^\infty$. Stability of linear systems is understood as the bounded input, bounded output stability, or BIBO stability for short. This means that if the system is excited by a bounded signal at its input, it is considered BIBO stable if the output is also bounded.

Let us look at a bounded linear operator. These operators will represent the systems. A linear operator $A: X \to Y$, with X and Y as normed spaces is bounded if there exist a *c*, so that the expression $\|Ax\| \le c\|x\|$ holds true for all $x \in X$. Using this boundedness property of linear operators, it is possible to define an induced norm for the operator itself:

$$\|A\| = \sup_{\|x\|=1} \|Ax\| \tag{2.89}$$

Since we would like to analyze MIMO systems not just SISO ones, a measure for the size of matrices is also needed. This measure is obtained through the singular value decomposition. Let $A \in F^{m\times n}$, where $F = \mathbb{R}$ or $F = \mathbb{C}$. There exist unitary matrices

$$U = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix} \in F^{m\times m} \; V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \in F^{n\times n} \tag{2.90}$$

such that

$$A = U\Sigma V^*, \quad \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}, \tag{2.91}$$

where

$$\Sigma_1 = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \tag{2.92}$$

and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0, \quad k = \min\{m,n\}$. The largest singular value of a system or a matrix is denoted by $\bar{\sigma}[\cdot]$.

Now we focus on probably the most important norms in robust control theory, namely system norms. These norms are the input-output gains of the system. Let G be a linear and bounded system that maps the input signal $u(t)$ to the output signal $y(t)$. This system is a bounded linear operator and the induced norm of G is given by

$$\|G\| = \sup_{\|u\|=1} \|Gu\|. \tag{2.93}$$

We are interested mostly in the ∞-norm, which for our system $G: \mathcal{L}_n^2 \to \mathcal{L}_m^2$ (where n is the dimension of the input and m is the dimension of the output) is

$$\|G\|_\infty = \sup_{\omega\in\mathbb{R}} \|G(j\omega)\|_2 \tag{2.94}$$

where the notion $\|G(j\omega)\|_2$ refers to the spectral norm of the $m \times n$ matrix $G(j\omega)$. The spectral norm can be calculated using the singular values and the previously mentioned *p*-norm formula. Let $A \in F^{m\times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0, \quad k = \min\{m,n\}$. The spectral norm of A is

$$\|A\|_p = \left(\sum_{i=1}^{k} \sigma_i^p \right)^{\frac{1}{p}}. \tag{2.95}$$

This implies that the ∞-norm of the system G is the maximum energy gain of the system, which depends on the peak of the largest singular value of the frequency response matrix over the whole frequency axis.

We do not aim to introduce in detail the concepts of Hardy spaces and Hilbert space, which are function spaces frequently mentioned in robust control theory. However throughout our studies, we deal with

transfer functions that are an element of H_∞, which is a Banach space of functions that are analytic and bounded in the open right-half plane. In this space the norm is defined by

$$\|F\|_\infty = \sup_{\omega\in\mathbb{R}} \bar{\sigma}\left[F(j\omega)\right] \tag{2.96}$$

which is called the H_∞ norm. An interesting subspace of H_∞ is the space consisting of all proper and real rational stable transfer functions denoted by RH_∞.

2.3.2 Uncertainties

Models of real dynamics contain uncertainties and real systems are also subject to external disturbances and noises. Uncertainty is considered to be a mismatch between the dynamics of the real system and its mathematical model and we consider such uncertainties here. Concerning LTI systems, uncertainties are concentrated into a transfer matrix $\Delta(s)$. There are several ways to incorporate the unknown dynamics into the system model. Let $G(s)$ denote the uncertain system and $G_0(s)$ denote the nominal system. Then the uncertain models can be, but not limited to

- Additive uncertainty model

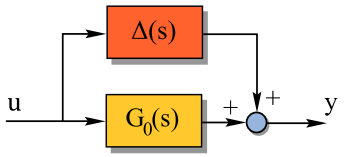

Fig. 13. Additive uncertainty

$$G(s) = G_0(s) + \Delta(s) \tag{2.97}$$

- Input multiplicative uncertainty model

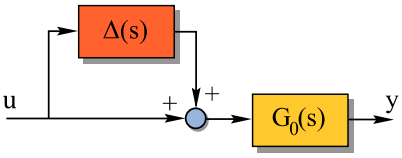

Fig. 14. Input multiplicative uncertainty

$$G(s) = G_0(s)\left[I + \Delta(s)\right] \tag{2.98}$$

- Output multiplicative model

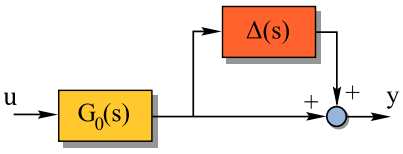

Fig. 15. Output multiplicative uncertainty

$$G(s) = \left[I + \Delta(s)\right]G_0(s) \tag{2.99}$$

The term $\Delta(s)$ is unknown, but norm-bounded. It can be bounded by a transfer function $W(j\omega)$ meaning the norm of the bounding transfer function is less than or equal that of the unknown transfer function at all frequencies such that $\bar{\sigma}\left[\Delta(j\omega)\right] \le \delta(j\omega)$. If the only known property of $\Delta(s)$ is its norm, then it is called an unstructured uncertainty. This way it is possible to choose $\|\Delta(s)\|_\infty \le 1$ without

loss of generality and shaping it by two weighting functions $W_1(s)$ and $W_2(s)$ according to the given application. In this thesis we mainly concern ourselves with the output multiplicative structure, however the uncertainty models presented are interchangeable. With these considerations in mind, the uncertain linear system is described by

$$G(s)=\left[I+W_1(s)\Delta(s)W_2(s)\right]G_0(s). \tag{2.100}$$

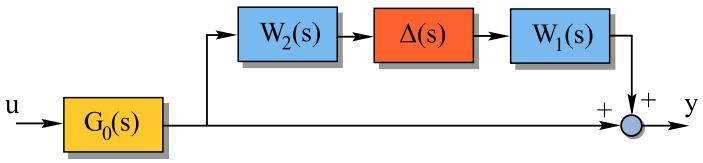

Fig. 16. Output multiplicative uncertainty model with uncertainty weighting functions

One needs to find the appropriate weighting functions $W_1(s)$ and $W_2(s)$ such that the inequality $\|\Delta(s)\|_\infty \le 1$ is satisfied. In the SISO case, $W_2(s)$ can be set to 1, and the equation rearranged so that

$$\frac{\|G(j\omega)-G_0(j\omega)\|}{\|G_0(j\omega)\|} \le \|W_1(j\omega)\| \quad \forall\omega>0 \tag{2.101}$$

holds true for any appropriate $W_1(s)$. The algorithm for determining the weighting functions for MIMO systems is presented in [7].

2.4 Stability and performance of control loops

Several requirements must be satisfied by the closed-loop with the controller to be deigned, including stability. Using the block diagram of a standard feedback loop depicted in Fig. 17 we can obtain the transfer function of the closed-loop.

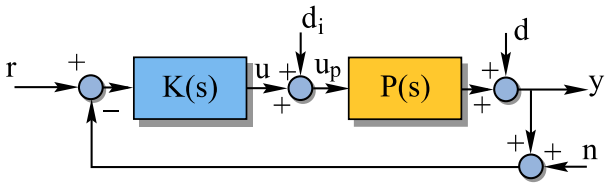

Fig. 17. Classical feedback control loop

$$\begin{aligned} y &= PKr - PKy \\ (I+PK)\,y &= PKr \\ W_{yr} &= (I+PK)^{-1}PK \end{aligned} \tag{2.102}$$

Since we generally consider MIMO systems, the multiplications are not commutative. The closed-loop can be rearranged for the analysis of internal stability the way it is shown in Fig. 18.

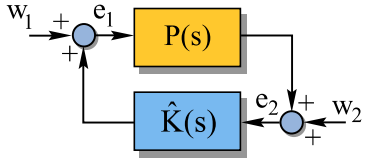

Fig. 18. Feedback loop used in the stability analysis

where

$$\begin{aligned} e_1 &= \hat{K}e_2 + w_1 \\ e_2 &= Pe_1 + w_2 \end{aligned} \tag{2.103}$$

which can be written in matrix form

$$\begin{bmatrix} I & -\hat{K} \\ -P & I \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}. \tag{2.104}$$

It is clear from the elementary theory of linear systems that a transfer function is only realizable if it is proper.

The system depicted in Fig. 18 is internally stable if it is well posed and

$$\begin{bmatrix} I & -\hat{K} \\ -P & I \end{bmatrix}^{-1} \in RH_\infty . \tag{2.105}$$

The basis of robust stability analysis is the so-called small-gain theorem. Consider the feedback structure depicted in Fig. 19, where $\Delta(s)$ and $M(s)$ are transfer function matrices.

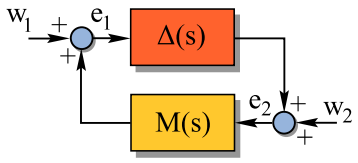

Fig. 19. Feedback loop used for the small-gain theorem

Small-gain theorem. Suppose that $M(s) \in RH_\infty$ and let $\gamma > 0$. The feedback structure shown in Fig. 19 is well-posed and internally stable for all $\Delta(s) \in RH_\infty$ if and only if:

- $\|\Delta\|_\infty \le \frac{1}{\gamma}$ and $\|M\|_\infty < \gamma$ or
- $\|\Delta\|_\infty \le \frac{1}{\gamma}$ and $\|M\|_\infty \le \gamma$

2.5 Linear fractional transformation

The linear fractional transformation allows rearranging the uncertain system's block diagram in a form where the unstructured uncertainty block and the controller can be "pulled-out". We distinguish between two types of LFT: upper LFT and lower LFT depending on if the "upper" or the "lower" loop is closed by the block $\Delta(s)$ or $K(s)$ as shown in Fig. 20.

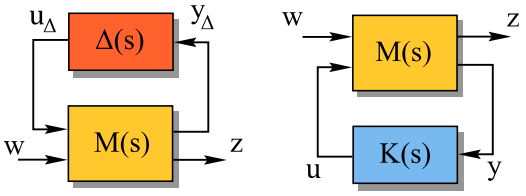

Fig. 20. Upper and lower linear fractional transformations

Suppose now that the transfer function matrix M is partitioned as

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}. \tag{2.106}$$

For upper LFT, the transfer reads

$$z = \left[M_{22} + M_{21}\Delta\left(I - M_{11}\Delta\right)^{-1} M_{12} \right] w \tag{2.107}$$

and exists if $\left(I - M_{11}\Delta\right)$ is invertible. Hence the expression

$$F_u\left(M,\Delta\right) = M_{22} + M_{21}\Delta\left(I - M_{11}\Delta\right)^{-1} M_{12} \tag{2.108}$$

is the lower linear fractional transformation of M and Δ. The lower LFT usually indicates how the controller is applied to the system and it reads

$$F_l\left(M,K\right) = M_{11} + M_{12}K\left(I - M_{22}K\right)^{-1} M_{21} \tag{2.109}$$

2.6 H_∞ synthesis

Fig. 22 shows the setup of the H-infinity synthesis. This contains the "pulled" out uncertainty block, the augmented plant and the robust controller $K\left(s\right)$. The synthesis problem is formulated as an optimization problem.

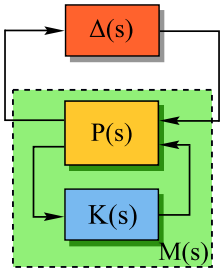

Fig. 22. The interconnection used for the H-infinity synthesis

The aim is to minimize the infinity norm of $M\left(s\right) = F_l\left(P\left(s\right),K\left(s\right)\right)$, which is determined by the choice of $K\left(s\right)$, since $P\left(s\right)$ is given. This is the so-called optimal H-infinity problem, formulated as

$$\min_{K \text{ stabilizing}} \left\| F_l\left(P\left(s\right),K\left(s\right)\right) \right\|_\infty \tag{2.110}$$

There is unfortunately no analytic formula that gives the solution to this problem. However, in practice it is sufficient to solve a slightly different problem, called the suboptimal H-infinity problem, given as

$$\left\| F_l\left(P\left(s\right),K\left(s\right)\right) \right\|_\infty < \gamma \tag{2.111}$$

The scalar γ may be larger than the global minimum of the norm, hence the term "suboptimal'.

Multiple possibilities exist to get the solution of the suboptimal H-infinity problem, which given the limits of the thesis is not detailed, however the solution itself is presented.

Let us introduce the solution of the suboptimal H-infinity problem, which means, that if we take the block diagram in Fig. 22, the augmented system is given by

$$G(s)=\begin{bmatrix} A & B_1 & B_2 \\ C_1 & 0 & D_{12} \\ C_2 & D_{21} & 0 \end{bmatrix} \tag{2.112}$$

Consider the following properties true:

- the pair (A,B_1) is controllable and the pair (C_1,A) is observable
- the pair (A,B_2) is stabilizable and the pair (C_2,A) is detectable
- $D_{12}^*\begin{bmatrix} C_1 & D_{12} \end{bmatrix}=\begin{bmatrix} 0 & I \end{bmatrix}$
- $\begin{bmatrix} B_1 \\ D_{21} \end{bmatrix}D_{21}^*=\begin{bmatrix} 0 \\ I \end{bmatrix}$

Now to solve the suboptimal H-infinity problem take the matrices

$$H_\infty=\begin{bmatrix} A & \gamma^2 B_1B_1^* - B_2B_2^* \\ -C_1^*C_1 & -A^* \end{bmatrix}\ J_\infty=\begin{bmatrix} A^* & \gamma^2 C_1^*C_1 - C_2^*C_2 \\ -B_1B_1^* & -A \end{bmatrix} \tag{2.113}$$

corresponding to the tow Ricatti equations. We seek the stabilizing solution . Let us suppose that there exists a stabilizing solution to H_∞ in the form $X_\infty = Ric(H_\infty)$ and there exists a stabilizing solution for J_∞ in the form $Y_\infty = Ric(J_\infty)$. Also, let us assume that $\rho(X_\infty Y_\infty) < \gamma^2$ which means that the spectral radius of the matrix is less than γ^2.

Omitting the details, a realization of the suboptimal controller $K_{sub}(s)$ satisfying the conditions mentioned above is given by

$$K_{sub}(s)=\begin{bmatrix} \hat{A}_\infty & -Z_\infty L_\infty \\ F_\infty & 0 \end{bmatrix} \tag{2.114}$$

where the matrices of the realization are based on X_∞ and Y_∞ such that

$$\hat{A}_\infty = A+\gamma^{-2}B_1B_1^*X_\infty + B_2F_\infty + Z_\infty L_\infty C_2 \quad F_\infty = -B_2^*X_\infty \quad L_\infty = -Y_\infty C_2^* \quad Z_\infty = \left(I-\gamma^{-2}Y_\infty X_\infty\right)^{-1}. \tag{2.115}$$

Using Matlab, the synthesis is executed by providing the transfer function matrix of the augmented plant and also the number of measured and control signals as the argument of the Matlab command `hinfsyn`. This command returns the controller as an LTI system in state space form, and also some additional results such as the dynamics of the closed-loop as an LTI system and the value of the achieved γ [8].

This concludes the theoretical summary, where the aim was to highlight the necessary background and notations, used in the sequel.

3 Control architecture to robustify the exact linearization

The proposed control architecture combines the nonlinear plant, its linearizing feedback and a robust controller in a cascaded structure.

Our studies focus on the synthesis of a 2DOF robust controller which is expected to exhibit better dynamical response than a serial compensator although the workflows of designing a robustifying serial and 2DOF compensators are similar. In fact, the 2DOF structure aims to improve the performance presented in [9], where earlier results with a robustifying serial compensator was proposed. The 2DOF controller is synthetized by solving a standard H-infinity, mixed sensitivity optimization problem discussed in the previous chapter.

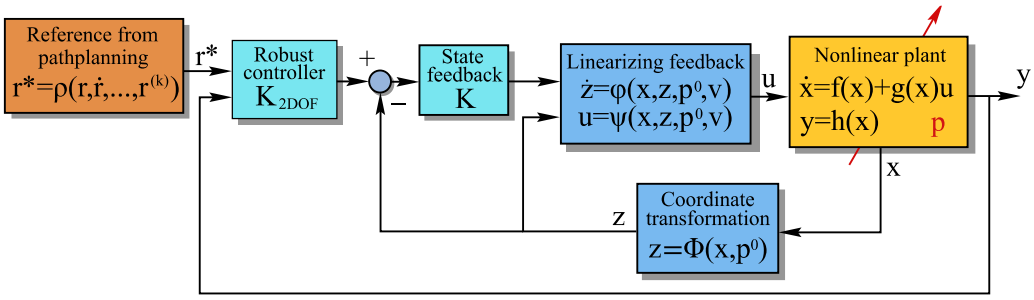

Fig. 23. The proposed control architecture

Fig. 23 shows the control architecture, which may contain a MIMO system linearized by dynamic feedback. The state feedback is used to move the poles from 0 to places defined by the designer to ensure quick and stable transients for the system, while also considering the arising controller efforts, for example by a linear quadratic regulator (LQR). The outer loop contains the robustifying controller . The input of the 2DOF controller is the result of the path planning that uses the derivatives of the original reference signal, a more detailed explanation is presented later on at the MIMO examples.

The uncertain parameters span the parameter space. A point in this space is described by the parameter vector $p=\begin{bmatrix} p_1 & p_2 & \cdots & p_N \end{bmatrix}^T$. Each parameter takes its value inside a bounded interval $Q_i \subset \mathbb{R}$, hence the whole parameter space is defined by $Q=Q_1 \times Q_2 \times \cdots Q_N$. The nominal parameter vector is denoted by $p^0 \in Q$.

For the robust controller synthesis, a set of linear systems must be obtained to describe the behavior of the dynamics for different parameter values in Q. Let $\pi_{NL}(p)$ denote the closed-loop dynamics, containing the nonlinear system with parameters p, and the linearizing feedback designed using p^0. This implies that $\pi_{NL}(p^0)=G_0(s)$, but there is no reason that $\pi_{NL}(p)$ be linear in general. Consider

$$\Pi_Q=\{\pi_{NL}(p): p\in Q\} \tag{3.1}$$

and define a sufficiently fine grid to cover the uncertainties as

$$P=\{p^i : i=1,\cdots,P; p^i \in Q\}. \tag{3.2}$$

Let $G^i(s)$ denote the linearized dynamic of $\pi_{NL}(p^i)$. An output multiplicative structure is considered to cover the set of these linearized dynamics, hence the weighting functions $W_1(s)$ and $W_2(s)$ must satisfy

$$G^i(s) = \left(I + W_1(s)\Delta(s)W_2(s)\right)G_0(s) \qquad \forall i \in [1, P] \tag{3.3}$$

such that $\|\Delta(s)\|_\infty \leq 1$. After defining the performance weighting functions the optimization can be carried out to synthetize the robust controller. For all of the examples presented later, the following (standard) performance weighting functions were used: a closed-loop model transfer function $M(s)$, describing the desired closed-loop behavior; a transfer function $W_u(s)$ to specify the available control signal bandwidth; and the weighting function $W_e(s)$ to penalize the difference of the model and system outputs, or in other words, the closed-loop model error. These are connected to the output multiplicative structure with the 2DOF controller as shown in Fig. 24.

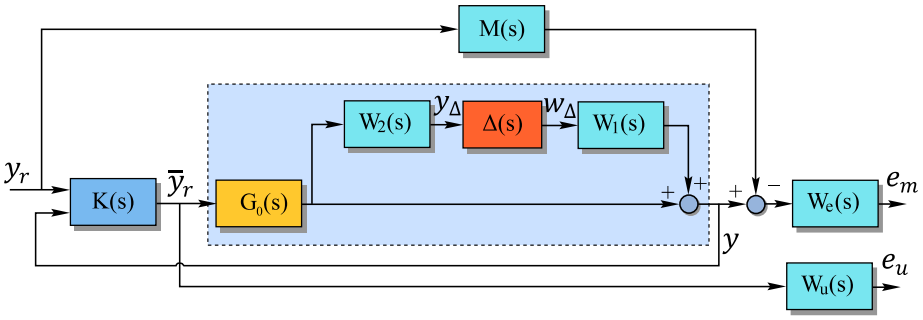

Fig. 24. The interconnection with the uncertainty and performance weighting functions

This block diagram is used after pulling out the uncertainty block $\Delta(s)$ and controller $K(s)$ as it is shown before in the theoretical overview. This leaves an augmented plant denoted by $G_a(s)$ which is the transfer between the input signal $[u_\Delta, y_r, \bar{y}_r]^T$ and the output signal $[y_\Delta, e_m, e_u, y_r, y]^T$, where the notions are concise with Fig. 24. So the transfer matrix of the augmented plant reads

$$G_a(s) = \begin{bmatrix} 0 & 0 & W_2(s)G_0(s) \\ W_e(s)W_1(s) & -W_e(s)M(s) & W_e(s)G_0(s) \\ 0 & 0 & W_u(s) \\ 0 & I & 0 \\ W_1(s) & 0 & G_0(s) \end{bmatrix}. \tag{3.4}$$

We seek a controller $K(s)$ that minimizes the H-infinity norm of the lower LFT of $G_a(s)$ and $K(s)$, in other words, the optimal controller minimizes $\|F_l(G_a(s), K(s))\|_\infty$.

In the next few sections several nonlinear example systems have been studied to demonstrate the design workflow and to study the usefulness of the controllers. We start with some simple SISO examples.

3.1 Nonlinear spring-mass system (SISO)

Let us consider the system depicted in Fig. 25. The friction between the mass and the floor is neglected, the spring attached to the mass and the wall has nonlinear characteristics and its stiffness is given by

$$k(x) = cx + \varepsilon x^2 \tag{3.5}$$

where x denotes the displacement of the mass, and $c, \varepsilon \in \mathbb{R}$ are constants. Using Newton's second law the dynamics of the system are easily obtained and read

$$\ddot{x} + \frac{c}{m}x + \frac{\varepsilon}{m}x^2 = \frac{1}{m}F \tag{3.6}$$

where m denotes the mass and F denotes the external force input. The output of the system is x, so $y = x$.

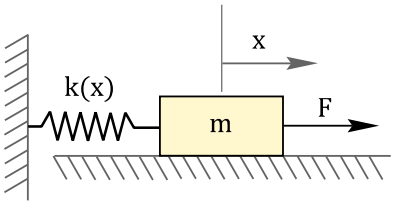

Fig. 25. The spring-mass system

We assume that the control signal is the external force $F = u$. The linearizing feedback has the form

$$u = cx + \varepsilon x^2 - 2\xi\omega_0 m\dot{x} - \omega_0^2 mx + \omega_0^2 mv \tag{3.7}$$

and it is easy to see that applying this feedback results the closed-loop being an underdamped second-order system in the form

$$G_0(s) = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 + \omega_0^2} \tag{3.8}$$

where ξ and ω_0 are design parameters, set to 0.7 and 2, respectively, in our example. The uncertain parameters are summarized in Table 1.

Table 1. The uncertain parameters for the spring-mass system

Parameter [unit]	Nominal value	Min. value	Max. value
$m\ [kg]$	1	0.6	1.4
$c\ [N/m]$	0.6	0.4	0.8
$\varepsilon\ [N/m^2]$	0.2	0.1	0.3

A grid with 125 vertices is constructed in the parameter space. A set of linear systems for the robust controller design is obtained by taking the previously discussed nonlinear system with parameter values at each vertex of the grid, applying the linearizing feedback with nominal parameters and lastly, by approximating the dynamics by the linear part of its Taylor series neglecting the higher-order terms. Matlab's Robust Control Toolbox is used to cover the uncertainties by an adequate $W_1(s)$ transfer function. In the SISO case it is possible to choose $W_2(s)$ as 1. The $W_1(s)$ transfer function reads

$$W_1(s) = \frac{4.3 \cdot 10^{-7} s^2 + 0.0009595s + 0.7353}{s^2 + 2.796s + 4.746} \tag{3.9}$$

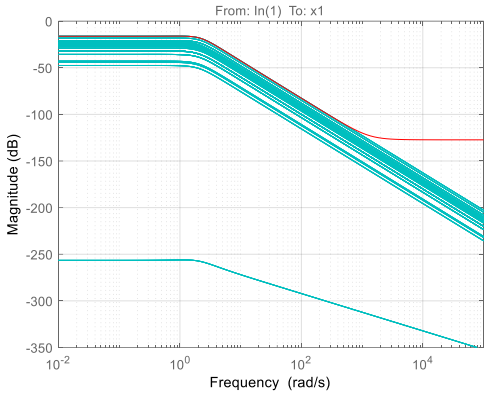

Fig. 26. The gain of the uncertainty weighting transfer function

The chosen model transfer function is a second order underdamped one, that reads

$$M(s) = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 + \omega_0^2} \tag{3.10}$$

which is the same as the nominal transfer function $G_0(s)$. The performance weighting functions for the control signal and output error are set to be

$$W_u(s) = \frac{10(s+10\omega_0)^2}{(s+200\omega_0)^2} \qquad W_e(s) = \frac{800\omega_0}{s+4\omega_0}. \tag{3.11}$$

The weighting functions play a major role in finding an adequate controller and are the result of a trial and error process. The designer needs to find a trade-off between controller effort and tracking error using these transfer functions. The augmented plant now can be constructed as described in (3.4). The robust controller that minimizes the H-infinity norm is of order 17. The 2DOF controller has excellent disturbance attenuation, which is shown in the simulations.

The system has been simulated with three closed-loops variants: a variant without a robustifying controller (referred to as NORC), one with a robust serial compensator (referred to as SRC) and one with 2DOF controller (referred to as 2DOFRC). All three controller structures are simulated for the same six, randomly sampled parameter combinations. The reference signal is set to be a step function that jumps from 0 to 1 at 1 second. An external disturbance is also added, in form of a step function that jumps from 0 to 1 at 5 seconds.

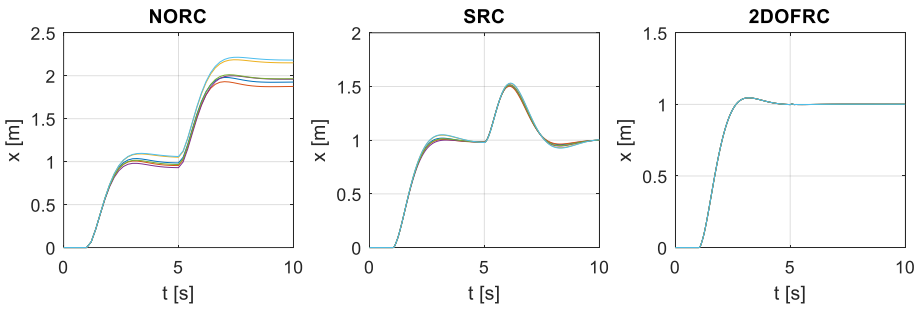

Fig. 27. The output of the SISO system for the three controller variants

Fig. 27 shows the plant output for the three controller variants. It is seen that the NORC solution performs poorly as the output does not converge to one, and the disturbance makes the error substantially larger. The SRC variant has better transients, but disturbance attenuation is still not satisfactory. With

the 2DOFRC, the response is robustly close to the second-order specification and the effect of the disturbance is barely visible.

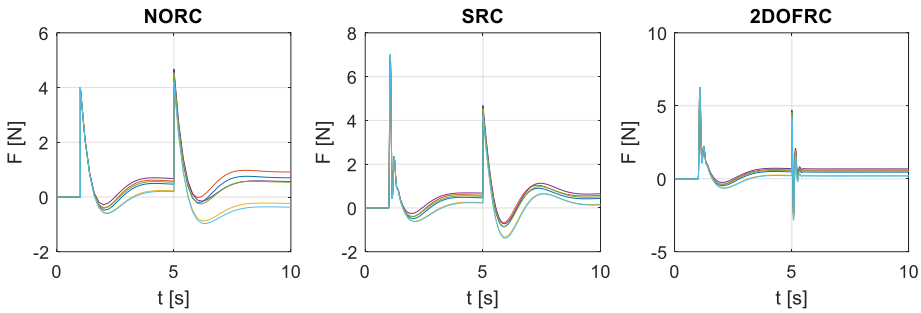

Fig. 28. The controller efforts of the SISO system for the three controller variants

By looking at the outputs of the different controllers at Fig. 28, we see that the 2DOFRC does not require significantly more controller effort than the other setups. We may conclude that for this SISO system, the best closed-loop performance is achieved by the 2DOF robust controller.

3.2 Robotic arm (MIMO)

The first MIMO example is an RR-type robotic arm, which moves in the horizontal plane. This is a fully actuated system and we consider all links to be rigid. Its mathematical model can be obtained by the Euler-Lagrange equations.

A real version of this robotic arm can be found at the department, so the geometric and inertial parameters are that of the real system. During the derivation of the model we use a general algorithm to show, that this synthesis is easily applicable to all fully actuated open-chain mechanisms [10].

The relevant parameters for the real system are specified in Fig. 29. These are:

- The coordinate systems fixed to each link. The choice of these coordinate systems is done by using the Denavit-Hartenberg (DH) convention.
- The length of the links: l_1 and l_2.
- The center of mass of the links: C_1 and C_2.
- The coordinates of the center off masses in their respective coordinate systems: l_{1cx}, l_{1cy} and l_{2cx}.
- The relative angle between the links: q_1 and q_2.
- The masses and moment of inertias at the center of mass: $m_1, m_2 \in \mathbb{R}$, $\Theta_1^1, \Theta_2^2 \in \mathbb{R}^{3\times 3}$. The lower index for the moment of inertia matrix denotes the corresponding link and the upper index denotes the coordinate system where it is expressed. It is advised to express this matrix in its "own" coordinate system which is fixed to the link, since then it becomes constant. Usually the axes of the DH-coordinate systems coincide with the principal axes of the moment of inertia matrix, so it is in the form

$$\Theta_i^i = \begin{bmatrix} \Theta_{i,x}^i & 0 & 0 \\ 0 & \Theta_{i,y}^i & 0 \\ 0 & 0 & \Theta_{i,z}^i \end{bmatrix}. \tag{3.12}$$

The Euler-Lagrange equations read

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = Q_i \quad \forall \text{i=1,2,}\cdots\text{,n} \tag{3.13}$$

where T is the kinetic energy and U is the potential energy of the system, q_i is the i-th general coordinate and Q_i is the corresponding generalized force (or moment). The term Q_i is separated as

$$Q_i = \tau_i - \tau_{d,i} \tag{3.14}$$

where τ_i is the active torque of the i-th motor applied between the i-1 and i-th links. The term $\tau_{d,i}$ symbolizes the disturbance torques. These disturbances generally come from friction or for example in our case gravity, since the real physical robotic arm doesn't perfectly horizontal. For the mathematical model we only consider the linear part of these disturbances, which are the viscous friction of the joints, setting $\tau_{d,i} = \sigma_i \dot{q}_i$, where σ_i is the coefficient of the viscous friction.

We can write the kinetic energy in the form

$$T = \frac{1}{2}\dot{q}^T H(q)\dot{q} \tag{3.15}$$

where $q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^T$, $\dot{q} = \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & \cdots & \dot{q}_n \end{bmatrix}^T$ and $M(q) \in \mathbb{R}^{n \times n}$ is the generalized inertia matrix. In our case $n = 2$.

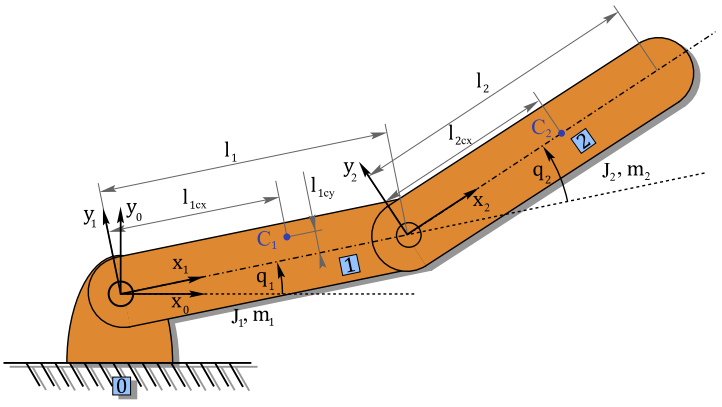

Fig. 29. The two-degrees-of-freedom robotic arm

To obtain the generalized inertia matrix, the first step is to determine the appropriate transformation matrices, which are

$$T_{0,1}(q_1) = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_{1,2}(q_2) = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

where $c_i = \cos q_i$ and $s_i = \sin q_i$. These matrices are in the form

$$T_{i-1,i} = \begin{bmatrix} R_{i-1,i} & p_{i-1,i} \\ 0^T & 1 \end{bmatrix}, \quad R_{i-1,i} \in SO(3), \quad p_{i-1,i} \in \mathbb{R}^3. \tag{3.17}$$

The second step is the expression of the coordinates of the center of masses in their appropriate coordinates systems, and their transformation back to the inertial frame, which is fixed to the base of the robotic arm.

$$r_{1c}^1 = \begin{bmatrix} l_{1cx} & l_{1cy} & 0 \end{bmatrix}^T,\ r_{2c}^2 = \begin{bmatrix} l_{2cx} & 0 & 0 \end{bmatrix}^T \tag{3.18}$$

where the upper index indicates the coordinate system where the vector is expressed. To transform the vectors back to the base coordinate system we apply

$$\begin{bmatrix} r_i^0 \\ 1 \end{bmatrix} = T_{0,1}(q_1) \cdot T_{1,2}(q_2) \cdot \ldots \cdot T_{i-1,i}(q_i) \cdot \begin{bmatrix} r_i^i \\ 1 \end{bmatrix},\ \ \forall \text{i=1,}\cdots\text{,n}\ . \tag{3.19}$$

After this, the velocities of the center of masses are obtained in the base coordinate system by calculating the Jacobians and multiplying it by the first derivative with respect of time of the general coordinate vector q.

$$\dot{r}_i^0 = \frac{\partial r_i^0(q_1,\cdots,q_i)}{\partial q}\dot{q} = J_{vi}\dot{q},\quad \forall \text{i=1,}\cdots\text{,n} \tag{3.20}$$

where $J_{vi} \in \mathbb{R}^{3\times n}$ is the Jacobian matrix corresponding to the velocity of the i-th link's center of mass. The angular velocities are also needed to express the kinetic energy. These are in the form

$$\left[\omega_i^0 \times\right] = \dot{R}_{0,i} \cdot R_{0,i}^T \tag{3.21}$$

where $[\omega\times]$ denotes the skew symmetric matrix of the cross product, which satisfies $[\omega\times]r = \omega \times r,\ \forall \omega, \text{r} \in \mathbb{R}^3$ and

$$[\omega\times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{3.22}$$

The Jacobian matrix for the angular velocity of the i-th link in the base coordinate system is obtained by

$$J_{\omega,i} = \frac{\partial \omega_i^0}{\partial \dot{q}}\ . \tag{3.23}$$

Now the generalized mass matrix can easily be expressed by

$$H(q) = \sum_{i=1}^{n}\left(m_i J_{v,i}^T \cdot J_{v,i} + J_{\omega,i}^T \cdot R_{0,i} \cdot \Theta_i^i \cdot R_{0,i}^T \cdot J_{\omega,i}\right). \tag{3.24}$$

Applying the Euler-Lagrange equations we get

$$H(q)\ddot{q} + \left(\sum_{i=1}^{n}\frac{\partial H(q)}{\partial q_i}\dot{q}_i\right)\dot{q} - \frac{1}{2}\dot{q}^T\left(\sum_{i=1}^{n}\frac{\partial H(q)}{\partial q_i}\right)\dot{q} + \frac{\partial U}{\partial q} - \sigma\dot{q} = \tau\,. \tag{3.25}$$

where $\sigma = \begin{bmatrix} \sigma_1 & \cdots & \sigma_n \end{bmatrix}^T$ and $\tau = \begin{bmatrix} \tau_1 & \cdots & \tau_n \end{bmatrix}^T$. Let us rewrite this equation to a simpler form

$$H(q)\ddot{q} + h(q,\dot{q}) = \tau \tag{3.26}$$

where $h(q,\dot{q}) = \left(\sum_{i=1}^{n} \frac{\partial H(q)}{\partial q_i} \dot{q}_i \right) \dot{q} - \frac{1}{2} \dot{q}^T \left(\sum_{i=1}^{n} \frac{\partial H(q)}{\partial q_i} \right) \dot{q} + \frac{\partial U}{\partial q} - \sigma \dot{q}$. Going back to our example, the following expressions are obtained

$$H(q) = \begin{bmatrix} \Theta_{1,z} + \Theta_{2,z} + m_1 \left(l_{1cx}^2 + l_{1cy}^2 \right) + m_2 \left(l_1^2 + 2 c_2 l_1 l_{2cx} + l_{2cx}^2 \right) & \Theta_{2,z} + m_2 \left(l_{2cx}^2 + c_2 l_1 l_{2cx} \right) \\ \Theta_{2,z} + m_2 \left(l_{2cx}^2 + c_2 l_1 l_{2cx} \right) & \Theta_{2,z} + m_2 l_{2cx}^2 \end{bmatrix} \tag{3.27}$$

$$h(q,\dot{q}) = \begin{bmatrix} -m_2 l_1 l_{2cx} s_2 \dot{q}_2 \left(2\dot{q}_1 + \dot{q}_2 \right) \\ m_2 l_1 l_{2cx} s_2 \dot{q}_1^2 \end{bmatrix}. \tag{3.28}$$

Since $H(q)$ is a positive-definite matrix, its inverse, $H(q)^{-1}$ exists. The state-space representation of the system is

$$\begin{aligned} \frac{d}{dt} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} &= \begin{bmatrix} \dot{q} \\ -H(q)^{-1} h(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} 0 \\ H(q)^{-1} \end{bmatrix} \tau \\ y &= q \end{aligned} \tag{3.29}$$

The robotic arm is exact linearizable by static feedback with

$$\tau = H(q) v + h(q,\dot{q}) \tag{3.30}$$

which is also referred to as the computed torque method. The closed-loop becomes

$$\ddot{q} = v. \tag{3.31}$$

The same result is obtained following the algorithm for the exact linearization of MIMO systems via feedback. The reader can check that the relative degree of the system is $r = \begin{bmatrix} r_1 & r_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \end{bmatrix}$ with

$$A(x) = \begin{bmatrix} L_{g1} L_f^{r_1 - 1} h_1(x) & L_{g2} L_f^{r_1 - 1} h_1(x) \\ L_{g1} L_f^{r_2 - 1} h_2(x) & L_{g2} L_f^{r_2 - 1} h_2(x) \end{bmatrix} = H(q)^{-1} \qquad b(x) = \begin{bmatrix} L_f^{r_1} h_1(x) & L_f^{r_2} h_2(x) \end{bmatrix}^T = -H(q)^{-1} h(q,\dot{q}) \tag{3.32}$$

where $x = \begin{bmatrix} q_1 & q_2 & \dot{q}_1 & \dot{q}_2 \end{bmatrix}^T$ and the state space model is partitioned into the form

$$\dot{x} = f(x) + g_1(x) u_1 + g_2(x) u_2 \qquad y_1 = h_1(x) = x_1, \;\; y_2 = h_2(x) = x_2. \tag{3.33}$$

The feedback law is $u = \alpha(x) + \beta(x) v$ with $\alpha(x) = -A(x)^{-1} b(x)$ and $\beta(x) = A(x)^{-1}$. This way the closed loop becomes

$$\begin{aligned} \ddot{x}_1 &= \dot{x}_3 = v_1 \\ \ddot{x}_2 &= \dot{x}_4 = v_2 \end{aligned} \tag{3.34}$$

which is clearly equivalent to (3.31).

To ensure exponential decay of the reference tracking error $e = r - q$, we need to set the coefficients λ_1 and λ_2 of the differential equation

$$\ddot{e} + \lambda_1 \dot{e} + \lambda_2 e = 0 \tag{3.35}$$

appropriately. To this aim, let us choose the eigenvalues of the error dynamics to be negative real numbers such that

$$s^2 + \lambda_1 s + \lambda_2 = (sT+1)(sT+1) = s^2T^2 + 2sT + 1 = 0\,. \tag{3.36}$$

Thus the coefficients are

$$\lambda_1 = \frac{2}{T},\quad \lambda_2 = \frac{1}{T^2}\,. \tag{3.37}$$

Then the nominal system's closed loop transfer function has the form

$$G_0 = diag\left\{\frac{\lambda_2}{s^2 + \lambda_1 s + \lambda_2}\right\}. \tag{3.38}$$

Now let us suppose that the robustifying controller in the outer loop successfully enforces this transfer function on the exact linearized system. Then if we set the input of the controller to

$$r^* = r + \frac{\lambda_1}{\lambda_2}\dot{r} + \frac{1}{\lambda_2}\ddot{r} \tag{3.39}$$

then the output becomes

$$Y(s) = G_0(s)\cdot R^*(s) = diag\left\{\frac{\lambda_2}{s^2 + \lambda_1 s + \lambda_2}\right\}\cdot\left(R(s) + \frac{\lambda_1}{\lambda_2}sR(s) + \frac{1}{\lambda_2}s^2R(s)\right) = R(s) \tag{3.40}$$

where $Y(s), R^*(s), R(s)$ are the Laplace transforms of the vectors $y(t), r^*(t), r(t)$. The complete control loop for the robotic arm is shown in Fig. 30, a similar approach was presented in [11].

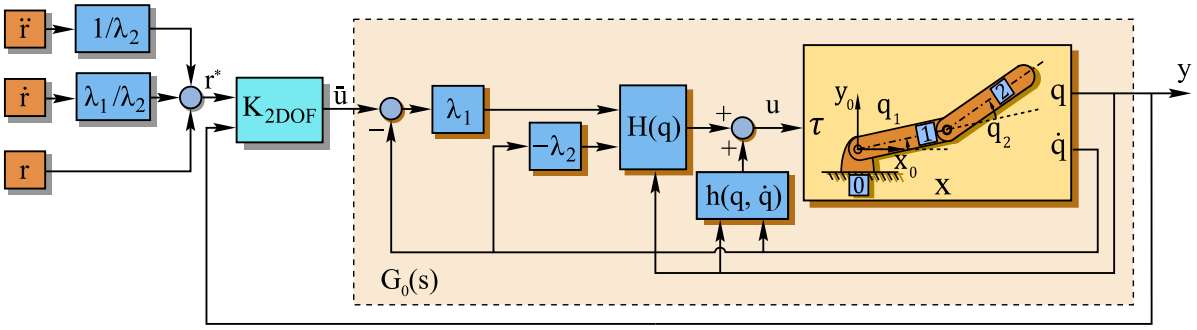

Fig. 30. The proposed controller solution for the robotic arm

For the robotic arm in our lab, we consider the geometrical parameters ($l_1, l_2, l_{1cx}, l_{1cy}, l_{2cx}$) and masses ($m_1, m_2$) known, system identification is performed for the inertias ($\Theta_{1,z}, \Theta_{2,z}$) and the viscous friction coefficients ($\sigma_1.\sigma_2$). The parameters are identified separately for each link. The first link is fixed to the base and the second link is rotated by the second motor. Luckily the construction of the robotic arm is such, that this second link can be rotated around indefinitely. After that, the second link is fixed to the first as depicted in Fig. 31. A relay-based identification method is used in both cases. Since static friction is present, which is a nonlinear effect, the relay's input is the angular velocity of the link in such a way, that the link's rotation doesn't change direction and does not get close to zero. This way the effect of static friction can be eliminated.

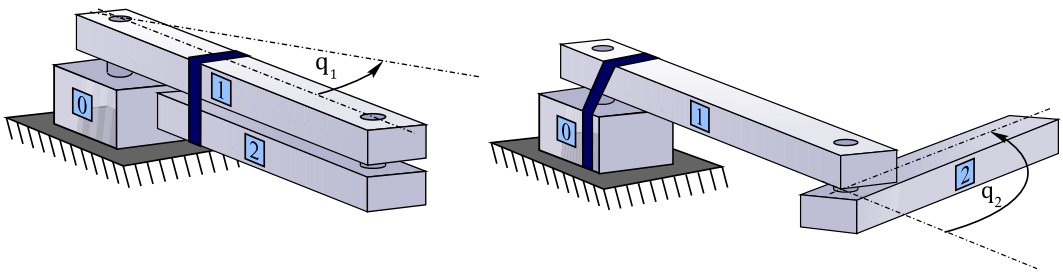

Fig. 31. The setups used for the parameter identification

Let us focus on the identification of the second link, the method used for the first link being similar. The identification is done using Matlab. The sampling time of the data collection was $T_s = 10\ ms$. We found that the best fit is obtained by an ARMAX model, which has the form

$$A(z)y(t) = B(z)u(t-nk) + C(z)e(t). \tag{3.41}$$

Since we know that the structure of the transfer function is

$$W_2(s) = \frac{Q_2(s)}{T_2(s)} = \frac{1}{s(s\Theta_2 + \sigma_2)} \tag{3.42}$$

where $Q_2(s)$ and $T_2(s)$ are the Laplace transform of the angle $q_2(t)$ and the torque $\tau_2(t)$ respectively, and Θ_2 is the moment of inertia of the second link computed for the axis of rotation, not for the center of mass. Since there has to be an integrator in the system, we let the integral of the torque be the input of the system, and after rearranging we arrive at

$$W_2^*(s) = \frac{Q_2(s)}{T_2(s)\frac{1}{s}} = \frac{1}{s\Theta_2 + \sigma_2} = \frac{K_2}{sT_2 + 1}. \tag{3.43}$$

Using the fact that the mapping between continuous poles and discrete poles is bijective the order of the polynomial $A(z)$ is known to be 1. Unfortunately the same property does not apply to the zeros of the system, but as a rule of thumb the order of $B(z)$ and $C(z)$ is chosen to be the same as $A(z)$.

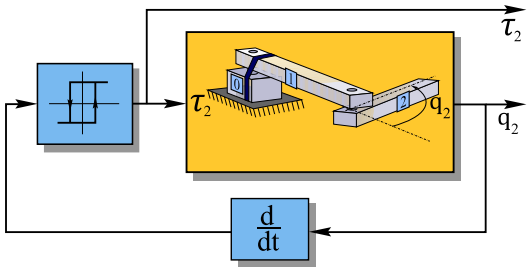

Fig. 32. The concept of the relay identification method presented in a block diagram

The identification thus results in a system with one pole. This discrete pole can be easily transformed into a continuous one by $s_p = \frac{\ln(z_p)}{T_s}$. The time constant is $T_2 = \frac{1}{-s_p}$. The DC gain or K_2 is obtained

by substituting ones in places of the variable z, such that $K_2 = \frac{B(z=1)}{A(z=1)}$. Then by using …. we get the following parameters

$$\Theta_2 = \frac{T_2}{K_2}, \quad \sigma_2 = \frac{1}{K_2}. \tag{3.44}$$

The same procedure is carried out with the first joint coordinate q_1, the obvious difference is that Θ_1 is such that it also contains the rotational inertia of the second link fixed to the first one as depicted in Fig. 31. In the mathematical model, we used the inertias computed at the center of masses, so we need to express them as such. Using Steiner's theorem, it is clear that

$$\begin{aligned} \Theta_2 &= \Theta_{2,z} + m_2 l_{2cx}^2 \\ \Theta_1 &= \Theta_{1,z} + m_1\left(l_{1cx}^2 + l_{1cy}^2\right) + \Theta_{2,z} + m_2\left(l_1 - l_{2cx}\right)^2 \end{aligned} \tag{3.45}$$

where the moments of inertias $\Theta_{1,z}$, $\Theta_{2,z}$ can easily be calculated.

Now we are in position to choose the uncertain parameters and their parameter space. Since the usual task that a robotic arm has to carry out is to move an object in space which is fixed to the last link (due to it being a tool used in manufacturing, painting, welding etc. or grabbing an object with the help of its grippers), we consider the mass, center of mass and moment of inertia of the second link uncertain, as well as the coefficients of viscous friction for both joints.

Let m_{load} denote the mass of the load moved around by the manipulator. Then the new center of mass of the second link reads

$$l_{2cx,load} = \frac{m_2 l_{2cx} + m_{load} l_2}{m_2 + m_{load}}. \tag{3.46}$$

The physical robotic arm at the department is built by a former student, the geometrical parameters and masses have been determined from his thesis. These are given in Table 2.

Table 2. The parameters which are used in the identification

Parameter [unit]	Value
$m_1[kg]$	1.815
$m_2[kg]$	0.735
$l_1[m]$	0.44
$l_2[m]$	0.28
$l_{1cx}[m]$	0.265
$l_{1cy}[m]$	0.009
$l_{2cx}[m]$	0.082

The identification described before is carried out using these values. The results for the moments of inertia and viscous friction coefficients are

$$\Theta_{1,z} = 0.102 \text{ kgm}^2$$

$$\Theta_{2,z} = 0.0033 \text{ kgm}^2$$

$$\sigma_1 = 0.066 \ \ Nms/rad$$

$$\sigma_2 = 0.012 \ \ Nms/rad$$

Now consider that the maximal payload has mass $m_{load} = 1$ kg . This gives us a range for the uncertain parameters, which the robustifying controllers need to cover. For the viscous friction coefficients we take an arbitrary range. Table 3 summarizes the uncertain parameters and their range.

Table 3. The uncertain parameters for the robotic arm

Parameter [unit]	Nominal value	Min. value	Max. value
$m_2 [kg]$	0.735	0.6	1.7
$l_{2cx} [m]$	0.082	0.08	0.1
$\Theta_{2,z} [kg/m^2]$	0.0033	0.003	0.032
$\sigma_1 [Nms/rad]$	0.066	0.02	0.1
$\sigma_2 [Nms/rad]$	0.012	0.005	0.05

A grid with $3^5 = 243$ vertices has been defined to cover the uncertain parameter ranges. The numerical linearization is carried out on the closed loop system containing the nonlinear dynamics with parameters from each vertex and the linearizing feedback with nominal parameters. This process results in a set of linear systems. The weighting functions $W_1(s)$ and $W_2(s)$ satisfying (3.3) for this example read

$$W_1(s) = \begin{bmatrix} \dfrac{0.6019s^2 + 10.53s + 52.51}{s^2 + 24.55s + 177.7} & 0 \\ 0 & \dfrac{1.32s^2 + 16.97s + 28.39}{s^2 + 11.69s + 12.1} \end{bmatrix}$$

$$W_2(s) = \begin{bmatrix} \dfrac{1.322s^2 + 7.159s + 6.044}{s^2 + 9.395s + 25.99} & 0 \\ 0 & \dfrac{0.9935s^2 + 13.24s + 54.03}{s^2 + 12.47s + 23.24} \end{bmatrix} \qquad (3.47)$$

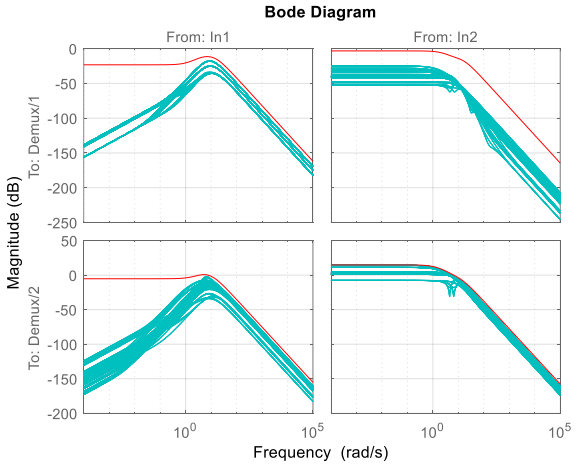

Fig. 33. The uncertainty weighting functions for the robotic arm

Using the interconnected structure defined in (3.4) the synthesis of the robust controllers can be done using the hinfsyn() command in Matlab. To test the performance of the closed loop system a circular reference path is defined in the workspace in the form

$$\begin{aligned} x_d &= 0.6 + r\sin(\omega t) \\ y_d &= r\cos(\omega t) \end{aligned} \tag{3.48}$$

where the subscript d denotes that it is the desired trajectory, with $r = 0.16\ m$ and $\omega = 3\ rad/s$. The derivatives of the trajectory can easily be calculated. The system starts from the initial state $x_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$ so an initial polynomial reference trajectory is constructed that leads the system to the beginning of the circular path. This initial path takes $T_0 = 3\ s$, the boundary conditions are

$$\begin{aligned} &q_{i,d}(0) = 0 && \dot{q}_{i,d}(0) = 0 && \ddot{q}_{i,d}(0) = 0 \\ &q_{i,d}(T_0) = q_{i,circular}(0) && \dot{q}_{i,d}(T_0) = \dot{q}_{i,circular}(0) && \ddot{q}_{i,d}(T_0) = \ddot{q}_{i,circular}(0) \end{aligned} \tag{3.49}$$

for $i = 1,2$, where the subscript 'circular' denotes that the joint variable's value corresponds to the circular portion of the reference trajectory at time 0. To satisfy all boundary conditions the function of the desired joint variables takes the form

$$q_{i,d}(t) = a_{0,i}t^5 + a_{1,i}t^4 + a_{2,i}t^3 + a_{3,i}t^2 + a_{4,i}t + a_{5,i} \tag{3.50}$$

where $a_{j,i} \in \mathbb{R}$. The derivatives are easily obtained

$$\dot{q}_{i,d}(t) = 5a_{0,i}t^4 + 4a_{1,i}t^3 + 3a_{2,i}t^2 + 2a_{3,i}t + a_{4,i} \quad \ddot{q}_{i,d}(t) = 20a_{0,i}t^3 + 12a_{1,i}t^2 + 6a_{2,i}t + 2a_{3,i} \tag{3.51}$$

Using the boundary conditions (3.48), the three coefficients $a_{3,i}$, $a_{4,i}$ and $a_{5,i}$ are set to 0. The remaining three coefficients are obtained by solving a simple linear system of equations.

$$\begin{bmatrix} T_0^5 & T_0^4 & T_0^3 \\ 5T_0^4 & 4T_0^3 & 3T_0^2 \\ 20T_0^3 & 12T_0^2 & 6T_0 \end{bmatrix} \begin{bmatrix} a_{0.i} \\ a_{1.i} \\ a_{2.i} \end{bmatrix} = \begin{bmatrix} q_{i,circular}(0) \\ \dot{q}_{i,circular}(0) \\ \ddot{q}_{i,circular}(0) \end{bmatrix} \tag{3.52}$$

This provides as all the necessary coefficients to construct the initial reference signal in the joint coordinates. The circular reference trajectory however have to be transformed from the workspace to the joint space defined by q_1 and q_2 which is called the problem of inverse kinematics. In this simple example there exist an exact analytical computational method to determine the inverse kinematics. Since

$$\begin{aligned} x_d &= l_1\cos q_1 + l_2\cos(q_1 + q_2) \\ y_d &= l_1\sin q_1 + l_2\sin(q_1 + q_2) \end{aligned} \tag{3.53}$$

To determine the values of q_1 consider the following operations:

$$\begin{aligned} l_2\cos(q_1 + q_2) &= x_d - l_1\cos q_1 \\ l_2\sin(q_1 + q_2) &= y_d - l_1\sin q_1 \end{aligned} \tag{3.54}$$

Squaring both equations, summing them then simplifying and rearranging yields

$$2l_1x_d\cos q_1 + 2l_1y_d\sin q_1 = x_d^2 + y_d^2 + l_1^2 - l_2^2 \tag{3.55}$$

which is an equation of the form $A\cos\alpha + B\sin\beta = D$. Which has solutions

$$\sin\alpha = \frac{DB + \delta A\sqrt{A^2 + B^2 - D^2}}{A^2 + B^2}, \quad \cos\alpha = \frac{DA + \gamma B\sqrt{A^2 + B^2 - D^2}}{A^2 + B^2} \tag{3.56}$$

where the pair (δ, γ) can either be $(1,-1)$ or $(-1,1)$ since the equality $\sin^2\alpha + \cos^2\alpha = 1$ has to be satisfied. This provides us two pairs of $(\sin\alpha, \cos\alpha)$ where each pair defines exactly one angle α, which is obtained by $\alpha = \text{atan}\,2(\sin\alpha, \cos\alpha)$ if the workspace condition $A^2 + B^2 - D^2 \geq 0$ is satisfied. If not, it means that the desired point cannot be reached by the robotic arm, since it is not in its workspace. In our case out of the two possible solutions of q_1 we choose the one that is closer to the previously defined q_1. The other joint variable is obtained by considering

$$\sin(q_1 + q_2) = \frac{y_d - l_1\sin q_1}{l_2}, \quad \cos(q_1 + q_2) = \frac{x_d - l_1\cos q_1}{l_2} \tag{3.57}$$

and then proceeding with the calculation

$$q_2 = \text{atan}\,2(\sin(q_1 + q_2), \cos(q_1 + q_2)) - q_1. \tag{3.58}$$

Since these are not proper mathematical formulas in the sense that they are not in closed form, the derivatives must be obtained in a different way. Thankfully, the joint velocities are much easier to compute using the Jacobian of the TCP. Let $r_{d,TCP}$ denote the vector $\begin{bmatrix} x_d & y_d \end{bmatrix}^T$. The derivative of the reference trajectory in the workspace is

$$\dot{r}_{d,TCP} = \frac{\partial r_{d,TCP}}{\partial q}\dot{q} = J_{TCP}\dot{q} \tag{3.59}$$

where J_{TCP} is the Jacobian of the TCP. From this equation the joint velocities are

$$\dot{q} = J_{TCP}^{-1}\dot{r}_{d,TCP} \tag{3.60}$$

if the inverse of the Jacobian exists. If we integrate this expression, we get the joint references. Indeed, in the case where there is no analytical solution for the inverse kinematics, this (numerical) integration is used for the joint reference signals. One more derivative is needed however, given by

$$\ddot{r}_{d,TCP} = \frac{d}{dt}(J_{TCP}\dot{q}) = \frac{dJ_{TCP}}{dt}\dot{q} + J_{TCP}\ddot{q} \tag{3.61}$$

which means that

$$\ddot{q} = J_{TCP}^{-1}\left(\ddot{r}_{d,TCP} - \frac{dJ_{TCP}}{dt}\dot{q}\right). \tag{3.62}$$

The final reference signal then consists of the initial path leading the system to the joint variables corresponding to the beginning of the circular path and the joint variables of the circular path itself.

Before the implementation of the controller on the real robotic arm, the three different control structures have been simulated: one without robustifying controller (referred to as NORC), a variant with a robust serial compensator (referred to as SRC) and a variant with the 2DOF robustifying controller (referred to as 2DOFRC). For each one of them, five simulations have been executed with distinct samples from the uncertain parameter space. For each cycle of simulations each controller architecture is controlling a

plant with the same parameters. The results of the first simulations are shown below. This is the ideal case; joint frictions are neglected except for the linear/viscous part and there are no disturbances at the plant inputs. Fig. 34 show the path of the TCP for each controller. Each one of them gives acceptable results, however the best reference tracking performance is achieved with 2DOFRC.

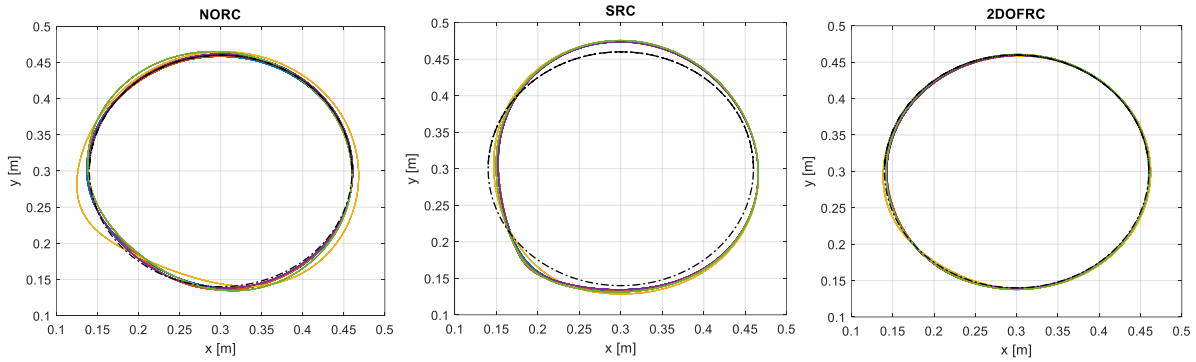

Fig. 34. The TCP paths for the three controller variants

One of the features of robust controllers is their ability to attenuate external disturbances which will inevitably arise in the physical realizations of control systems. Another set of simulations have been exceuted with disturbance torques acting on each joint. This effect is similar to that of the friction at the joints. The results are seen on Fig. 35.

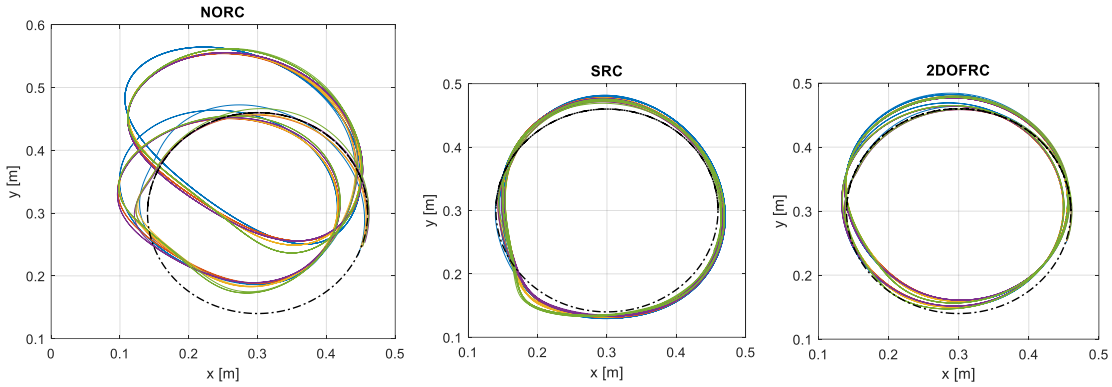

Fig. 35. The TCP paths for the three controller variants with external disturbances

Looking at the path of the TCP it is obvious that the NORC variant gives unsatisfactory performance. Both SRC and 2DOFRC seem to attenuate the disturbances well.

The controllers are tested on the physical system with the help of QUARC, a software solution to implement controllers in real-time designed in Simulink with PCs running non-real-time operating system, e.g. Windows. The Simulink block diagram is evaluated at the sampling rate specified by the user. Our implementation uses a sampling time of 10 ms, which means that whatever the operating system's task is, it is interrupted, and the Simulink diagram is called for one iteration. The solver used is called ode14x in Matlab. The accuracy of lower order solvers was all insufficient and made the closed-loop system containing the robust controllers unstable. As the MathWorks document states this solver uses an implicit function of the state and the state derivative to compute the next state of the system,

$$x(n+1)-x(n)-h\cdot\dot{x}(n+1)=0 \tag{3.63}$$

where $x(n)$ is the state variable and $\dot{x}(n)$ is the derivative of the state variable at the n-th time step, with h as the step size. This is then evaluated by the combination of Newton's method and extrapolation from the current state.

The torques on the joints of the physical system are applied by geared DC motors which are driven by servo drivers. These drivers are set to current control mode, meaning that they output a current proportional to the voltage at their input. Since the torque of a DC motor is proportional to the current of the armature, this basically means that this is a torque control setup. There are special blocks in Simulink allowing the user to interface with the encoders and the servo controller. The incoming angle and outgoing torque signals have to be multiplied by gains determined by, for example, the gear ratio, encoder resolution and the gain of the servo driver.

The reference trajectory during the physical test has the same form as in the simulations, however the radius and speed have been lowered simply for safety reasons, they are $r = 0.08$ m and $\omega = 1.5$ rad/s. The joint angle outputs are shown below in Fig. 36.

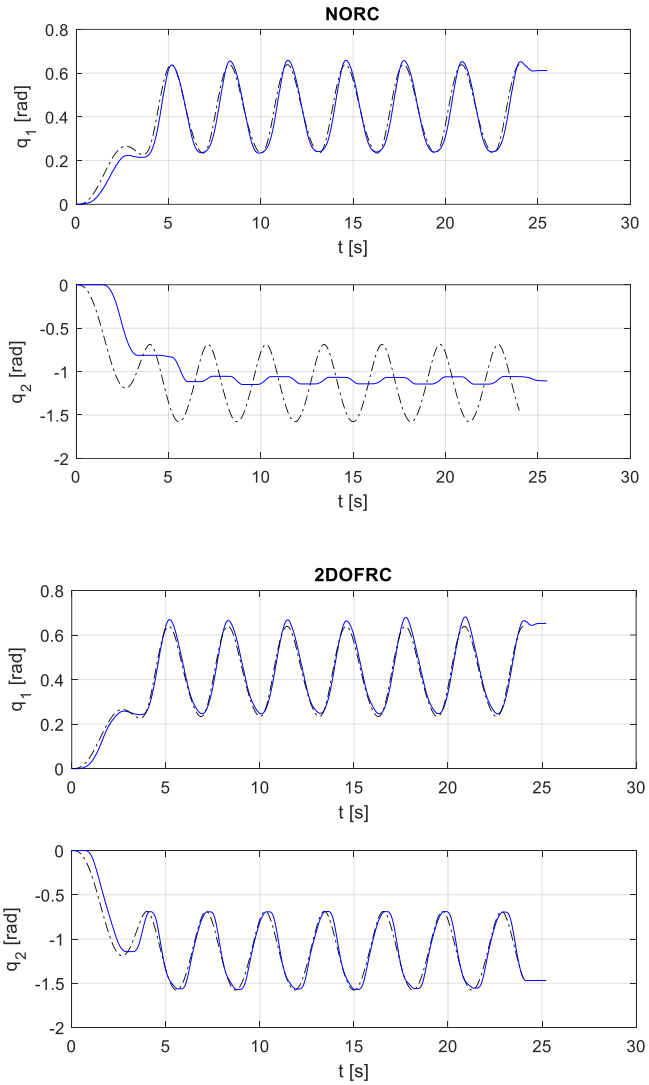

Fig. 36. The joint angles of the physical robotic arm for the non-robust and 2DOF controller variants

This test really shows the advantage of the robust 2DOF controller: as it is clearly seen, the classical computed torque method was unable to attenuate the effect of friction at the second joint, the "plateaus" that show the presence of static friction at low torques and speeds are really prominent. In contrast, the robust controller provides significantly better performance. Fig. 37 shows the same test in the workspace, where the path of the TCP is traced.

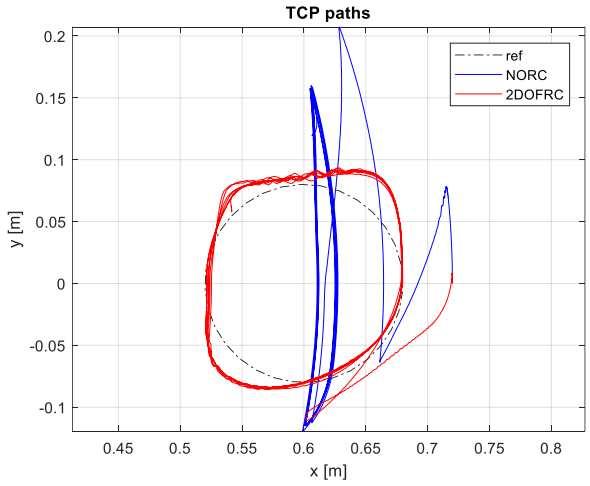

Fig. 37. The TCP path of the physical robotic arm the non-robust and 2DOF controller variants

It is obvious that the robust controller variant has better performance, which is the result of the previously discussed attenuation of the static friction.

3.3 Quadcopter (MIMO)

Next a more complex system is presented as a quadcopter moving through 3D space [12]. This is also an underactuated system, meaning it has less actuators than degrees of freedom. To create the forces and torques acting on the vehicle, the four propellers need to rotate in specific directions and speeds. For simplicity we omit these considerations and the actuator dynamics since they can easily be implemented on a real quadcopter and not relevant in the proposed control architecture. This means that the inputs of the system are the torques and forces themselves.

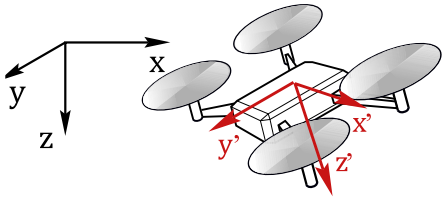

Fig. 38. The coordinate frames used to determine the orientation of the quadcopter

To derive the mathematical model two separate coordinate systems are considered: one is inertial or "stationary" and one is fixed to the quadcopter. The inertial frame is conventionally chosen as the North-East-Down coordinate system, where the z axis points downwards. The orientation of the vehicle is represented by the coordinate frame fixed to it at the center of mass and is described as a rotation matrix relative to the inertial frame. The rotation matrix is expressed with ZYX Euler angles. The matrix describing the orientation is thus the product of the three elementary rotational matrices

$$R_x(\phi)=\begin{bmatrix}1 & 0 & 0\\ 0 & c_\phi & -s_\phi\\ 0 & s_\phi & c_\phi\end{bmatrix} \qquad R_y(\theta)=\begin{bmatrix}c_\theta & 0 & s_\theta\\ 0 & 1 & 0\\ -s_\theta & 0 & c_\theta\end{bmatrix} \qquad R_z(\psi)=\begin{bmatrix}c_\psi & -s_\psi & 0\\ s_\psi & c_\psi & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3.64}$$

and is denoted by $R_{zyx}(\phi,\theta,\psi)=R_z(\psi)\cdot R_y(\theta)\cdot R_x(\phi)\in SO(3)$. The rotation matrix thus has the form

$$R_{zyx}(\phi,\theta,\psi)=\begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}. \tag{3.65}$$

The mathematical model is derived using Newton and Euler equations for a rigid body in space. Let $\Theta=\begin{bmatrix}\psi & \theta & \varphi\end{bmatrix}^T$ denote the vector for the Euler angles. The quadcopter's coordinates and linear velocities are $r=\begin{bmatrix}x & y & z\end{bmatrix}^T$ and $v=\begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix}^T$ respectively. The angular velocity in the inertial frame is the derivative of Θ with respect to time and is denoted by $\omega=\dot{\Theta}$. The linear and angular velocities in the body frame are denoted by $v_b=\begin{bmatrix}u & v & w\end{bmatrix}^T$ and $\omega_b=\begin{bmatrix}p & q & r\end{bmatrix}^T$. The transformations of these velocities between the inertial and body frame are given in

$$\begin{bmatrix} v_b \\ \omega_b \end{bmatrix}=\begin{bmatrix} R_{zyx}^T(\Theta) & 0_{3\times3} \\ 0_{3\times3} & T^{-1}(\Theta) \end{bmatrix}\begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.66}$$

where $T(\psi,\theta,\varphi)$ can be derived using the fact that $[\omega\times]=\dot{R}\cdot R^T$ and has the form

$$T(\psi,\theta,\varphi)=\begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \tag{3.67}$$

Its inverse exists in the neighborhood of the origin, in other words in the neighborhood of $\Theta=\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$. Now the Newton Euler equations can be written in the body frame as

$$\begin{bmatrix} mI_3 & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}\begin{bmatrix} \dot{v}_b \\ \dot{\omega}_b \end{bmatrix}=\begin{bmatrix} -m[\omega_b\times] & 0_{3\times3} \\ 0_{3\times3} & -[\omega_b\times]J \end{bmatrix}\begin{bmatrix} v_b \\ \omega_b \end{bmatrix}+\begin{bmatrix} f_b \\ \tau_b \end{bmatrix} \tag{3.68}$$

with m as the mass, $f_b=\begin{bmatrix}f_x & f_y & f_z\end{bmatrix}^T$ and $\tau_b=\begin{bmatrix}\tau_x & \tau_y & \tau_z\end{bmatrix}^T$ as the total force and torque applied in the body frame and

$$J=\begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{3.69}$$

as the matrix of the moment of inertia of the quadcopter, which is constant in the frame fixed to the body. The external forces are

$$f_b = mgR_{zyx}^T\hat{e}_{z,b} - f_t\hat{e}_{z,b} + f_d\,. \tag{3.70}$$

The unity vector $\hat{e}_{z,b}\in\mathbb{R}^3$ points in the direction of the z axis of the body frame, $f_t\in\mathbb{R}$ is the amplitude of the force produced by the propellers lifting the quadcopter (also known as the control force) and $f_d\in\mathbb{R}^3$ is the vector of the disturbing forces (e.g. wind). The external moments have the form

$$m_b = \tau_b - g_b + \tau_d \tag{3.71}$$

where $\tau_b \in \mathbb{R}^3$ is the vector of the control torques, $\tau_d \in \mathbb{R}^3$ are the disturbing torques and $g_b \in \mathbb{R}^3$ are the gyroscopic moments caused by the rotors. The gyroscopic moments are usually negligible in a typical application, we will consider them disturbances from now on.

The state space representation of the whole system is given below. The vector of the states is $x = \begin{bmatrix} x & y & z & \psi & \theta & \phi & \dot{x} & \dot{y} & \dot{z} & p & q & r \end{bmatrix}^T$. Here, a simplification is made by setting $\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T = \begin{bmatrix} p & q & r \end{bmatrix}^T$ which holds true for small angles of movements [13].

The state space equations are in the form

$$\dot{x} = f(x) + \sum_{i=1}^{4} g_i(x) u_i \tag{3.72}$$

where

$$f(x) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ q \cdot s_\phi / c_\theta + r \cdot c_\phi / c_\theta \\ q \cdot c_\phi - r \cdot s_\phi \\ p + q \cdot s_\phi t_\theta + r \cdot c_\phi t_\theta \\ 0 \\ 0 \\ g \\ (J_y - J_z) \cdot qr / J_x \\ (J_z - J_x) \cdot pr / J_y \\ (J_x - J_y) \cdot pq / J_z \end{bmatrix}, \qquad g_1(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/m(s_\phi s_\psi + c_\phi c_\psi s_\theta) \\ -1/m(s_\phi c_\psi - c_\phi s_\psi s_\theta) \\ -1/m(c_\phi c_\theta) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.73}$$

$$g_2(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/J_x & 0 & 0 \end{bmatrix}^T$$
$$g_3(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/J_y & 0 \end{bmatrix}^T$$
$$g_4(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/J_z \end{bmatrix}^T$$

The controlled outputs of the quadcopter is the absolute position and the yaw angle ψ. This results in an output function in the form

$$y = h(x) = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T . \tag{3.74}$$

As previously discussed, we seek a static feedback using the states as

$$u = \alpha(x) + \beta(x) v \tag{3.75}$$

where v is the new input. The first step is to define the vector of the relative degree. However we find that the relative degree does not exists, since the matrix (2.63) of this system is

$$A(x)=\begin{bmatrix} -1/m\left(s_\phi s_\psi + c_\phi c_\psi s_\theta\right) & 0 & 0 & 0 \\ -1/m\left(s_\phi c_\psi - c_\phi s_\psi s_\theta\right) & 0 & 0 & 0 \\ -1/m\left(c_\phi c_\theta\right) & 0 & 0 & 0 \\ 0 & 0 & s_\phi/\left(J_y c_\theta\right) & c_\phi/\left(J_z c_\theta\right) \end{bmatrix} \tag{3.76}$$

which is clearly singular. This happens because the input u_1 affects $\ddot{y}_1, \ddot{y}_2$ and $\ddot{y}_3$, while none of them is affected by the rest of the inputs. The solution to these kinds of problem was presented earlier in the chapter discussing the theoretical background of MIMO exact linearization, that is to delay u_1 in such a way that it appears later in (2.63). This is done by applying a double integrator at the input of the original system, which delays u_1 by two differentiations.

$$\begin{aligned} u_1 &= \zeta_1 \\ \dot{\zeta}_1 &= \zeta_2 \\ \dot{\zeta}_2 &= \bar{u}_1 \end{aligned} \tag{3.77}$$

for a consistent notation we also set $u_i = \bar{u}_i$, i=2,3,4 as the other inputs. The resulting augmented system's state space model uses the state variable vector

$$\bar{x}=\begin{bmatrix} x & y & z & \psi & \theta & \phi & \dot{x} & \dot{y} & \dot{z} & \zeta_1 & \zeta_2 & p & q & r \end{bmatrix}^T \tag{3.78}$$

and has the form

$$\dot{\bar{x}} = \bar{f}\left(\bar{x}\right) + \sum_{i=1}^{4} \bar{g}_i\left(\bar{x}\right)\bar{u}_i \tag{3.79}$$

where

$$\bar{f}\left(\bar{x}\right)=\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ q\cdot s_\phi/c_\theta + r\cdot c_\phi/c_\theta \\ q\cdot c_\phi - r\cdot s_\phi \\ p + q\cdot s_\phi t_\theta + r\cdot c_\phi t_\theta \\ -1/m\left(s_\phi s_\psi + c_\phi c_\psi s_\theta\right)\zeta_1 \\ -1/m\left(s_\phi c_\psi - c_\phi s_\psi s_\theta\right)\zeta_1 \\ -1/m\left(c_\phi c_\theta\right)\zeta_1 \\ \zeta_2 \\ 0 \\ \left(J_y - J_z\right)\cdot qr/J_x \\ \left(J_z - J_x\right)\cdot pr/J_y \\ \left(J_x - J_y\right)\cdot pq/J_z \end{bmatrix} \tag{3.80}$$

$$g_1(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]^T$$
$$g_2(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1/J_x \quad 0 \quad 0]^T$$
$$g_3(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1/J_y \quad 0]^T$$
$$g_4(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1/J_z]^T$$

It is straightforward to show that this augmented system has a relative degree vector of $[r_1 \quad r_2 \quad r_3 \quad r_4] = [4 \quad 4 \quad 4 \quad 2]$ and that

$$\left[y_1^{(r_1)} \quad y_2^{(r_2)} \quad y_3^{(r_3)} \quad y_4^{(r_4)} \right]^T = b(\bar{x}) + A(\bar{x})\bar{u} = v \tag{3.81}$$

where

$$A(\bar{x}) = \begin{bmatrix} \frac{-(s_\phi s_\psi + c_\phi c_\psi s_\theta)}{m} & \frac{-\zeta_1(c_\phi s_\psi - s_\phi c_\psi s_\theta)}{J_x m} & \frac{-\zeta_1 c_\psi c_\theta}{J_y m} & 0 \\ \frac{-(s_\phi c_\psi - c_\phi s_\psi s_\theta)}{m} & \frac{-\zeta_1(c_\phi c_\psi + s_\phi s_\psi s_\theta)}{J_x m} & \frac{\zeta_1 s_\psi c_\theta}{J_y m} & 0 \\ \frac{-c_\phi c_\theta}{m} & \frac{\zeta_1 s_\phi c_\theta}{J_x m} & \frac{\zeta_1 s_\theta}{J_y m} & 0 \\ 0 & 0 & \frac{s_\phi}{J_y c_\theta} & \frac{c_\phi}{J_z c_\theta} \end{bmatrix} \tag{3.82}$$

which is not singular in the neighborhood where the system is usually operated. The linearizing feedback is then

$$\bar{u} = A^{-1}(\bar{x})[-b(\bar{x}) + v] = \alpha(\bar{x}) + \beta(\bar{x})v\,. \tag{3.83}$$

The block diagram of the system with dynamic linearizing feedback is seen in Fig. 39. The coordinate transformation $z = \Phi(\bar{x})$ is given by

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \end{bmatrix} = \begin{bmatrix} h_1(x) \\ L_f h_1(x) \\ L_f^2 h_1(x) \\ L_f^3 h_1(x) \\ h_2(x) \\ L_f h_2(x) \\ L_f^2 h_2(x) \\ L_f^3 h_2(x) \\ h_3(x) \\ L_f h_3(x) \\ L_f^2 h_3(x) \\ L_f^3 h_3(x) \\ h_4(x) \\ L_f h_4(x) \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ x^{(3)} \\ y \\ \dot{y} \\ \ddot{y} \\ y^{(3)} \\ z \\ \dot{z} \\ \ddot{z} \\ z^{(3)} \\ \psi \\ \dot{\psi} \end{bmatrix} \tag{3.84}$$

which if applied together with the linearizing feedback transform the system into a linear one. This MIMO linear system can be thought of as 4 different SISO linear systems, since we achieved non-interacting control using this algorithm. These systems in state space form have the following matrices

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad b_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad c_i^T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \text{i=1,2,3} \tag{3.85}$$

and

$$A_4 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad b_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c_4^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{3.86}$$

The state vectors are

$$\begin{aligned} x_1 &= \begin{bmatrix} x & \dot{x} & \ddot{x} & x^{(3)} \end{bmatrix}^T = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 \end{bmatrix}^T \\ x_2 &= \begin{bmatrix} y & \dot{y} & \ddot{y} & y^{(3)} \end{bmatrix}^T = \begin{bmatrix} z_5 & z_6 & z_7 & z_8 \end{bmatrix}^T \\ x_3 &= \begin{bmatrix} z & \dot{z} & \ddot{z} & z^{(3)} \end{bmatrix}^T = \begin{bmatrix} z_9 & z_{10} & z_{11} & z_{12} \end{bmatrix}^T \\ x_4 &= \begin{bmatrix} \psi & \dot{\psi} \end{bmatrix}^T = \begin{bmatrix} z_{13} & z_{14} \end{bmatrix}^T \end{aligned} \tag{3.87}$$

The MIMO system thus has the form

$$\dot{z} = \begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & A_3 & 0 \\ 0 & 0 & 0 & A_4 \end{bmatrix} z + \begin{bmatrix} b_1 & 0 & 0 & 0 \\ 0 & b_2 & 0 & 0 \\ 0 & 0 & b_3 & 0 \\ 0 & 0 & 0 & b_4 \end{bmatrix} v = Az + Bv$$
$$y = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & c_4 \end{bmatrix} z = Cz \tag{3.88}$$

Our now is to construct a state feedback to this linear system, where the poles are moved to a more favorable location, since the type of robust control design we will consider cannot be executed for nominal systems containing poles at the origin.

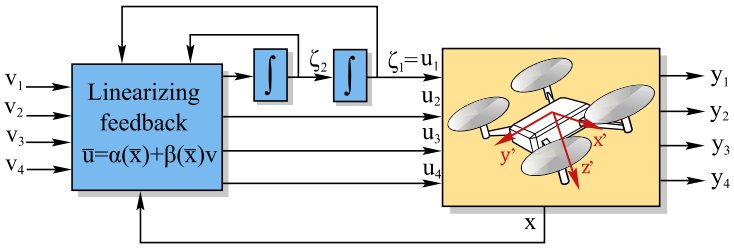

Fig. 39. The exact linearizing control of the quadcopter

For simplicity we want all the poles to be at $s = -2$. We found that for sufficient performance of the proposed control algorithm (and for the non-robust variant) integral control is also needed. The block diagram of this controller structure is shown in Fig. 40. This means that for each SISO subsystem the state vector is extended by another variable, $z_I = \int Cz\ dt$.

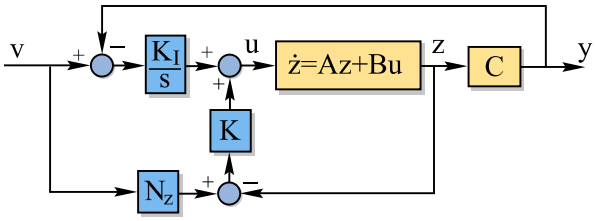

Fig. 40. Integral state feedback control

The state feedback in question need to be designed for the extended linear system containing z_I in the state vector. This is given by

$$\begin{bmatrix} \dot{z} \\ \dot{z}_I \end{bmatrix} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} \begin{bmatrix} z \\ z_I \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u = \tilde{A}\tilde{z} + \tilde{B}u\,. \tag{3.89}$$

Considering SISO subsystems the synthesis of the gains is obtained by the Ackermann-formula.

$$\begin{bmatrix} K_i & K_{i,I} \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} M_{i,C}^{-1} \varphi_{i,C}(A_i) \quad \forall \text{i=1,2,3,4}\,. \tag{3.90}$$

The input for the linearized MIMO system is thus

$$u = \begin{bmatrix} K_1 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 \\ 0 & 0 & K_3 & 0 \\ 0 & 0 & 0 & K_4 \end{bmatrix} \left(N_z v - z \right) + \begin{bmatrix} K_{1,I} & 0 & 0 & 0 \\ 0 & K_{2.I} & 0 & 0 \\ 0 & 0 & K_{3,I} & 0 \\ 0 & 0 & 0 & K_{4,I} \end{bmatrix} \int \left(v - y \right) dt \tag{3.91}$$

where the constant matrix N_z is used to match the dimension of v to z.

$$N_z = \begin{bmatrix} N_{1,z} & 0 & 0 & 0 \\ 0 & N_{2,z} & 0 & 0 \\ 0 & 0 & N_{3,z} & 0 \\ 0 & 0 & 0 & N_{4,z} \end{bmatrix}, \quad N_{1,z} = N_{2,z} = N_{3,z} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad N_{4,z} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{3.92}$$

Finally, we are ready to design the robustifying controller. The uncertain parameters are the moments of inertias and the mass of the quadcopter. This consideration is useful in for example delivery drones that need to carry payloads of varying masses. In this example the chosen values for the nominal parameters and their uncertainty range is given in Table 4.

Fig. 41. Delivery drone with payload [14]

The same procedure is used as described in the previous examples, namely covering the uncertain parameter range with a sufficiently fine grid, and linearizing the system with parameter values at each vertex to obtain the uncertainty shaping transfer matrices.

Table 4. The uncertain parameters of the quadcopter

Parameter [unit]	Nominal value	Min. value	Max. value
$m[kg]$	1	0.8	1.6
$J_x\left[kg/m^2\right]$	0.01	0.006	0.016
$J_y\left[kg/m^2\right]$	0.01	0.006	0.016
$J_z\left[kg/m^2\right]$	0.03	0.018	0.048

The resulting linearized models confirm that the uncertainty weighting function $W_2(s)$ can be set to I. This results in

$$W_1(s) = diag\left\{\begin{array}{c}\dfrac{1.09s^2 + 0.0058s + 1.56\cdot 10^{-4}}{s^2 + 0.8371s + 2.284},\\ \dfrac{1.085s^2 + 0.005s + 1.52\cdot 10^{-4}}{s^2 + 0.8697s + 2.279},\\ \dfrac{0.478s^2 + 0.0016s + 6.57\cdot 10^{-5}}{s^2 + 0.3597s + 1.427},\\ \dfrac{0.68s^2 + 0.0146s + 3.03\cdot 10^{-4}}{s^2 + 8.908s + 5.127}\end{array}\right\} \tag{3.93}$$

This uncertainty weighting transfer function matrix is depicted in Fig. 42, where the linearized systems sampled at the grid vertices are drawn in cyan and $W_1(s)$ is drawn in red.

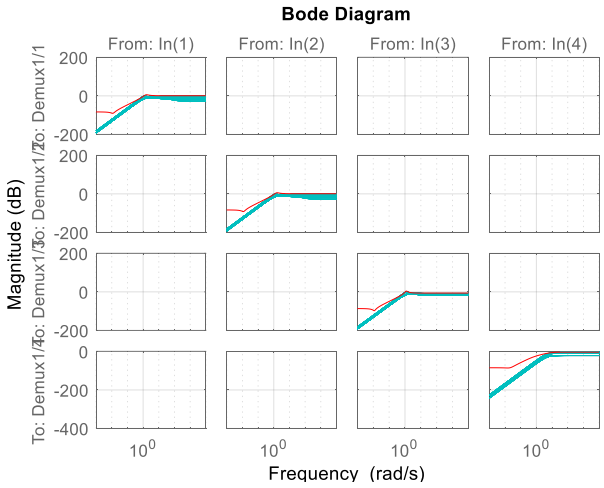

Fig. 42. The uncertainty weighting functions for the quadcopter

Despite the resulting uncertainty weighting transfer function matrix from linearization we found that if the synthesis is carried out with (3.93) the resulting robust controller performs rather poorly if the mass differs from the nominal value. This happens at the third output, the altitude of the aircraft. This is probably caused by gravity, which means that there are significant uncertainties at lower frequencies, which does not appear during the numerical linearization. To solve it the third entry of (3.93) was changed to a unity gain. This phenomenon needs to be further investigated.

The nominal transfer function has the form $G_0(s) = diag\{g_1, g_2, g_3, g_4\}$, where

$$\begin{aligned} g_i(s) &= \frac{80s + 32}{s^5 + 10s^4 + 40s^3 + 80s^2 + 80s + 32}, \quad \text{i=1,2,3} \\ g_4(s) &= \frac{75s + 125}{s^3 + 15s^2 + 75s + 125} \end{aligned} \tag{3.94}$$

The reference model is the same as the nominal system, since our aim with the robust controller is to enforce nominal behavior of the closed loop system. The resulting controller is of order 108, however during simulations it was reduced to 50, where adequate performance was achieved even with this rather drastic reduction.

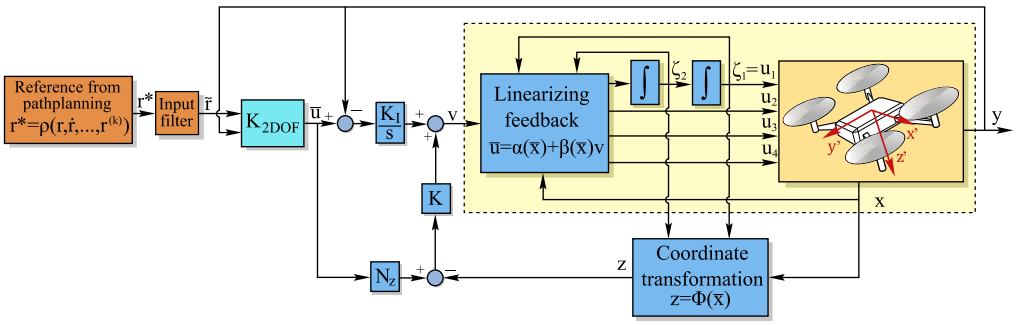

Fig. 43. The proposed controller solution for the quadcopter

The input reference is such, that for x, y and z the derivatives up to the fifth order is needed, while for the yaw angle ψ up to the third order. The following expressions show, how to formulate the input signals. First let us look at x, y and z. Taking the linear combination of the derivatives as

$$r^* = r^{(5)} + 10 \cdot r^{(4)} + 40 \cdot r^{(3)} + 80 \cdot \ddot{r} + 80 \cdot \dot{r} + 32 \cdot r \tag{3.95}$$

and taking its Laplace transform $r^*(t) \to R^*(s)$ through a filter we obtain

$$\tilde{R}(s) = \frac{1}{80s + 32} R^*(s) = \frac{s^5 + 10s^4 + 40s^3 + 80s^2 + 80s + 32}{80s + 32}. \tag{3.96}$$

Applying this modified signal to the nominal system yields

$$Y(s) = g_1(s)\tilde{R}(s) = R(s) \tag{3.97}$$

which means that the tracking error is decaying during operation. The same process is carried out for all the reference signals.

First the controller was tested where there were no external disturbances applied at the inputs. At first, the controller design did not contained an integrator which meant that there was a large, constant error at the output $y_3 = z$ (which is the altitude of the aircraft), if there was a difference in the nominal and actual mass. That's why the previously detailed design with the integrator was applied. The outputs of the non-robust variant are depicted in Fig. 45. It performs relatively well, however as it is seen in Fig. 46 this can be improved with the 2DOF robust design. Here it is clear that the tracking error is less than for the non-robust case.

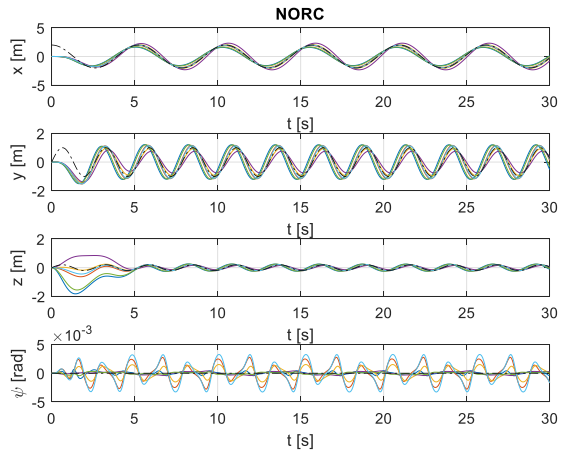

Fig. 44. The outputs for the quadcopter with the non-robust controller

During experimentation it became clear that generally speaking, the faster the reference signal is changing the difference between the two variants becomes more significant.

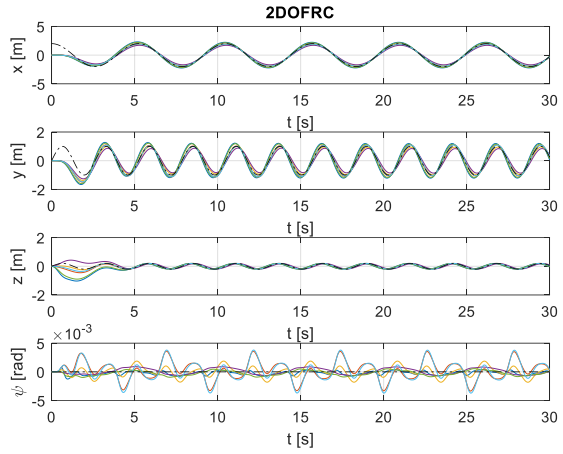

Fig. 45. The outputs for the quadcopter with the proposed 2DOF robust controller

Next a series of simulations were executed with external disturbances. No significant differences were present, which may be attributed to the integral action present in the control law, with the exception of the output $y_3 = z$, where the robust controller had better disturbance attenuation.

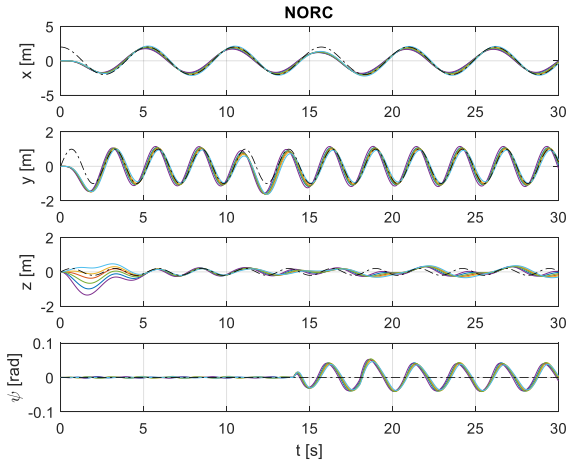

Fig. 46. The outputs for the quadcopter with the non-robust controller with external disturbances

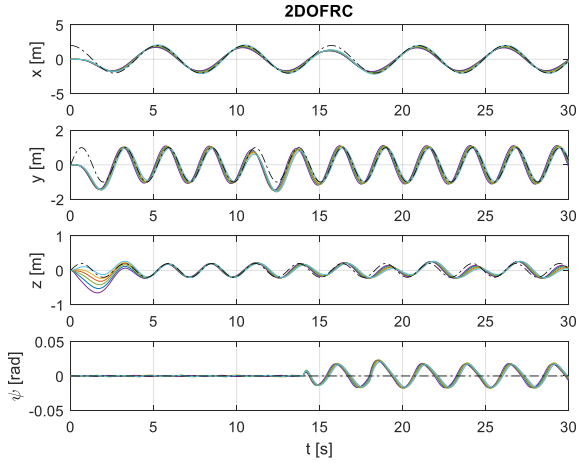

Fig. 47. The outputs for the quadcopter with the proposed 2DOF controller with external disturbances

In conclusion the performance of the closed loop system was improved with the robustifying controller, however it was not as significant as with the other examples. This may be further improved if other performance weighting functions are chosen. We could also take into consideration the external disturbances with adequate weighting functions, which was omitted in this thesis.

4 Conclusion

This report studied at a novel control architecture which joins two state-of-the-art techniques, namely nonlinear feedback linearization and robust linear control (the related background has also been presented for self-containedness). The proposed method is a cascaded structure with the linearizing feedback inside an inner loop and the robust (or robustifying) feedback as an outer loop. A two-degrees-of-freedom controller realizes the robustifying feedback to ensure good stability and performance despite the parameter mismatch between the plant and the linearizing inner feedback. The design workflow to determine the robustifying controller was detailed. It is based on the sufficiently smooth sampling of the uncertain parameter space. Several examples were presented (ranging from simple SISO systems to complex, MIMO dynamics) to demonstrate the usefulness of the technique.

For one specific system, namely the two-degrees-of-freedom robotic arm, the algorithm was implemented in real-time and tested on the real plant. The uncertain parameters were chosen considering payloads with different masses, and the rest of the parameters were identified from measurement data. The proposed controller solution showed excellent performance in practice, which suggests that such controllers have real application potential. In the future this method will be tested on other mechatronic systems.

5 References

[1] L. Baratchart, M. Chyba and J.-B. Pomet, "A Grobman-Hartman theorem for control systems," *Journal of Dynamics and Differential Equations,* vol. 19, no. 1, pp. 75-107, 2007.

[2] B. Francesco és L. Andrew D., Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Systems, New York: Springer, 2005..

[3] A. Isidori, Nonlinear Control Systems, Springer, 1994.

[4] H. E. Taha, "ENGRMAE 276. Geometric Nonlinear Control," 2019.. [Online]. Available: http://taha.eng.uci.edu/Geometric_Control_Course.html. [Accessed 20. 10. 2020.].

[5] J. Lévine, Analysis and Control of Nonlinear Systems, Springer, 2009..

[6] K. Zhou, Essentials of Robust Control, Prentice Hall, 1999.

[7] P. M. Hof, C. Sherer and P. S. Heuberger, Model-Based Control: Bridging Rigorous Theory and Advanced Technology, Springer, 2009..

[8] D.-W. Gu, P. Petkov and M. M. Konstantinov, Robust Control Design with MATLAB, Springer, 2013.

[9] N. Wang and B. Kiss, "A Method to Robustify Exact Linearization Against Parameter Uncertainty," *International Journal of Control, Automation and Systems,* vol. 17, pp. 2441-2451, 2018.

[10] K. M. Lynch and F. C. Park, Modern Robotics: Mechanics, Planning and Control, Cambridge University Press, 2017.

[11] A. A. G. Siqueira, M. H. Terra, J. Y. Ishihara and T. L. S. Barbeiro, "Underactuated manipulator robot control via H2, H∞, H2/H∞, and μ-synthesis approaches: a comparative study," *Journal of the Brazilian Society of Mechanical Sciences and Engineering,* vol. 31, no. 4, pp. 279-288, 2009.

[12] D. Lee, T. Burg, D. Dawson, D. Shu, B. Xian and E. Tatlicioglu, "Robust Tracking Control of an Underactuated Quadrotor Aerial-Robot Based on a Parametric Uncertain Model," *Systems, Man and Cybernetics, 2009, SMC 2009, IEEE International Conference,* pp. 3187-3192., 2009..

[13] A. Das, K. Subbarao and F. Lewis, "Dynamic Inversion with Zero-Dynamics Stabilisation for Quadrotor Control," *IET Control Theory Applications,* vol. 3., no. 3., pp. 303-314., 2009..

[14] "https://menafn.com/updates/pr/2019-02/G_d53ed61c-5image_story.jpeg," 2020.. [Online]. [Accessed 20. 10. 2020.].