TÉCNICO LISBOA

Thesis Title

Candidate Full Name

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Full Name 1
Dr. Full Name 2

Examination Committee

Chairperson: Prof. Full Name
Supervisor: Prof. Full Name 1 (or 2)
Member of the Committee: Prof. Full Name 3

Month Year

Dedicated to someone special...

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Inserir o resumo em Português aqui com o máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave...

Palavras-chave: palavra-chave1, palavra-chave2,...

Abstract

Due to their inherent non-linear dynamics, designing control systems for both rotary and fixed wing aircraft is a non-trivial task, where using linear control approaches often reveal to be inaccurate. Feedback linearization is an approach to non-linear control design that algebraically transforms a non-linear system onto a linear one. From the linearised system linear control techniques can be used. However this approach, as well as other state-of-the art control systems such as model predictive control, backstepping and gain scheduling require the *a priori* exact mathematical model of the system to be controlled [1]. The lack of such a model leads to errors in the calculation of the model inversion, necessary to implement feedback linearization. Indeed, while feedback linearization control shows good tracking, it has poor disturbance rejection, that ultimately lead to these inversion errors [2]. One solution to this problem that has gained momentum over the last years is the use augmentation algorithms, namely neural networks and other intelligent algorithms, to minimize and cancel this error[3], [4], [5]. This thesis therefore aims to discuss and implement the existing alternatives to augment the precision, robustness and overall performance of feedback linearization control.

Keywords: non-linear control, feedback linearization, neural network, flight control

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Insert your chapter material here...

1.1 Motivation

Define: usefulness of NLI vs robust control and problems of NLI

1.2 Topic Overview

Brief description of the solutions to be studied and how they will be used to improve control

1.3 Objectives

Goals: create systems able to adapt to changes of the airplane's dynamic

1.4 Thesis Outline

Background includes in this order: NLI theory, NLI problematic, neural network theory Implementation: ? Results: ???

Chapter 2

Background

On this chapter the aircraft model used for this work will be described firstly. A theoretical model of the plane's dynamics will be discussed, and implementation detail will be provided on the chapter 3. The control strategy used will then follow, giving an overview of the feedback linearisation approach, as well as its limitations, namely its sensitivity to inversion errors and external interferences. An attempt to solve these limitation will be made, suggesting some solutions and finally describing the chosen methodology for this case.

2.1 Airplane Model

The work made in this thesis was built on top of the work done by H. Escamilla Nuñez and F. Mora Camino on 4D trajectory tracking [6]. The model used in this work of a six degree of freedom transport aircraft will be described in this section.

2.1.1 Frames of reference

The first step before describing the dynamics of a commercial aircraft will be to define the frames of reference used to do so. The first frame of reference, on which 4D trajectories are described, corresponds to the WGS84 frame of reference. A second frame of reference corresponding to the aircraft's body frame will be used to provide its fast rotational dynamics. Lastly all aerodynamic forces will be applied in the axial directions of the wind frame. This frame is aligned to the wind speed vector relative to the airplane, given by both the angle of attack α and the sideslip angle β. For these last two frames of reference, a rotation matrix can be defined from the wind frame to the body frame by

$$R_{WB} = \begin{bmatrix} c_\alpha c_\beta & -c_\alpha s_\beta & -s_\alpha \\ s_\beta & c_\beta & 0 \\ s_\alpha c_\beta & -s_\alpha s_\beta & c_\alpha \end{bmatrix} \tag{2.1}$$

To describe the attitude of the plane Euler angles will also be used, namely $\phi\{-\pi, \pi\}; \theta\{-\frac{\pi}{2}, \frac{\pi}{2}\}; \psi\{-\pi, \pi\}$.

From these angles the rotation matrix from the body to the earth's frame is given by

$$R_{BE} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta c_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2.2}$$

2.1.2 Fast Dynamics

The considered actuators of the aircraft that control its attitude are given by $\delta = [\delta_{ail}\delta_{ele}\delta_{rud}]^T$, each applying a torque along an axis of the body frame. These torques are giver by

$$\begin{bmatrix} L' \\ M \\ N \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) \tag{2.3}$$

where $\bar{c}$ and b represent the wing chord length and span respectively, C_δ and the moment coefficients $[C_l C_m C_n]^T$ are given by

$$C_\delta = \begin{bmatrix} bC_{l\delta_{ail}} & 0 & bC_{l\delta_{rud}} \\ 0 & \bar{c}C_{m\delta_{ele}} & 0 \\ bC_{n\delta_{ail}} & 0 & bC_{n\delta_{rud}} \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \begin{bmatrix} C_{l\beta}\beta + C_{l_p} p\frac{b}{2V_a} + C_{l_r} r\frac{b}{2V_a} \\ C_{m_0} + C_{m_\alpha}\alpha + C_{m_q} q\frac{\bar{c}}{2V_a} \\ C_{n\beta}\beta + C_{n_p} p\frac{b}{2V_a} + C_{n_r} r\frac{b}{2V_a} \end{bmatrix} \tag{2.5}$$

Where p, q, r are the body's angular rates ($\Omega = [p \quad q \quad r]^T$) and V_a is the airspeed. The method of obtaining of the coefficients of equation 2.5 will be provided in the chapter to follow. Having defined the torques applied to the aircraft the rotational dynamics equation can now be stated as per [6], I being the aircraft's inertial matrix.

$$\dot{\Omega} = I^{-1} M_{ext} - I^{-1}\Omega \times (I\Omega) \tag{2.6a}$$

$$\dot{\Omega} = \frac{1}{2}\rho S I^{-1} V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) - I^{-1}\Omega \times (I\Omega) \tag{2.6b}$$

These two equations can be rearranged to account for the effect of the wind, allowing further on to simulate the behaviour of the airplane in the presence of wind disturbances. Let $\vec{V_G} = [u \quad v \quad w]^T$ be the speed of the CG relative to the ground, $\vec{V}$ the speed of the CG relative to the air mass and $\vec{W}$ the

speed of the wind relative to the ground, then as per Etkin and Reid [7]

$$\vec{V_G} = \vec{V} + \vec{W} = \begin{bmatrix} V_a c_\alpha c_\beta + V_{w_x} \\ V_a s_\beta + V_{w_y} \\ V_a s_\alpha c_\beta + V_{w_z} \end{bmatrix} \tag{2.7}$$

and α and β can be computed by

$$\alpha = arctan\left(\frac{w}{u}\right) \tag{2.8a}$$

$$\beta = arctan\left(\frac{v}{V_a}\right) \tag{2.8b}$$

From these three equations, differentiating 2.8a and 2.8b comes that

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} H_{11} & 1 & H_{13} \\ H_{21} & 0 & H_{23} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{2.9}$$

SEE ANNEX HERE

The angular rates are also related to the Euler angles. The relationship between the euler angles and the rotation rates is also one that will prove useful when implementing the model on a Matlab simulation, and is given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{psi} \end{bmatrix} = \begin{bmatrix} 1 & tg_\theta s_\phi & tg_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.10}$$

2.2 Guidance Dynamics

This subsection on the forces applied to aircraft, introducing a new actuation variable, the thrust force T. These forces are applied along the three axis of the wind frame, lift, drag and side force, given by

$$\begin{bmatrix} D \\ Y \\ L \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \begin{bmatrix} C_D \\ C_Y \\ C_L \end{bmatrix} \tag{2.11}$$

Once again, the method used to compute these coefficients will be given in the chapter 3 in detail. These coefficients, similarly to the moment coefficient, are functions of the angle of attack, sideslip angle and airspeed, the three most relevant variables when determining aerodynamic forces and moments. Although aerodynamic forces are usually expressed on the wind frame, as the thrust is always applied along the x axis of the body frame, it is necessary to rotate the aerodynamic forces to this frame. This way the sum of the airplane's forces can be obtained.

$$\begin{bmatrix} F_{xa} \\ F_{ya} \\ F_{za} \end{bmatrix} = R_{WB} \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix} \tag{2.12}$$

From Newton's Second Law comes the aircraft's acceleration

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(F_{xa} + T) - gs_\theta + rv - qw \\ \frac{1}{m}F_{ya} + gc_\theta s_\phi + pw - ru \\ \frac{1}{m}F_{za} + gc_\theta c_\phi + qu - pv \end{bmatrix} \tag{2.13}$$

An expression in the Earth frame can also be obtained

$$\begin{bmatrix} \ddot{x_E} \\ \ddot{y_E} \\ \ddot{z_E} \end{bmatrix} = \frac{1}{m} R_{BE} \begin{bmatrix} F_{xa} + T \\ F_{ya} \\ F_{za} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.14}$$

2.2.1 Actuator Dynamics

Finally, to simulate the delay response in actuation in order to have a realistic simulation, first order systems were introduced to the actuator dynamics as per [6]. For the control surfaces δ_i, given a desired δ_i^d comes

$$\dot{\delta_i} = \frac{1}{\xi_i}(\delta_i^d - \delta_i) \tag{2.15}$$

Similarly for the thrust

$$\dot{T} = \frac{1}{\xi_T}(T^d - T) \tag{2.16}$$

ξ_i and ξ_T being time constants. As the responsiveness of the resultant thrust will be much slower than that of the control surfaces, $\xi_T >> \xi_i$.

2.3 Feedback linearisation

Feedback linearisation, also known as dynamic inversion, is an approach based on the idea of algebraically transforming a non-linear system into a linear one, from which linear control laws can be used to control the resulting system. Unlike Jacobian linearisation, that assumes linearity of the system around an equilibrium value, feedback linearisation implements a feedback loop that cancels non-linearities of the given system [8]. Given a nonlinear system

$$\dot{x} = f(x, u) \tag{2.17}$$

one must find both a state transformation $z = z(x)$ and an input transformation $\upsilon = \upsilon(x, u)$ in order

for the transformed system to be a linear and time-invariant system $\dot{z} = Az + Bv$. This process is called Input-State linearisation.

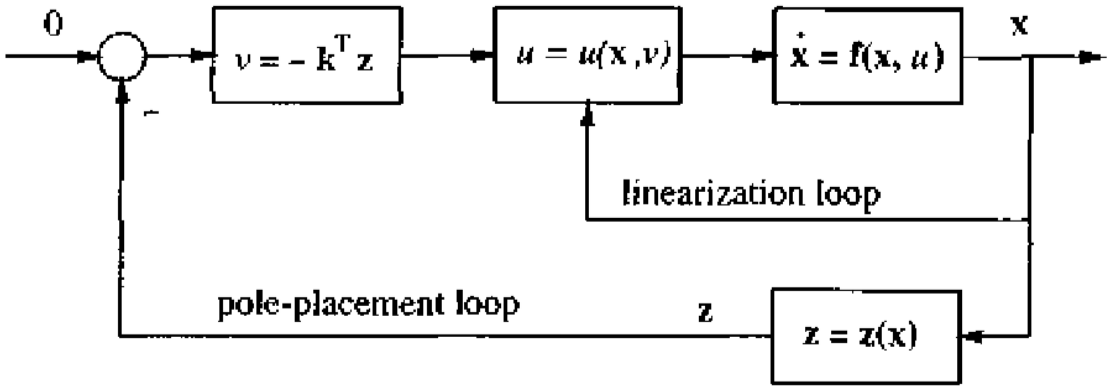


Figure 2.1: Feedback linearisation example

EXPLAIN NECESSARY MATHEMATICAL CONCEPTS HERE

2.3.1 SISO systems

Although an aircraft is always considered as a MIMO system, a description of this control concept will firstly be provided for a general SISO case, before generalising to a MIMO case. Nonlinear systems that can be represented as $\dot{x} = f(x) + g(x)u$ will be discussed in this section. From [8], a definition of Input-State linearisation is to find, for a nonlinear system of relative degree n, a *diffeomorphism* $\phi : \Omega \to R^n$ and nonlinear control law

$$u = \alpha(x) + \beta(x)v \tag{2.18}$$

such that the new state variables $z = \phi(x)$ and *pseudo-input* v satisfy the linear time invariant system $\dot{z} = Az + Bv$, where the following equations are satisfied

$$\begin{cases} \dot{z_i} = z_{i+1} & if \quad i < n-1 \\ \dot{z_n} = v & if \quad i = n-1 \end{cases} \tag{2.19}$$

In order to find such a function and control law, one must find a state z_1 such that

$$\nabla z_1 ad_f^i g = 0 \qquad i = 0, ..., n-2 \tag{2.20a}$$

$$\nabla z_1 ad_f^{n-1} g \neq 0 \tag{2.20b}$$

The remaining states are then obtained from $z(x) = [z_1 \quad L_f z_1 \quad ... \quad L_f^{n-1} z_1]^T$ and the input transformation from

$$\alpha(x) = -\frac{L_f^n z_1}{L_g L_f^{n-1} z_1} \tag{2.21a}$$

$$\beta(x) = \frac{1}{L_g L_f^{n-1} z_1} \tag{2.21b}$$

2.3.2 MIMO systems

These concepts can also be extended to to MIMO systems in a similar manner, by similarly differentiating the outputs until the inputs explicitly appear. This time however, there are individual relative degrees per output. The sum of these relative degrees is called total degree r and must satisfy $r < n$, n being the order of the system. Given a MIMO system

$$\dot{x} = f(x) + G(x)u \tag{2.22}$$

$$y = h(x) \tag{2.23}$$

the ϕ functions are defined as $\phi_j^i(x) = L_f^{j-1} h_i(x)$ and satisfy a condition similar to 2.19

$$\dot{\phi}_1^i(x) = \phi_2^i, ..., \dot{\phi}_{r_i-1}^i = \phi_{r_i}^i \quad \text{and} \quad \dot{\phi}_{r_i}^i = L_f^{r_i} h_i(x) + \sum_{j=1}^{m} L_{g_j} L_f^{r_i-1} h_i(x) u_j \tag{2.24}$$

As this equation holds for $1 < i < p$, p being the number of outputs of the system, giving an expression for the pseudo control input υ

$$\upsilon = \begin{bmatrix} \dot{\phi}_{r_1}^1 \\ \vdots \\ \dot{\phi}_{r_p}^p \end{bmatrix} \tag{2.25}$$

It is also interesting to note the resulting system is not only linear, but also completely decoupled, as each new pseudo input only affects one output.

2.4 Limitations of feedback linearisation

Although feedback control answers to many of the problems of linear controllers such as PID and LQR, it has a big downside. Indeed, being a model based controller, it requires the system model to be quite accurately known. Let υ be the vector of pseudo inputs, for a second order $\ddot{x} = f(x, \dot{x}, u)$, if it is input-output linearisable and the exact system is known, then it comes that, from [9]

$$\ddot{x} = \upsilon \tag{2.26}$$

However, even for simpler models, a real system is difficult to accurately describe, and an approximation $\hat{f}$ of f is usually chosen, resulting in $\upsilon = \hat{f}(x, \dot{x}, u)$. Therefore, defining a dynamic inversion error, $\Delta(x, \dot{x}, u) = f(x, \dot{x}, u) - \hat{f}(x, \dot{x}, u)$, comes that, as stated in [9]

$$\ddot{x} = \upsilon + \Delta(x, \dot{x}, u) \tag{2.27}$$

There can be many causes for a dynamic inversion error to be present, as not only modelling incertitudes can lead to larger errors, but also external interference (wind gusts) and actuator faults. The goal of this section will be to present the reader with some solutions to minimize this error.

2.4.1 Control improvement

2.5 Neural Networks

Artificial Neural Networks (NN) are a biologically inspired simulation of brain's nervous system. They are composed of simulated neurons and synapses. The sum of the inputs of a neuron are summed and the result is fed into an activation function, which then return a bounded value, either between -1 and 1 or 0 and 1. The inputs and outputs of a neuron are multiplied by a weight, representing the synapses between neurons. We get the following equation for one neuron

$$y(x_1, x_2, ..., x_n) = f((\sum_{i=1}^{n} w_i x_i) + b) \tag{2.28}$$

where b is the bias term and f a given activation function. Neuron are then set in three different types of layers: the input, the output and the hidden layers. These are connected between each other, and the hidden part can be composed of several layers. The weights can then be tuned based on experience, making it suitable for intelligent learning. To tune these weights learning algorithms are used. There are two types of learning algorithms: batch and online training. It is called training to the process of iteratively reaching the ideal set of weights that will minimise the error between the output of the neural network and that of a given unknown function.

Batch training is mostly based one a property NN is that, according to the Universal Approximation Theorem showed in 1989 by George Cybenko, a feed forward neural network can approximate any continuous function. The network is trained by providing input and output pairs. The weights are found in order to approximate the output of the neural network to that of the original function. Once these weights are computed, the network can then be used to compute outputs from inputs that were not part of the initial knowledge of input-output pairs.

Although this learning method approach is used in many of NN applications, it cannot however be easily used to improve an existing feedback linearisation controller, as it requires an *a priori* knowledge of the modelling error for each given input.

Online training, contrary to batch training, can deal with dynamically changing environments. This training paradigm is not based on an *a priori* knowledge set, and instead relies on update laws that compute the new weights after each control iteration. It is this method that will be retained to adaptively correct the inversion error from the feedback linearisation. Training a network online, however, is less trivial when compared to batch training. An algorithm named backpropagation will be presented to provide a way of training a network online.

The backpropagation algorithm is one of the most common NN training algorithm, commonly used in

batch training, used to update the weights of each layer of a network, in order to minimize a cost function. However, research has been done in order to use backpropagation techniques to create online adaptive and augmented control such as [10], [11], [12]. Indeed, it is also possible to use such algorithms to continuously update the weights of a network as the data is generated. Backpropagation can only be implemented if the activation function of the NN is differentiable.

The backpropagation algorithm can be described by a two phase cycle, Propagation and Weight update. During this last phase the gradient descent algorithm is used to optimize a cost function. As such, a cost function E must be chosen, and must be a function of the outputs of the neural network. A commonly used error function is

$$E = \frac{1}{2}(y_d - y)^2 \tag{2.29}$$

Where y_d is the desired output of the network and y is the actual output of the network. The $\frac{1}{2}$ factor is added to be cancelled when differentiating. For a given neuron i, the sum of its inputs sum_i is given by

$$sum_i = \sum_{k=1}^{n} w_{ki} x_k \tag{2.30}$$

Where w_{ki} is the weight of the k^{th} input of neuron i, and x_k are the outputs of the neurons form the previous layer. Given an activation function $f(x)$, we define the output of neuron i as

$$x_i = f(sum_i) \tag{2.31}$$

In order to use the gradient descent algorithm, the derivative of the cost function with respect to a given weight w_{ij} must be computed. To do so, the chain rule is used as such

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial sum_j} \frac{\partial sum_j}{\partial w_{ij}} \tag{2.32}$$

These three factors can now be easily computed. The last two factors are independent of weather x_j is a hidden or output layer and will for this reason be calculated firstly. From 2.30 and 2.31 comes that

$$\frac{\partial x_j}{\partial sum_j} = \frac{\partial f(sum_j)}{\partial sum_j} \tag{2.33}$$

For an output layer, where values may be unbounded, linear output functions may be used, for which the equation above has a trivial solution. For output layers of classification neural networks or for hidden layers in general, logistic or tangent sigmoid functions can be used instead. For the first case let $f(x) = \frac{1}{1+e^{-x}}$, then

$$\frac{\partial f(sum_j)}{\partial sum_j} = f(sum_j)(1 - f(sum_j)) \tag{2.34}$$

For the case of a tangent sigmoid activation function $f(x) = tanh(x)$ comes

$$\frac{\partial f(sum_j)}{\partial sum_j} = 1 - f^2(sum_j) \tag{2.35}$$

For the term $\frac{\partial sum_j}{\partial w_{ij}}$, from 2.30 this partial derivative is easily computed

$$\frac{\partial sum_j}{\partial w_{ij}} = \frac{\partial \left(\sum_{k=1}^{n} w_{kj} x_k\right)}{\partial w_{ij}} = x_i \tag{2.36}$$

For the output layer, the first term of equation 2.32 can easily be evaluated, as $x_j = y$ for such a case, giving

$$\frac{\partial E}{\partial x_j} = \frac{\partial(\frac{1}{2}(y_d - y)^2)}{\partial y} = y - y_d \tag{2.37}$$

For input layers however, this derivative is less obvious to estimate. One can consider that the error function E is a function of the inputs of all neurons that take x_j as input. Let S be such a set of inputs, from the total derivative to x_j comes that

$$\frac{\partial E}{\partial x_j} = \sum_{s \in S} \left(\frac{\partial E}{\partial x_s} \frac{\partial x_s}{\partial sum_s} \frac{\partial sum_s}{\partial x_j} \right) = \sum_{s \in S} \left(\frac{\partial E}{\partial x_s} \frac{\partial f(sum_s)}{\partial sum_s} w_{js} \right) \tag{2.38}$$

Note that the term $\frac{\partial E}{\partial x_s}$ represent the all the derivatives of the errors with respect to the outputs of the next layer of neurons. Taking into account that the last layer is easily evaluated without this method, the derivatives of all hidden neurons can be computed. Knowing the value of 2.32, an update law for the weights can now be stated from the gradient descent algorithm

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{2.39}$$

Where α is a learning coefficient, and its choice is crucial to assure a fast convergence of the solution. Indeed, while a exceedingly large learning coefficient can cause this algorithm to miss the minimum error, a small learning coefficient will also increase the convergence rate. The -1 factor is added in the equation above to converge to a local minima, removing it would make this algorithm maximize the cost function. A limitation of the backstepping algorithm that must also be taken into account is that it only guarantees local minima convergence.

Chapter 3

Implementation

Insert your chapter material here...

3.1 Numerical Model

Description of the numerical implementation of the models explained in Chapter 2...

3.2 Verification and Validation

Basic test cases to compare the implemented model against other numerical tools (verification) and experimental data (validation)...

Chapter 4

Results

Insert your chapter material here...

4.1 Problem Description

Description of the baseline problem...

4.2 Baseline Solution

Analysis of the baseline solution...

4.3 Enhanced Solution

Quest for the optimal solution...

4.3.1 Figures

Insert your section material and possibly a few figures...

Make sure all figures presented are referenced in the text!

Images

Figure 4.1: Caption for figure.

(a) Airbus A320

(b) Bombardier CRJ200

Figure 4.2: Some aircrafts.

Make reference to Figures 4.1 and 4.2.

By default, the supported file types are *.png,.pdf,.jpg,.mps,.jpeg,.PNG,.PDF,.JPG,.JPEG*.

See `http://mactex-wiki.tug.org/wiki/index.php/Graphics_inclusion` for adding support to other extensions.

Drawings

Insert your subsection material and for instance a few drawings...

The schematic illustrated in Fig. 4.3 can represent some sort of algorithm.

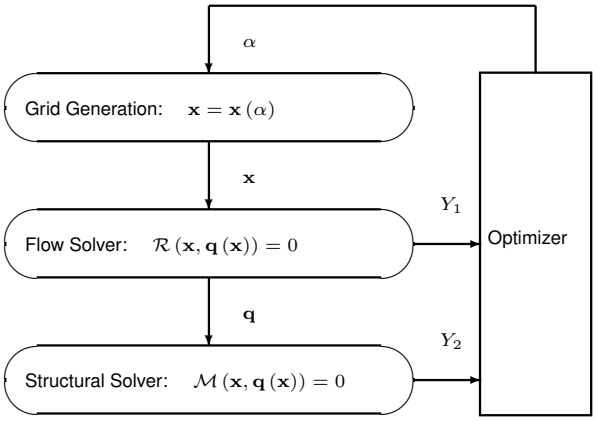


Figure 4.3: Schematic of some algorithm.

4.3.2 Equations

Equations can be inserted in different ways.

The simplest way is in a separate line like this

$$\frac{\mathrm{d}q_{ijk}}{\mathrm{d}t} + \mathcal{R}_{ijk}(\mathbf{q}) = 0\,. \tag{4.1}$$

If the equation is to be embedded in the text. One can do it like this $\partial\mathcal{R}/\partial\mathbf{q} = 0$.

It may also be split in different lines like this

$$\begin{aligned} \text{Minimize} \quad & Y(\alpha, \mathbf{q}(\alpha)) \\ \text{w.r.t.} \quad & \alpha\,, \\ \text{subject to} \quad & \mathcal{R}(\alpha, \mathbf{q}(\alpha)) = 0 \\ & C(\alpha, \mathbf{q}(\alpha)) = 0\,. \end{aligned} \tag{4.2}$$

It is also possible to use subequations. Equations 4.3a, 4.3b and 4.3c form the Naver–Stokes equations 4.3.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_j\right) = 0\,, \tag{4.3a}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j + p\delta_{ij} - \tau_{ji}\right) = 0, \quad i = 1, 2, 3\,, \tag{4.3b}$$

$$\frac{\partial}{\partial t}\left(\rho E\right) + \frac{\partial}{\partial x_j}\left(\rho E u_j + p u_j - u_i \tau_{ij} + q_j\right) = 0\,. \tag{4.3c}$$

4.3.3 Tables

Insert your subsection material and for instance a few tables...

Make sure all tables presented are referenced in the text!

Follow some guidelines when making tables:

- Avoid vertical lines
- Avoid “boxing up” cells, usually 3 horizontal lines are enough: above, below, and after heading
- Avoid double horizontal lines
- Add enough space between rows

Model	C_L	C_D	C_{My}
Euler	0.083	0.021	-0.110
Navier–Stokes	0.078	0.023	-0.101

Table 4.1: Table caption.

Make reference to Table 4.1.

Tables 4.2 and 4.3 are examples of tables with merging columns:

An example with merging rows can be seen in Tab.4.4.

If the table has too many columns, it can be scaled to fit the text widht, as in Tab.4.5.

	Virtual memory [MB]	
	Euler	Navier–Stokes
Wing only	1,000	2,000
Aircraft	5,000	10,000
(ratio)	$5.0\times$	$5.0\times$

Table 4.2: Memory usage comparison (in MB).

	$w = 2$			$w = 4$		
	$t = 0$	$t = 1$	$t = 2$	$t = 0$	$t = 1$	$t = 2$
$dir = 1$						
c	0.07	0.16	0.29	0.36	0.71	3.18
c	-0.86	50.04	5.93	-9.07	29.09	46.21
c	14.27	-50.96	-14.27	12.22	-63.54	-381.09
$dir = 0$						
c	0.03	1.24	0.21	0.35	-0.27	2.14
c	-17.90	-37.11	8.85	-30.73	-9.59	-3.00
c	105.55	23.11	-94.73	100.24	41.27	-25.73

Table 4.3: Another table caption.

ABC	header			
	1.1	2.2	3.3	4.4
IJK	group		0.5	0.6
			0.7	1.2

Table 4.4: Yet another table caption.

Variable	a	b	c	d	e	f	g	h	i	j
Test 1	10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000	90,000	100,000
Test 2	20,000	40,000	60,000	80,000	100,000	120,000	140,000	160,000	180,000	200,000

Table 4.5: Very wide table.

4.3.4 Mixing

If necessary, a figure and a table can be put side-by-side as in Fig.4.4

Legend		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Figure 4.4: Figure and table side-by-side.

Chapter 5

Conclusions

Insert your chapter material here...

5.1 Achievements

The major achievements of the present work...

5.2 Future Work

A few ideas for future work...

Bibliography

[1] S. et al. State-of-the-art intelligent flight control systems in unmanned aerial vehicles. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, 2017.

[2] A. Zulu and S. John. A review of control algorithms for autonomous quadrotors. *Open Journal of Applied Sciences*, 2014.

[3] A. K. S. et al. Neuro-adaptive augmented dynamic inversion controller for quadrotor. *IFAQ Papers Online*, 2016.

[4] M. P. et al. Augmentation of a non linear dynamic inversion scheme within the nasa ifcs f-15 wvu simulator. *Proceedings of the American Control Conference*, 2013.

[5] T. X. et al. Adaptive flight control for quadrotor uavs with dynamic inversion and neural networks. *IEEE Intemational Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016.

[6] F. M. C. H. Escamilla Nuñez. Towards 4d trajectory tracking for transport aircraft. *IFAQ World Congress*, 2017.

[7] B. Etkin and L. D. Reid. *Dynamics of Flight Stability and Control*. 1996.

[8] W. L. Jean-Jacques E. Slotine. *Applied nonlinear control*. 1991.

[9] L. Q. YANG Ning, WU Liao-ni*. Adaptive flight control law based on neural networks and dynamic inversion for unmanned aerial vehicle. *International Conference on Automatic Control and Artificial Intelligence*, 2012.

[10] Y. Tang and R. J. Patton. Reconfigurable flight control using feedback linearization with online neural network adaption. *Conference on Control and Fault-Tolerant Systems*, 2013.

[11] T. Xiang, F. Jiang, Q. Hao, and W. Cong. Adaptive flight control for quadrotor uavs with dynamic inversion and neural networks. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 174–179, Sept 2016. doi: 10.1109/MFI.2016.7849485.

[12] Q. Lin, Z. Cai, Y. Wang, J. Yang, and L. Chen. Adaptive flight control design for quadrotor uav based on dynamic inversion and neural networks. In *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pages 1461–1466, Sept 2013. doi: 10.1109/IMCCC.2013.326.

Appendix A

Vector calculus

In case an appendix if deemed necessary, the document cannot exceed a total of 100 pages...

Some definitions and vector identities are listed in the section below.

A.1 Vector identities

$$\nabla \times (\nabla \phi) = 0 \tag{A.1}$$

$$\nabla \cdot (\nabla \times \mathbf{u}) = 0 \tag{A.2}$$

Appendix B

Technical Datasheets

It is possible to add PDF files to the document, such as technical sheets of some equipment used in the work.

B.1 Some Datasheet

SUNPOWER

C60 SOLAR CELL

MONO CRYSTALLINE SILICON

BENEFITS

Maximum Light Capture
SunPower's all-back contact cell design moves gridlines to the back of the cell, leaving the entire front surface exposed to sunlight, enabling up to 10% more sunlight capture than conventional cells.

Superior Temperature Performance
Due to lower temperature coefficients and lower normal cell operating temperatures, our cells generate more energy at higher temperatures compared to standard c-Si solar cells.

No Light-Induced Degradation
SunPower n-type solar cells don't lose 3% of their initial power once exposed to sunlight as they are not subject to light-induced degradation like conventional p-type c-Si cells.

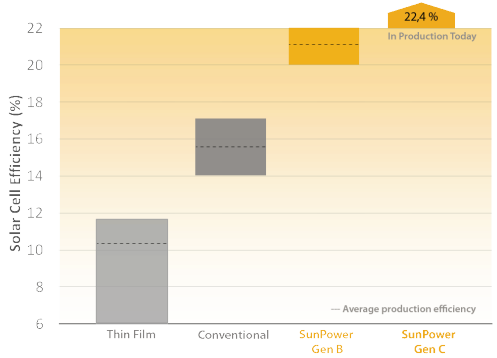
Broad Spectral Response
SunPower cells capture more light from the blue and infrared parts of the spectrum, enabling higher performance in overcast and low-light conditions.

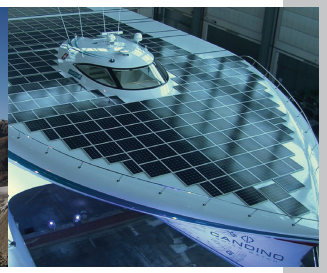
Broad Range Of Application
SunPower cells provide reliable performance in a broad range of applications for years to come.

The SunPower™ C60 solar cell with proprietary Maxeon™ cell technology delivers today's highest efficiency and performance.

The anti-reflective coating and the reduced voltage-temperature coefficients provide outstanding energy delivery per peak power watt. Our innovative all-back contact design moves gridlines to the back of the cell, which not only generates more power, but also presents a more attractive cell design compared to conventional cells.

SunPower's High Efficiency Advantage





C60 SOLAR CELL

SUNPOWER

C60 SOLAR CELL

MONO CRYSTALLINE SILICON

Electrical Characteristics of Typical Cell at Standard Test Conditions (STC)

STC: 1000W/m², AM 1.5g and cell temp 25°C

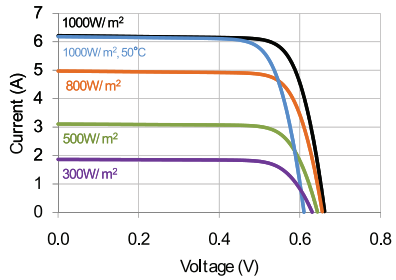
Bin	Pmpp (Wp)	Eff. (%)	Vmpp (V)	Impp (A)	Voc (V)	Isc (A)
G	3.34	21.8	0.574	5.83	0.682	6.24
H	3.38	22.1	0.577	5.87	0.684	6.26
I	3.40	22.3	0.581	5.90	0.686	6.27
J	3.42	22.5	0.582	5.93	0.687	6.28

All Electrical Characteristics parameters are nominal
Unlaminated Cell Temperature Coefficients
Voltage: -1.8 mV / °C Power: -0.32% / °C

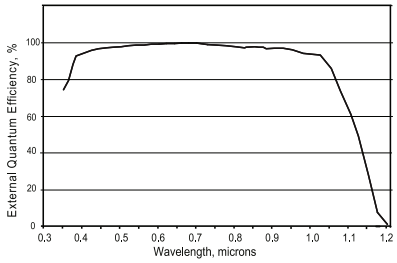
Positive Electrical Ground

Modules and systems produced using these cells must be configured as "positive ground systems".

TYPICAL I-V CURVE



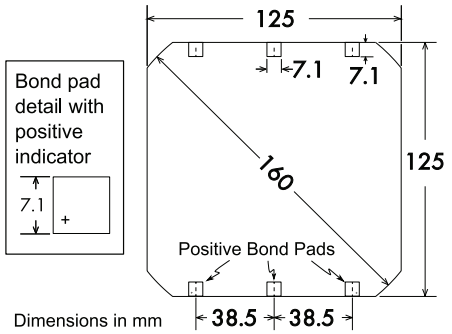
SPECTRAL RESPONSE



Physical Characteristics

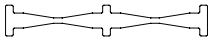
Construction:	All back contact
Dimensions:	125mm x 125mm (nominal)
Thickness:	165µm ± 40µm
Diameter:	160mm (nominal)

Cell and Bond Pad Dimensions



Bond pad area dimensions are 7.1mm x 7.1mm
Positive pole bond pad side has "+" indicator on leftmost and rightmost bond pads.

Interconnect Tab and Process Recommendations



Tin plated copper interconnect. Compatible with lead free process.

Packaging

Cells are packed in boxes of 1,200 each; grouped in shrink-wrapped stacks of 150 with interleaving. Twelve boxes are packed in a water-resistant "Master Carton" containing 14,400 cells suitable for air transport.

Interconnect tabs are packaged in boxes of 1,200 each.

About SunPower

SunPower designs, manufactures, and delivers high-performance solar electric technology worldwide. Our high-efficiency solar cells generate up to 50 percent more power than conventional solar cells. Our high-performance solar panels, roof tiles, and trackers deliver significantly more energy than competing systems.

sunpowercorp.com

Document #001-66352 Rev** A4_en