Julius Maximilian University of Würzburg
Faculty of Mathematics and Computer Science

Aerospace Information Technology

Chair of Computer Science VIII Prof. Dr. Sergio Montenegro

Lulea University of Technology
Department of Computer Science, Electrical and Space Engineering

Space Technology

Chair of Space Technology Division Dr. Victoria Barabash

Master Thesis

Design and Implementation of a 6DOF Control System for an Autonomous Quadrocopter

by
Alexander Lebedev

Examiner: Prof. Dr. Sergio Montenegro
Examiner: Dr. Anita Enmark
Supervisor: Dipl.-Ing. Nils Gageik

Würzburg, 07. 09. 2013

Declaration

I, Alexander Lebedev, hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another author nor material which to a substantial extent has been accepted for the award of any other degree or diploma of a university or other institute of a higher education, except where due acknowledgment has been made in the text.

Würzburg, 07.09.2013 Alexander Lebedev

Abstract

This thesis is dedicated to design and implementation of a 6DOF control system for a quadrocopter. At the beginning of the work the quadrocopter was analyzed as a plant and physical effects with behavior of continuous /discrete elements were described. Based on the mathematical equations, continuous time invariant nonlinear mathematical model was designed. This mathematical model was linearized to create a 6DOF control system and validated thought experiments by test benches and a flying prototype of the quadrocopter. For the control system design a pole-placement approach was chosen and based on the linear validated model, with taking into account requirements to a settling time, an overshoot and a steady-state error, the control system was designed. Its behavior was checked in simulation and showed adequate results. Afterwards designed control system was implemented as a script and incorporated in a soft, developed inside 'Aerospace Information Technology' Department, University of Würzburg. Then series of experiments by test benches and the flying prototype were fulfilled. Based on comparing experimental and theoretical results a conclusion was made. At the end of the work advantages and drawbacks of the control system were discussed and suggestions for future work were declared.

Acknowledgment

First, I would like to express my sincerely gratitude to Prof. Dr. Montenegro, Dipl. Ing. Nils Gageik and all other team members for their help and support during this project. Also I would like to thank Dr. Anita Enmark for her patients and advice.

My sincere thanks to Ms. Shahmary and Ms. Winneback for their kind help.

Finally I would like to thank Dr. Victoria Barabash for her support and understanding.

Table of Contents

Abbreviations

CoG	center of gravity
CoM	center of mass
CST	Control System Toolbox
DOF	degree of freedom
EMF	electromotive force
MM	mathematical model
MoI	moment of inertia
TF	transfer function
ToI	tensor of inertia
UAV	unmanned aerial vehicle
YPR	yaw-pitch-roll

Chapter 1 Introduction

1.1 Motivation and tasks of this work

A quadrocopter is a flying object, which changes its altitude and attitude by four rotating blades. Quadrocopters are a variation of multicopters, which are rotorcrafts. During 20th century there were several attempts to implement manned quadrocopters, earliest known cases are in 1922 by Etienne Oemichen in France [25] and by George Bothezat in USA [26]. However, during the progress in rotorcrafts industry, the helicopters with different schemes of rotors adjusting were chosen.

In last decades, because of great achievements in technologies such as electronics, microcontrollers, motors, sensors and software, an opportunity of building small unmanned aerial vehicles (UAVs) became wide world available. This one leads to growing research and engineering interest to quadrocopters, which can be easily built. Nowadays quadrocopters are used mostly as toys, objects for teaching purposes in universities and for panorama video recording, but ones have good prospects in other areas. For expansion of application areas they should be more autonomous and intelligent. They are planned to be used in rescue operations [28], as a fire-fighter [27] or working as a group for fulfillment tasks with general purposes [29].

Quadrocopters have advantages such as a high maneuverability, a relatively cheap price and a simple construction and have a great potential for using as robotic autonomous devices. However, there are several problems that should be solved or improved for making ones closer to real applications. One of these problems is a real time 6DOF control system that can control a position and an orientation of a quadrocopter, its linear and angular velocities. Such type of the control system is very important for fulfillment series of tasks, e.g. grasping other objects, tracking other objects or transmitting video information about other objects. Some good results of controlling a quadrocopter behavior were obtained and demonstrated by GRASP laboratory of Pennsylvania University [30] and inside project ‘Flying Machine Area’ from Zürich University [31].

Hereby, the main motivation of this project is creating real time a 6DOF control system. This control system should control a position and an orientation of a quadrocopter.

For creating such time of the system, several tasks should be solved. At the beginning a mathematical model of a quadrocopter should be created. Then, based on this mathematical model, a 6DOF control system should be designed. At the end, designed control system should be implemented as a code in a microcontroller for a real quadrocopter.

A quadrocopter that will be under consideration in this thesis is the one from 'AQopterI8' project, which is developed at Aerospace Information Technology Department, University of Würzburg.

1.2 State of the Art

A mathematical model of a quadrocopter consists of describing rigid body dynamics, kinematics of fixed and body reference frames and forces applied to the quadrocopter. There are several variants of the model. Firstly they vary in describing of rigid body dynamics; it can be done by Euler equations [5], Euler-Newton approach [20] or Lagrangian approach [21]. Secondly they vary in end representations of kinematics and direction of z axis of body reference frame. Thirdly, they differ in how many forces and other effects are taken into account. The most complete model is represented by S. Bouabdallah [21], the simplest variant by R. Beard [5] and the variant in the 'middle' by T. Luukkonnen [20]. Also some researchers simplified a model of a motor, which rotates a blade, as proportional coefficients [5], and some of them as a 1^{st} order transfer function [21]. A model for this thesis is based on models from two works [20], [5].

Control designs used in many works are based on the mathematical model. Usually, original model is linearized to linear continuous time invariant model [20, 5] or to discrete one [24]. A controller for attitude control is usually PD [23] and there are several variants for altitude control. Hover control represented by N.Michael and others [23] was chosen for the quadrocopter control. There are several variants for the structure of control system for a whole plant. The variant from N.Michael and others [23] was chosen.

1.3 Chapters overview

In chapter 2 ‘Mathematical model of the quadrocopter’ several issues are discussed. At the beginning an analysis of quadrocopter physical processes is done and collected as one process. Then deferential equations described each process are represented. Based on these equations transfer functions were obtained and implemented as a model in Matlab/Simulink.

Chapter 3 ‘Design of the Control System’ dedicates to choosing structures of controllers and calculation their coefficients. It starts from short discussion about pole-placement approach. A method for choosing poles based on quality requirements is discussed. Then calculation feedback coefficients by Ackerman method is represented. At the end, implementation in Matlab/Simulink is described and results of simulation are shown.

Chapter 4 ‘Implementation of the Control System’ contains information about experiments for a validation the mathematical model and a controllers adjusting. Firstly, the validation of the mathematical model for the pitch/roll, the calculation for controllers for pitch/roll and a comparison of modeling and experimental results were represented. Afterwards, the same information about the yaw was described. Then experiments with a flying prototype were shown and compared.

Chapter 5 ‘Conclusion’ contains discussion of the results and recommendation for a future work.

Chapter 2 Mathematical model of the quadrocopter

2.1 Analysis of the quadrocopter

The quadrocopter consists of four sticks, where each two are set symmetrically and perpendicularly to each other. On the end of each stick, symmetrically to geometrical center of the quadrocopter, actuators that provide flying are set. Each actuator consists of a motor and a blade, where the blade is fixed to the motor's shaft (fig. 2.1). Rotation of these blades can lead to qudrocopter's motion.

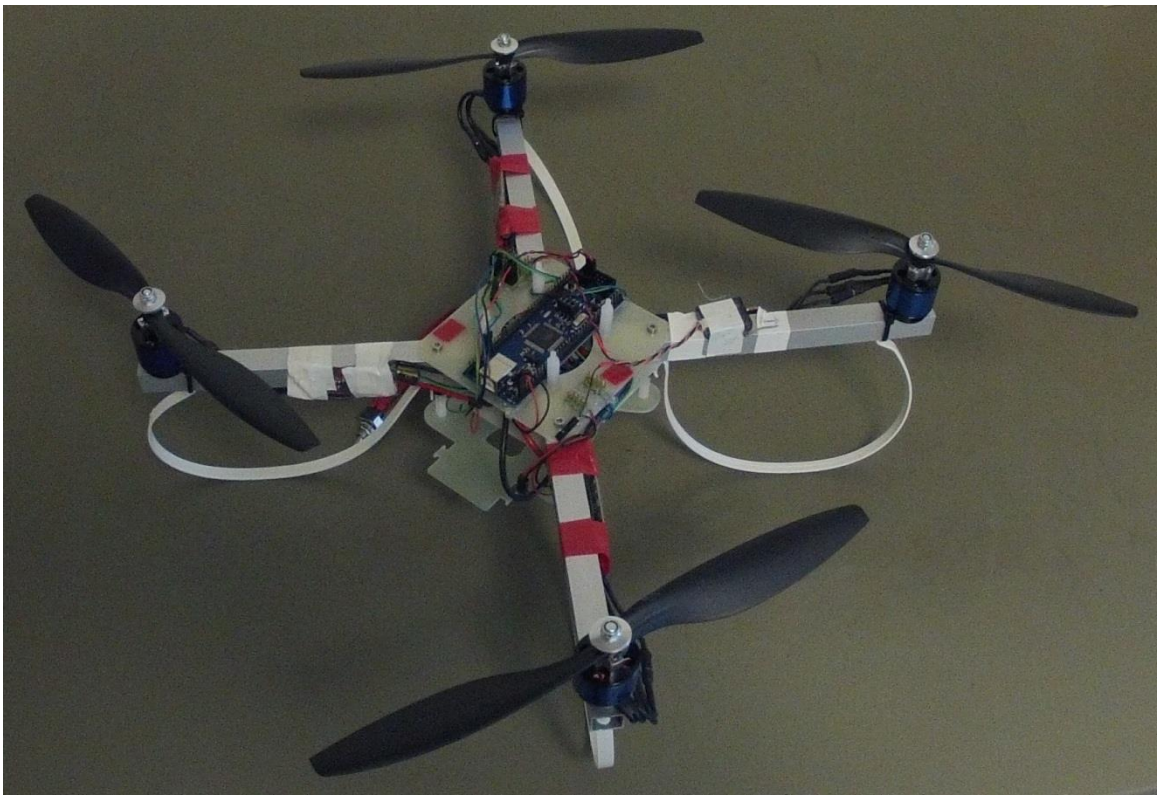

fig. 2.1 Structure of the quadrocopter

The quadrocopter has 6DOF that means it has linear and angular motions. This complex motion (called free motion) can be fully determined by two vectors: a position vector $\overrightarrow{pst}$ and an orientation vector $\overrightarrow{ort}$. The position vector has current position of the quadrocopter in Earth reference frame and the orientation vector has current orientation angles of the quadrocopter comparing to Earth reference frame. For calculation current values of $\overrightarrow{pst}$ the following parameters should be known: a vector of linear velocity $\vec{v}$, a vector of linear acceleration $\vec{a}$. For calculation the

current values of $\overrightarrow{ort}$ the following parameters should be known: a vector of angular velocity $\vec{\omega}$, a vector of angular acceleration $\vec{\varepsilon}$. Also initial conditions of all 6 vectors mentioned above should be known. In addition, external forces and torques, which could be substituted by net force $net\vec{F}$ and net torque $net\vec{\tau}$, lead to changing in the linear and the angular acceleration of the quadrocopter and these changing influences on the position and the orientation. Hereby, for creating the mathematical model (MM) the equations, for calculation vectors $\overrightarrow{pst}$ and $\overrightarrow{ort}$ based on vectors mentioned above, should be declared (fig. 2.2)

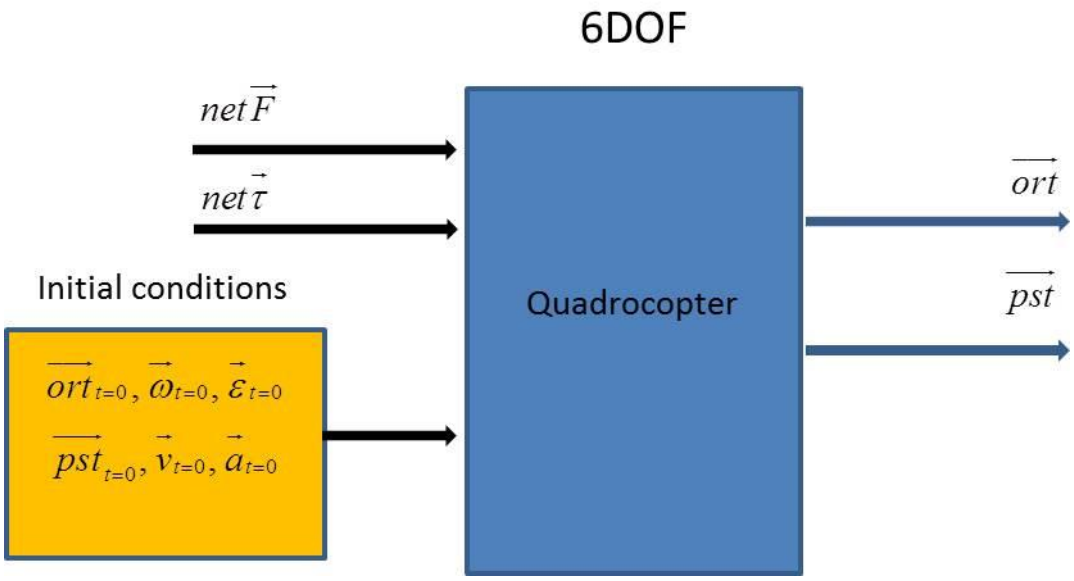

fig. 2.2 Logical diagram for calculation the position and the orientation of the quadrocopter

There are three sources of external forces such as gravitational field, air drag and rotations of the blades in the air. These sources create a gravitational force $\vec{F}_{mg}$, a drag force $\vec{F}_{drag}$ and a thrust force $\vec{T}$ respectively. External torque $\vec{H}$ is created only by blades rotation (fig 2.3).

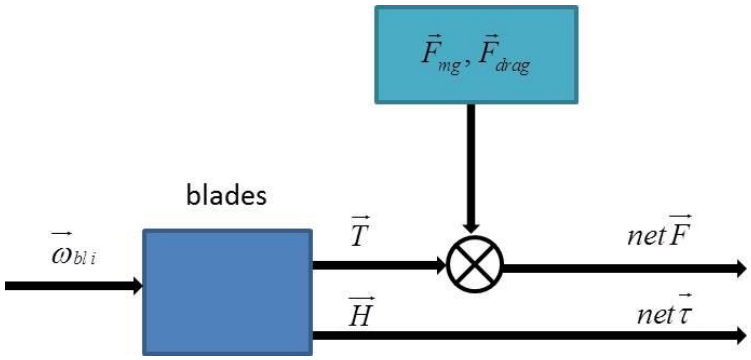

fig. 2.3 Logical diagram for calculation net force and net torque

In total there four blades and four BLDC motors. Assume that motors are numbered from 1 to 4. Each blade is rotated by the corresponding motor with particular angular velocity $\vec{\omega}_{bl\,i}$, where index i indicates the number of the motor. Angular velocity of the motor's shaft is regulated by a power bridge and each power bridge is regulated by a microcontroller (fig. 2.4).

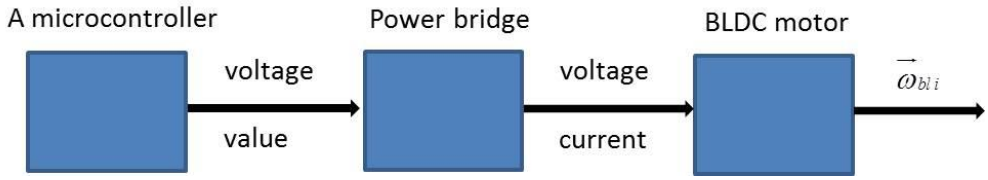

fig. 2.4 Logical diagram for calculation angular velocities of the blades

Hereby, the whole process of moving a quadrocopter in 6DOf can be described as: a microcontroller sets signals (analogous or digital) that are transferred through power bridges and motors to angular velocities of the blades. A rotation of the blades creates forces and torques that together with gravitational and drag forces change the quadrocopter position and orientation (fig. 2.5).

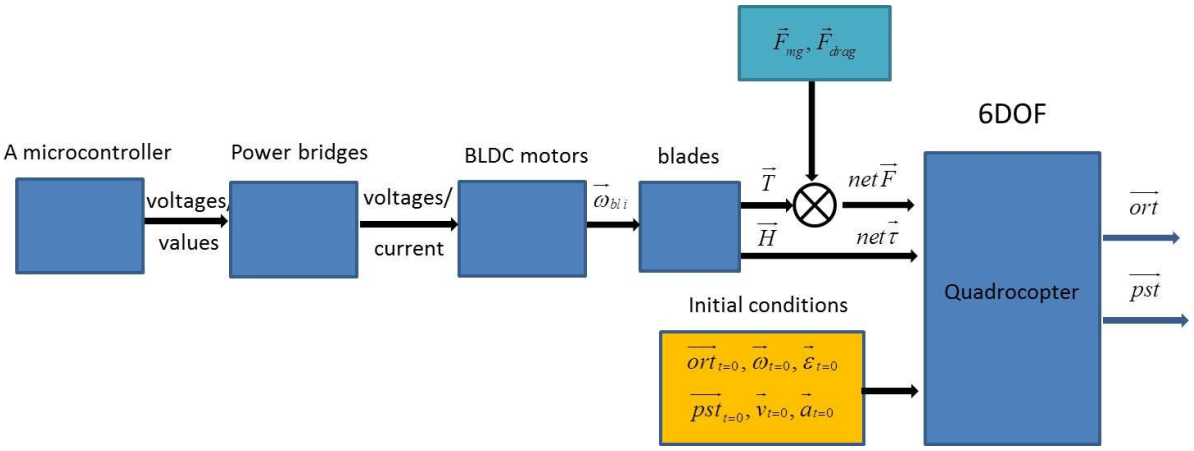

fig. 2.5 Logical diagram for creation the MM of the quadrocopter

The controller needs feedback information such as the position, orientation and appropriate parameters (e.g. linear angular velocity) that should be measured by sensors. Based on desired values of the position and the orientation and current feedback values measured by the sensors the designed controller should generate appropriate values for the power bridges (fig. 2.6).

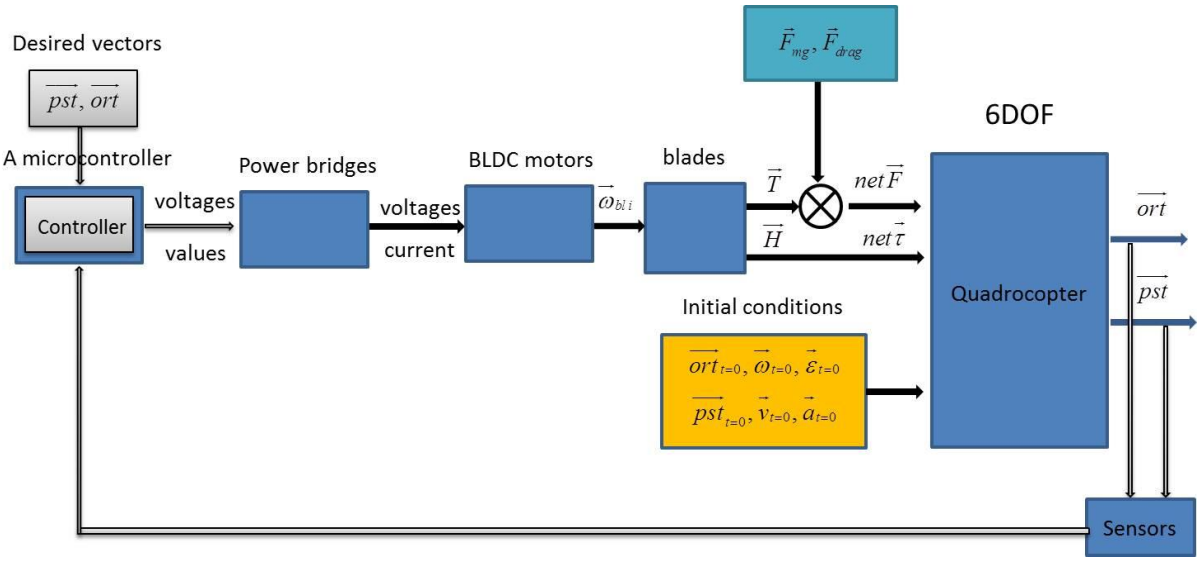

fig. 2.6 Logical diagram for the MM and the controller

It can be concluded that the MM is consists of transfer functions that describe or estimate elements and processes shown on fig. 2.6. So the behavior of the

quadrocopter in 6DOF should be described by differential equations. Then relationships between angular velocities of the blades and net force and net torque should be shown. External forces that cannot be controlled: gravitational and drag forces should be discussed. Elements that are needed for creating force and torque for moving: analogous elements (blades, motors, power bridges) and digital elements (sensors, a microcontroller) should be described. Based on the MM the controller can be designed.

2.2 Mathematical description of the quadrocopter elements

The MM should approximate the process shown on fig. 2.5. A description of this process includes free motion of the quadrocopter, influence from applied forces, how blades rotations are produced and effects of digital elements (sensors, a microcontroller).

2.2.1 Free motion of the quadrocopter

For describing the quadrocopter's free motion (process is shown on fig. 2.2), the theory of free motion of a rigid body is used. According to this one, free motion of the rigid body is considered as a complex motion, which consists of two simple motions:

- translation motion of a point with mass equals to the mass of the body (point mass), where the point is any point of the body
- angular rotation of the body around a fixed point, where the point mass chosen above is considered as the fixed one.

Translation motion of a point is described as:

$$net\vec{F} = \frac{d\vec{p}}{dt}, \quad (2.1)$$

where $net\vec{F}$ is a net force of all external forces applied to the point mass, $\vec{p}$ is linear momentum of the point mass and $\frac{d}{dt}$ is differential operator.

Angular rotation of a body around the fixed point can be described as:

$$net\vec{\tau} = \frac{d\vec{L}}{dt}\ , (2.2)$$

where $net\vec{\tau}$ is a vector sum of all external torques and $\vec{L}$ is an angular momentum of the body.

Translation motion of a point mass

For describing translation motion of a pointed mass a fixed reference frame should be chosen. In some arbitrary point of the space, noted as 'O', a fixed reference frame XYZ is created. Position of a point mass in frame XYZ can be described by radius vector $\vec{r}$ (fig. 2.7).

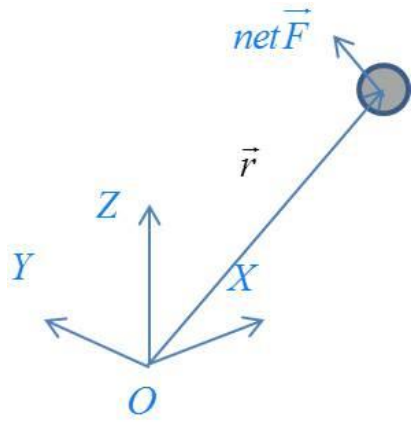

fig. 2.7 Position of the point mass in XYZ fixed reference frame

To apply eq. (2.1) to this point mass, a linear momentum of the point mass should be described:

$$\vec{p} = m * \vec{v}\ , (2.3)$$

where m is a mass of the point mass and $\vec{v}$ is a linear velocity of the point mass. With taking into account eq. (2.3), eq. (2.1) can be rewritten as:

$$net\vec{F} = m * \frac{d\vec{v}}{dt} = m * \frac{d^2\vec{r}}{dt}, (2.4).$$

The changing in position of the point can be expressed from eq. (2.4) as:

$$\ddot{\vec{r}} = \frac{net\vec{F}}{m}, \text{ (2.5).}$$

Scalar form of eq. (2.5) is:

$$\begin{aligned} \ddot{r}_X &= \frac{netF_X}{m} \\ \ddot{r}_Y &= \frac{netF_Y}{m} \\ \ddot{r}_Z &= \frac{netF_Z}{m} \end{aligned}, \text{ (2.6).}$$

Angular rotation around a fixed point

As it was mentioned above, for describing free motion of a body, an angular rotation around a fixed point should be considered. Assume a fixed reference frame xyz with origin in a fixed point of a rigid body (fig. 2.8).

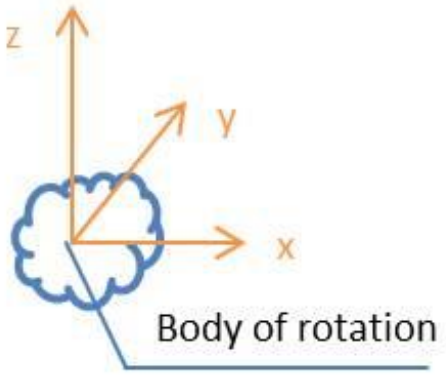

fig. 2.8 Rotation of a rigid body around a fixed point in xyz fixed reference frame

Rotation of the rigid body with random shape around the fixed point can be described in fixed reference frame xyz. For using eq. (2.2) the angular momentum of the body should be calculated.

The body is considered as a system of point masses, where mass of each point is dm and position of any of these points can be determined by vector $\vec{\rho}$ from origin of xyz till the element dm. In this case the angular momentum is:

$$\vec{L} = \iiint \vec{\rho} \times \vec{v}_{dm} * dm \text{, (2.7), [9]}$$

where $\iiint$ is an integral in the volume of the body, $\vec{v}_{dm}$ is a linear velocity of particular element. Eq. (1.7) in non-vector form is:

$$\begin{aligned} L_x &= \iiint \left(y * v_z - z * v_y \right) * dm \\ L_y &= \iiint \left(z * v_x - x * v_z \right) * dm \text{ , (2.8), [9]} \\ L_z &= \iiint \left(x * v_y - y * v_x \right) * dm \end{aligned}$$

where x, y, z are coordinates of an element dm (or in other words coordinates of correspondent radius vector $\vec{\rho}$), v_x ,v_y, v_z are projections of the velocity of this element.

Linear velocity of each element $\vec{v}_{dm}$ can be described as:

$$\vec{v}_{el} = \vec{\omega} \times \vec{\rho} = \begin{pmatrix} \omega_y * z - \omega_z * y \\ \omega_z * x - \omega_x * z \\ \omega_x * y - \omega_y * x \end{pmatrix} \text{, (2.9), [9]}$$

where $\vec{\omega}$ is angular velocity of the body and $\omega_x, \omega_y, \omega_z$ are its projections in xyz.
By rewriting eq. (2.7) with taking into account (2.9) the angular momentum is:

$$L_x = \iiint \left(y * (\omega_x * y - \omega_y * x) - z * (\omega_z * x - \omega_x * z) \right) * dm = \iiint \left(\omega_x * (y^2 + z^2) - \omega_y * x * y - \omega_z * z * x \right) * dm =$$
$$= \omega_x * \iiint (y^2 + z^2) * dm - \omega_y * \iiint (x * y) * dm - \omega_z * \iiint (z * x) * dm \text{ ,}$$

$$L_y = \iiint \left(z * (\omega_y * z - \omega_z * y) - x * (\omega_x * y - \omega_y * x) \right) * dm = \iiint \left(\omega_y * (z^2 + x^2) - \omega_z * z * y - \omega_x * x * y \right) * dm =$$
$$= \omega_y * \iiint (z^2 + x^2) * dm - \omega_z * \iiint z * y * dm - \omega_x * \iiint x * y * dm \text{ ,}$$

$$L_z = \iiint \left(x * (\omega_z * x - \omega_x * z) - y * (\omega_y * z - \omega_z * y) \right) * dm = \iiint \left(\omega_z * (x^2 + y^2) - \omega_x * x * z - \omega_y * y * z \right) * dm =$$
$$= \omega_z * \iiint (x^2 + y^2) * dm - \omega_x * \iiint x * z * dm - \omega_y * \iiint y * z * dm \text{ ,}$$

or in vector form:

$$\vec{L}=\begin{pmatrix}L_x\\L_y\\L_z\end{pmatrix}=\begin{pmatrix}\iiint(y^2+z^2)*dm & -\iiint(x*y)*dm & -\iiint(z*x)*dm\\ -\iiint(x*y)*dm & \iiint(z^2+x^2)*dm & -\iiint z*y*dm\\ -\iiint(z*x)*dm & -\iiint z*y*dm & \iiint(x^2+y^2)*dm\end{pmatrix}*\begin{pmatrix}\omega_x\\\omega_y\\\omega_z\end{pmatrix}=J*\vec{\omega},\ (2.10)\ ,\ [9]$$

where J is so-called tensor of inertia (ToI) and its components can be rewritten as follow:

$$J=\begin{pmatrix}\iiint(y^2+z^2)*dm & -\iiint(x*y)*dm & -\iiint(z*x)*dm\\ -\iiint(x*y)*dm & \iiint(z^2+x^2)*dm & -\iiint z*y*dm\\ -\iiint(z*x)*dm & -\iiint z*y*dm & \iiint(x^2+y^2)*dm\end{pmatrix}=\begin{pmatrix}J_{xx} & J_{xy} & J_{xz}\\ J_{yx} & J_{yy} & J_{yz}\\ J_{zx} & J_{zy} & J_{zz}\end{pmatrix},\ (2.11)\ ,\ [9]$$

where 1st index of J corresponds to the index of $\vec{L}$ and second one to the index of $\vec{\omega}$ and $J_{xy}=J_{yx};J_{xz}=J_{zx};J_{yz}=J_{zy};$.

Each component of inertia tensor is a moment of inertia (MoI) around particular axis. These components are constant, since origin of reference frame xyz is connected to the body. Tensor of inertia (ToI) can be simplified in a case if axes of reference frame xyz are coincident with principal axes of the body (axes of symmetry). To provide this case for rotating body, the axes of the reference frame should be fixed with the body. Assume new reference frame $x_by_bz_b$, which axes are coincidence with principal axes of the body and origin is in the fixed point of the body. In reference frame '$x_by_bz_b$' components $J_{xy}=J_{xz}=J_{yz}=0$ [9] and eq. (2.10) and eq. (2.11) can be rewritten as:

$$\vec{L}=\begin{pmatrix}L_{x_b}\\L_{y_b}\\L_{z_b}\end{pmatrix}=\begin{pmatrix}J_{x_bx_b} & 0 & 0\\ 0 & J_{y_by_b} & 0\\ 0 & 0 & J_{z_bz_b}\end{pmatrix}*\begin{pmatrix}\omega_{x_b}\\\omega_{y_b}\\\omega_{z_b}\end{pmatrix}=J*\vec{\omega},\ (2.12),\ [9]$$

To derive the angular momentum, an equation for the relative motion is used:

$$\dot{\vec{r}}_a=\dot{\vec{r}}_b+\vec{\omega}\times\vec{r},\ (2.13),\ [1]$$

where $\dot{\vec{r}}_a$ is an arbitrary vector in inertial reference frame, $\dot{\vec{r}}_b$ is the same vector in body (non - inertial) reference frame, $\vec{\omega}$ is an angular velocity of the body

reference frame in the fixed reference frame. Derivation of angular momentum $\frac{d\vec{L}}{dt}$ according to eq. (2.13), leads to so-called Euler's equation [2]:

$$net\vec{\tau} = \frac{d\vec{L}}{dt} = \frac{d(J*\vec{\omega})}{dt} + \vec{\omega}\times J*\vec{\omega} = J*\frac{d\vec{\omega}}{dt} + \vec{\omega}\times J*\vec{\omega}, \text{ (2.14), [2].}$$

Eq. (2.14) in more detail form is:

$$net\vec{\tau} = \begin{pmatrix} net\tau_{x_b} \\ net\tau_{y_b} \\ net\tau_{z_b} \end{pmatrix} = \begin{pmatrix} J_{x_bx_b} & 0 & 0 \\ 0 & J_{y_by_b} & 0 \\ 0 & 0 & J_{z_bz_b} \end{pmatrix} * \begin{pmatrix} \dot{\omega}_{x_b} \\ \dot{\omega}_{y_b} \\ \dot{\omega}_{z_b} \end{pmatrix} + \begin{pmatrix} \omega_{x_b} \\ \omega_{y_b} \\ \omega_{z_b} \end{pmatrix} \times \begin{pmatrix} J_{x_bx_b} & 0 & 0 \\ 0 & J_{y_by_b} & 0 \\ 0 & 0 & J_{z_bz_b} \end{pmatrix} * \begin{pmatrix} \omega_{x_b} \\ \omega_{y_b} \\ \omega_{z_b} \end{pmatrix} =$$

$$\begin{pmatrix} J_{x_bx_b} * \dot{\omega}_{x_b} \\ J_{y_by_b} * \dot{\omega}_{y_b} \\ J_{z_bz_b} * \dot{\omega}_{z_b} \end{pmatrix} + \begin{pmatrix} \omega_{x_b} \\ \omega_{y_b} \\ \omega_{z_b} \end{pmatrix} \times \begin{pmatrix} J_{x_bx_b} * \omega_{x_b} \\ J_{y_by_b} * \omega_{y_b} \\ J_{z_bz_b} * \omega_{z_b} \end{pmatrix} = \begin{pmatrix} J_{x_bx_b} * \dot{\omega}_{x_b} \\ J_{y_by_b} * \dot{\omega}_{y_b} \\ J_{z_bz_b} * \dot{\omega}_{z_b} \end{pmatrix} + \begin{pmatrix} \left(J_{z_bz_b} - J_{y_by_b}\right) * \omega_{y_b} * \omega_{z_b} \\ \left(J_{x_bx_b} - J_{z_bz_b}\right) * \omega_{x_b} * \omega_{z_b} \\ \left(J_{y_by_b} - J_{x_bx_b}\right) * \omega_{x_b} * \omega_{y_b} \end{pmatrix},$$

or in short form:

$$\begin{aligned} net\tau_{x_b} &= J_{x_bx_b} * \dot{\omega}_{x_b} + \left(J_{z_bz_b} - J_{y_by_b}\right) * \omega_{y_b} * \omega_{z_b} \\ net\tau_{y_b} &= J_{y_by_b} * \dot{\omega}_{y_b} + \left(J_{x_bx_b} - J_{z_bz_b}\right) * \omega_{x_b} * \omega_{z_b} \\ net\tau_{z_b} &= J_{z_bz_b} * \dot{\omega}_{z_b} + \left(J_{y_by_b} - J_{x_bx_b}\right) * \omega_{x_b} * \omega_{y_b} \end{aligned} \text{ , (2.16), [2].}$$

Eq. (2.16) should be rewritten in the form for finding changing in angular velocity as:

$$\dot{\omega}_{x_b} = \frac{net\tau_{x_b} - \left(J_{z_b z_b} - J_{y_b y_b}\right) * \omega_{y_b} * \omega_{z_b}}{J_{x_b x_b}}$$

$$\dot{\omega}_{y_b} = \frac{net\tau_{y_b} - \left(J_{x_b x_b} - J_{z_b z_b}\right) * \omega_{x_b} * \omega_{z_b}}{J_{y_b y_b}}, \text{(2.17), [2].}$$

$$\dot{\omega}_{z_b} = \frac{net\tau_{z_b} - \left(J_{y_b y_b} - J_{x_b x_b}\right) * \omega_{x_b} * \omega_{y_b}}{J_{z_b z_b}}$$

As it was mentioned above, description of rotation in body reference frame $x_b y_b z_b$ instead of fixed reference frame xyz simplifies calculation of angular momentum $\vec{L}$ to eq. 2.12. On the other hand because of this simplification, another equation, that links angular velocity $\vec{\omega}$ in $x_b y_b z_b$ and orientation $x_b y_b z_b$ relatively to xyz, is also needed. Orientation of the body is defined by unique rotation around instantaneous axis of rotation. This rotation can be considered as sum of three simple rotations. Sequences of simple rotations are not unique [4]. Commonly used sequence in Aerospace applications for flying objects is yaw-pitch-roll (YPR) rotation, where angles e.g. noted as ψ, θ, ϕ correspondingly (fig. 2.9).

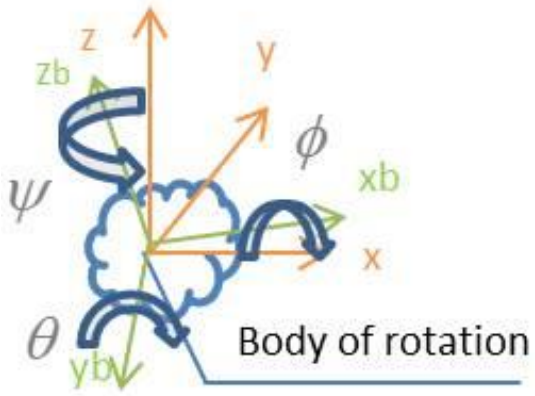

fig. 2.9 Orientation of the body reference frame $x_b y_b z_b$ by yaw-pitch-roll angles in fixed reference frame xyz

Changes in orientation are connected with $\vec{\omega}$ by following equation:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & sn\phi * t\theta & c\phi * t\theta \\ 0 & c\phi & -sn\phi \\ 0 & \frac{sn\phi}{c\theta} & \frac{c\phi}{c\theta} \end{pmatrix} * \begin{pmatrix} \omega_{x_b} \\ \omega_{y_b} \\ \omega_{z_b} \end{pmatrix}, \text{(2.18). [5]}$$

Hereby, for a block '6DOF' (from fig. 2.2) three equations are needed: eq. (2.6) describes dynamics of linear motion, eq. (2.18) describes kinematics of angular motion and eq. (2.17) describes dynamics of angular motion.

Tensor of inertia and mass of the quadrocopter

For calculation ToI, a real structure of the quadrocopter (fig. 2.1) is simplified to the structure, which consists of spherical dense center with mass M , radius R and several point masses of mass m_M located at distance l (fig. 2.10).[5]

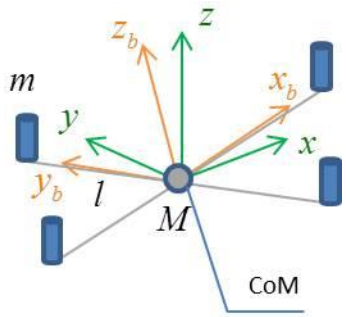

fig. 2.10 Simplified structure of the quadrocopter

ToI of the simplified structure based on eq. (2.11) and eq. (2.12) can be described as:

$$J = \begin{pmatrix} \iiint (y_b^{\,2} + z_b^{\,2}) * dm & 0 & 0 \\ 0 & \iiint (z_b^{\,2} + x_b^{\,2}) * dm & 0 \\ 0 & 0 & \iiint (x_b^{\,2} + y_b^{\,2}) * dm \end{pmatrix} = \begin{pmatrix} J_{x_b x_b} & 0 & 0 \\ 0 & J_{y_b y_b} & 0 \\ 0 & 0 & J_{z_b z_b} \end{pmatrix}$$

, (2.19)

where $J_{x_b x_b} = J_{y_b y_b} = \frac{2*M*R^2}{5} + 2*l^2*m_M$ and $J_{z_b z_b} = \frac{2*M*R^2}{5} + 4*l^2*m_M$.

Eq. (2.19) supplements eq. (2.17).

In case of chosen approximation the mass of quadrocopter is:

$$m = M + 4*m_M \text{ , (2.20),}$$

where m_M is total mass of the motor and the blade, M is mass of the spherical dense or in other words mass of the rest parts of the quadrocopter.

Eq. (2.14) supplements eq. (2.6).

Summary of equations for motion in 6DOF

Hereby, based on written in section 2.1, free motion of the quadrocopter can be represented in reference frame XYZ (fig. 2.11).

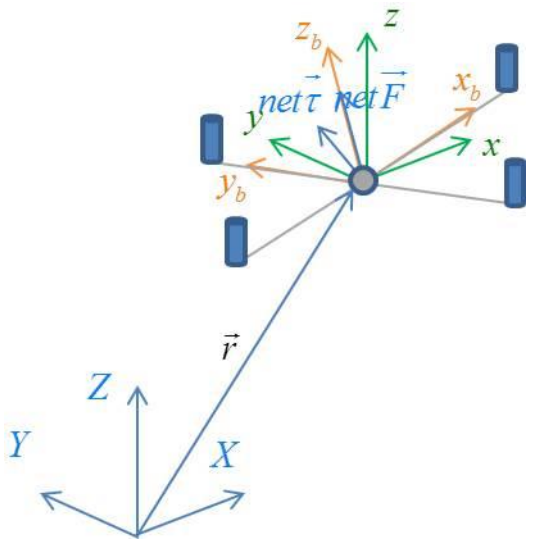

fig. 2.11 The quadrocopter's free motion

Parameters and functions of this motion can be determined in several steps. Initial conditions are set accordingly to current experiment (e.g. equal to zero). Then mass and ToI are calculated based on eq. (2.19) and eq.(2.20). Afterwards dynamics and kinematics of orientation are calculated based on eq. (2.18) and eq. (2.17) and dynamics of linear motion is calculated based on eq. (2.6).

2.2 External Forces

A position and an orientation of the quadrocopter can be found by eq. (2.6) and eq. (1.17) when the external net force $net\vec{F}$ and external net torque $net\vec{\tau}$ are known. As it was mentioned before there are three sources of external forces: gravitational field, air drag and rotations of the blades in the air, which lead to a gravitational force $\vec{F}_{mg}$, a drag force $\vec{F}_{drag}$ and a thrust force $\vec{T}$ respectively. Forces and torques from blades can be controlled. Gravitational and drag force cannot be controlled.

Gravitational forces

Forces of gravitational field applied to a body can be represented as a net gravitational force applied to a center of gravity (CoG) of the body. Direction of this force is constant and pointed to the center of the Earth. Relation between this force and acceleration of an object corresponds to Newton law and can be written as:

$$\vec{F}_{mg} = m * \vec{g}, \text{ (2.21)}$$

where m is mass of the quadrocopter and $\vec{g}$ is gravitational acceleration.

For simplification assume that the CoG coincides with the CoM of the quadrocopter.

Air drag force

When an object moves through the air, it overcomes air resistant. Air drag force can be described as:

$$F_{drag} = C_d * \rho * S * v^2, \text{ (2.22),[8]}$$

where C_d is drag coefficient, ρ is mass density of the air fluid, S reference area of the object, v is the speed of the object relative to the air fluid.

Coefficient C_d, which depends on the shape of the object, should be measured in advance in a wind tunnel. Drag coefficients for several shapes are well known and available in the form of the tables, e.g. for square shape C_d equals to 0.64 [8] . Mass density of the air ρ depends on the height above the see level, e.g. for 1

meter above sea level with temperature about 15 degree, air density equals to 1.226 [8].

For movement in 3D eq. (2.22) can be rewritten as:

$$\vec{F}_{drag} = C_d * \rho * S * |v| * \vec{v}, \text{ (2.23),}$$

Forces from blades

Interaction between a rotating blade and air can be described by vortex theory [7]. Assume that a blade is rotating with some angular velocity ω_{bl} in counterclockwise direction. This rotation leads to producing a number of forces. To find net forces, the blade surface is theoretically divided by small elements and the force that applied to an element represented as sum of vertical force $\vec{T}_{el}$ and horizontal force $\vec{Q}_{el}$ (fig. 2.12). Sum of all vertical forces $\vec{T}_{el}$ can be substituted by thrust force $\vec{T}$ and sum of all horizontal force $\vec{Q}_{el}$ as hub torque $\vec{H}$.

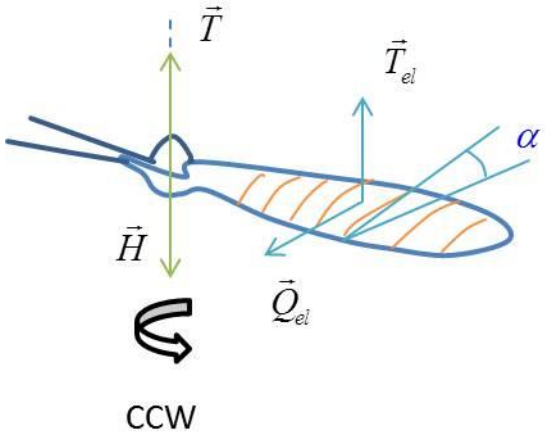

fig. 2.12 Net force and torque from interaction between a blade and the air

The thrust force $\vec{T}$ and hub torque $\vec{H}$ can be described as:

$$\vec{T} = b_T * \omega_{bl}^2, \text{ (1.24), [21]}$$

$$\vec{H} = b_H * \omega_{bl}^2, \text{ (1.25) , [21]}$$

where b_p and b_b are proportional coefficients, which depends on air density, angle of blade and area of blade.

The quadrocopter has four actuators; each of them consists of a blade, a motor and a power bridge. Notate linear movement of the quadrocopter as forward, backward, left and right and number actuators from 1 to 4 (fig. 2.13). Blades 2 and 4 rotate in counterclockwise direction with angular speed ω_2, ω_4 while blades 1 and 3 rotate in clockwise direction with angular speed ω_1, ω_3.

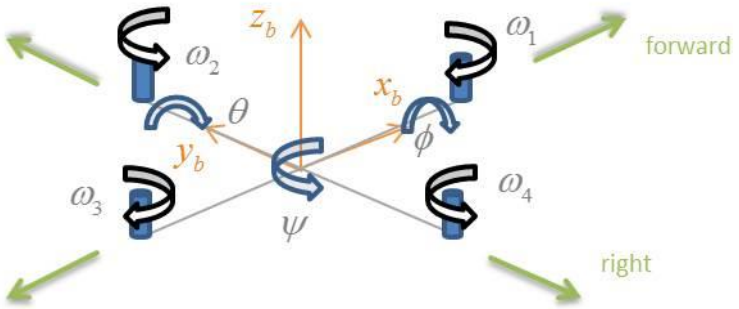

fig. 1.13 Quadrocopter structure

These rotations lead to four couples of thrust forces and hub torques (fig. 2.14).

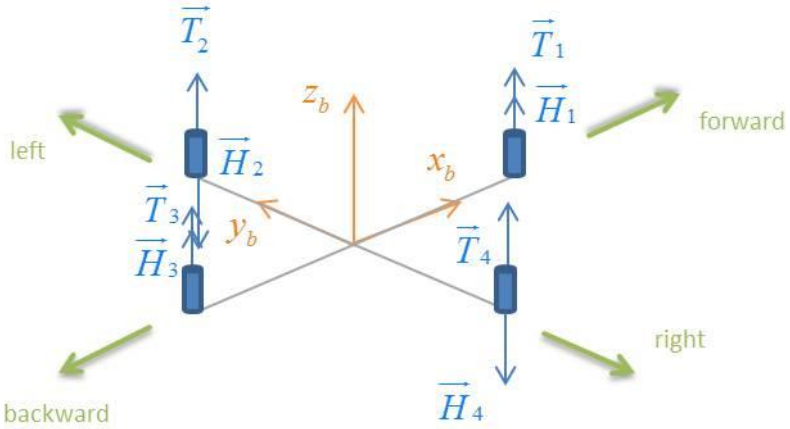

fig. 2.14 Thrust and hub forces from each blade

These 4 forces can be replaced by net thrust force $\vec{T}$ and 4 hub torques by net hub torque $\vec{H}$ (fig. 2.15).

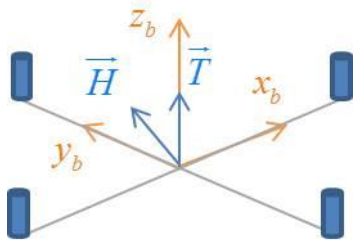

fig. 2.15 Thrust force and hub torque applied to the quadrocopter (an arbitrary direction of $\vec{H}$ is chosen)

Thrust force $\vec{T}$ can be represented as:

$$\vec{T} = \vec{T}_1 + \vec{T}_2 + \vec{T}_3 + \vec{T}_4, \text{ (2.26) , [21].}$$

Hub torque $\vec{H}$ can be represented as:

$$\vec{H} = \begin{pmatrix} H_{x_b} \\ H_{y_b} \\ H_{z_b} \end{pmatrix} = \begin{pmatrix} l*\left(\vec{T}_2 - \vec{T}_4\right) \\ l*\left(\vec{T}_3 - \vec{T}_1\right) \\ \left(\vec{H}_1 + \vec{H}_3\right) - \left(\vec{H}_2 + \vec{H}_4\right) \end{pmatrix}, \text{ (2.27) , [21]}$$

where sign minus corresponds to negative direction of roll, pitch and yaw angles.

Equation (2.27) supplements eq. (2.17) for calculation orientation of the quadrocopter, so (2.17) can be rewritten as:

$$\dot{\omega}_{x_b} = \frac{l*\left(\vec{T}_2 - \vec{T}_4\right) - \left(J_{z_b z_b} - J_{y_b y_b}\right)*\omega_{y_b}*\omega_{z_b}}{J_{x_b x_b}}$$

$$\dot{\omega}_{y_b} = \frac{l*\left(\vec{T}_3 - \vec{T}_1\right) - \left(J_{x_b x_b} - J_{z_b z_b}\right)*\omega_{x_b}*\omega_{z_b}}{J_{y_b y_b}} \quad \text{, (2.28).}$$

$$\dot{\omega}_{z_b} = \frac{\left(H_1 + H_3\right) - \left(H_2 + H_4\right) - \left(J_{y_b y_b} - J_{x_b x_b}\right)*\omega_{x_b}*\omega_{y_b}}{J_{z_b z_b}}$$

For finding translation motion thrust force $\vec{T}$ should be represented in *xyz* as:

$$\vec{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = R_{xyz}^{x_b y_b z_b} * \vec{T}_{x_b y_b z_b}, \text{ (2.29),}$$

where $R_{xyz}^{x_b y_b z_b}$ is a rotation matrix.

Rotation matrix of YPR is:

$$R_{xyz}^{x_b y_b z_b} = \left[R_x(\phi) * R_y(\theta) * R_z(\psi) \right]^{-1} =$$

$$= \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & c\phi & sn\phi \\ 0 & -sn\phi & c\phi \end{pmatrix} * \begin{pmatrix} c\theta & 0 & -sn\theta \\ 0 & 1 & 0 \\ sn\theta & 0 & c\theta \end{pmatrix} * \begin{pmatrix} c\psi & sn\psi & 0 \\ -sn\psi & c\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \right]^{-1} =$$

$$= \left[\begin{pmatrix} c\theta * c\psi & c\theta * sn\psi & -s\theta \\ sn\phi * sn\theta * c\psi - c\phi * sn\psi & sn\phi * sn\theta * sn\psi + c\phi * c\psi & sn\phi * c\theta \\ c\phi * sn\theta * c\psi + sn\phi * sn\psi & c\phi * sn\theta * sn\psi - sn\phi * c\psi & c\phi * c\theta \end{pmatrix} \right]^{-1} =$$

$$\begin{pmatrix} c\theta * c\psi - sn\phi * sn\theta * sn\psi & -c\theta * sn\psi & c\psi * sn\theta + c\theta * sn\phi * sn\psi \\ c\phi * sn\psi + sn\phi * sn\theta * c\psi & c\phi * c\psi & sn\theta * sn\psi - sn\phi * c\theta * c\psi \\ -c\phi sn\theta & sn\phi & c\phi * c\theta \end{pmatrix}, \text{ (2.30) , [4]}$$

where c and sn are abbreviations for cosine and for sine respectively.

Thrust force is always aligned with z_b axis, therefore, with taking into account eq. (2.30), eq. (2.29) can be rewritten as:

$$\vec{T}_{xyz} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = R_{xyz}^{x_b y_b z_b} * \vec{T} = R_{xyz}^{x_b y_b z_b} * \begin{pmatrix} 0 \\ 0 \\ T \end{pmatrix} =$$

$$= \begin{pmatrix} c\theta * c\psi - sn\phi * sn\theta * s\psi & -c\theta * sn\psi & c\psi * sn\theta + c\theta * sn\phi * sn\psi \\ c\phi * sn\psi + sn\phi * sn\theta * c\psi & c\phi * c\psi & s\theta * sn\psi - sn\phi * c\theta * c\psi \\ -c\phi sn\theta & sn\phi & c\phi * c\theta \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ T \end{pmatrix} =$$

$$= \begin{pmatrix} c\psi * sn\theta + c\theta * sn\phi * sn\psi \\ s\theta * sn\psi - sn\phi * c\theta * c\psi \\ c\phi * c\theta \end{pmatrix} * T \text{ , (2.31).}$$

With taking into account eq. (2.31), gravitational force eq. (2.21) and drag force eq. (2.23), eq. (2.6) can be rewritten as:

$$\ddot{r}_X = \frac{\left(c\psi * sn\theta + c\theta * sn\phi * sn\psi\right) * T - C_d * \rho * S * v_X^2}{m}$$
$$\ddot{r}_Y = \frac{\left(s\theta * sn\psi - sn\phi * c\theta * c\psi\right) * T - C_d * \rho * S * v_Y^2}{m} \text{ , (2.32).}$$
$$\ddot{r}_Z = \frac{c\phi * c\theta * T - m * g - C_d * \rho * S * v_Z^2}{m}$$

Thereby, the free motion of the quadrocopter in 6DOF can be represented as linear motion its center of the mass (CoM) with mass m in XYZ and angular rotation of the quadrocopter around CoM. The rotation is described by rotation of the quadrocopter in $x_b y_b z_b$ and orientation of $x_b y_b z_b$ relatively to xyz, where axes of xyz are parallel to relative axes of XYZ (fig. 2.15).

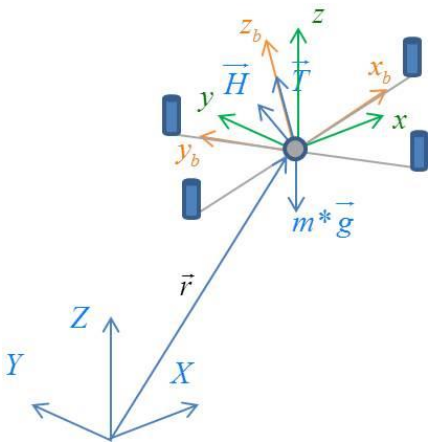

fig. 2.15 Motion of the quadrocopter in 6DOF

The influence from applied forces is described by eq. (2.32) (instead of (2.6)) and the influence from applied torques is described by eq. (2.28) (instead of 2.17).

2.2.3 Quadrocopter's actuators

The actuator consists of a blade, a motor and a power bridge. The behavior of the blade is described in previous section by eq. (2.24), eq. (2.25).

Angular velocities of the blades depend on angular velocities of corresponding motors. For calculation a shaft velocity of a brushless DC motor (BLDC) a mathematical model of the motor should be used. This model depends on motors construction. In opposite to brushed DC motor, a BLDC motor needs a control system for rotation of its rotor. Sometimes MM of BLDC is needed for creating such type of the control system. However, for current case the MM is needed to estimate relationship between the input and output. For this purpose MM of BLDC can be substituted by MM of brushed DC [6].

The MM of brushed DC motor is based on four equations:

$$U = i * R + L * \frac{di}{dt} + e_m, \text{ (2.33)}$$

$$e_m = k_\omega * \omega, \text{ (2.34)}$$

$$T = k_m * i, \text{ (2.35)}$$

$$T = J * \frac{d\omega}{dt}, \text{ (2.36).}$$

Eq. (2.33) describes the effect, when applied voltage leads to current in the armature with resistance R and to inductance L and to back EMF e_m. Eq. (2.34) indicates that back-EMF e_m proportional to angular velocity of the motor's shaft, where k_ω is back-EMF constant. Eq. (2.35) denotes that produced torque is proportional to the produced current, where k_m is the torque constant. Eq. (2.36) describes transferring from the torque to angular acceleration of the plant, where J is sum of the moment of inertia of the plant and motor shaft. For current case, the plant is the blade, which has minimal MoI, so the plant MoI can be omitted.

A power bridge can be represented as:

$$u = k_{PWM} * u_{\max}, \text{ (2.37)}$$

where u is input voltage to the BLDC , $u_{\max}$ is maximum input voltage of BLDC applied to the power bridge and k_{PWM} is percent of pulse-width modulation (PWM). Time delay of the power bridge can be neglected since time for changing the electrical signals is much smaller comparing to the time delay in mechanical part of the system. The power bridge can be considered as continuous element.

2.2.4 Discrete elements

The system has several discrete elements such as: a microcontroller and sensors. These elements are discrete in time and level. They should be substituted by quantizers with level of discretizations ld_M and ld_s correspond to their calculation precision and time discretizations td_M and td_s correspond to their delays in time.

These parameters can be calculated as:

$$ld_M = \frac{1}{2^{NmofBt}}; \quad ld_S = \frac{1}{2^{NmofBt}}, \text{ (2.38)}$$

where $NmofBt$ is length of the microcontroller's register in bites.

$$td_s = \frac{1}{f}, \text{ (2.39)}$$

where f is frequency of the sensor in Hz.

$$td_M = N * t_{ins}, \text{ (2.40)},$$

where N is number of instructions in a code and t_{ins} is time for fulfillment of one instruction.

2.3 Mathematical model of the quadrocopter

The MM of the quadrocopter consists of transfer functions (TFs) of elements described in previous sections. Logical diagram of this model with corresponding equations is shown on fig. 2.16.

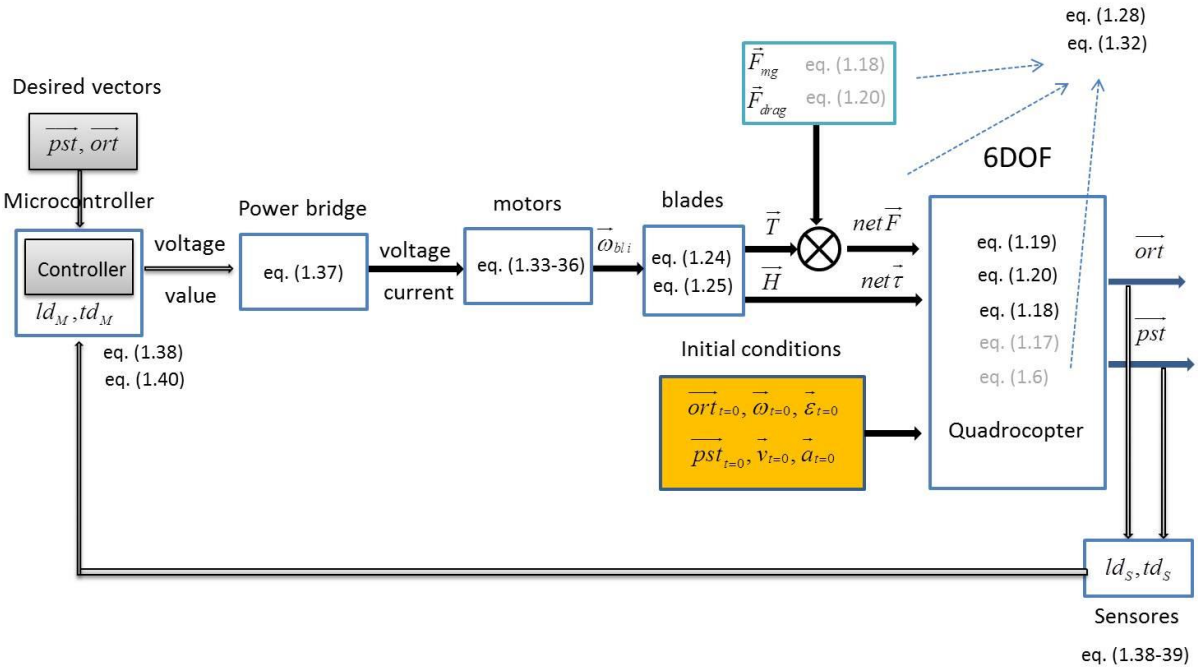

fig. 2.16 Logical diagram of MM of the quadrocopter in 6DOF

This model has the following simplifications:

- the quadrocopter is a rigid body
- center of mass (CoM) is in the geometrical center of the quadrocopter
- ToI of the quadrocopter is approximated as moment of inertia of several objects
- CoG is coincided with CoM
- MoI of the blades is neglected
- Time delay of the power bridge is neglected

For simulation purpose equations should be represented by transfer functions. TF will be represented in a form suitable for realization in Matlab/Simulink. TF for eq. (2.32) is:

$$r_X(s)=\frac{1}{s^2}\left(\frac{c\psi(s)*sn\theta(s)+c\theta(s)*sn\phi(s)*sn\psi(s)}{m}*T(s)+\frac{-C_d*\rho*S}{m}*(sr_X(s))^2\right)$$

$$r_Y(s)=\frac{1}{s^2}\left(\frac{s\theta(s)*sn\psi(s)-sn\phi(s)*c\theta(s)*c\psi(s)}{m}*T(s)+\frac{-C_d*\rho*S}{m}*(sr_Y(s))^2\right) \quad , (2.41)$$

$$r_Z(s)=\frac{1}{s^2}\left(\frac{c\phi(s)*c\theta(s)*}{m}*T(s)-g+\frac{-C_d*\rho*S}{m}*(sr_Z(s))^2\right)$$

where s is Laplace operator.

This TF is nonlinear since it has trigonometric functions and squaring of the inputs.

Based on eq. (2.41) it can be represented in Matlab/Simulink (fig. 2.17)

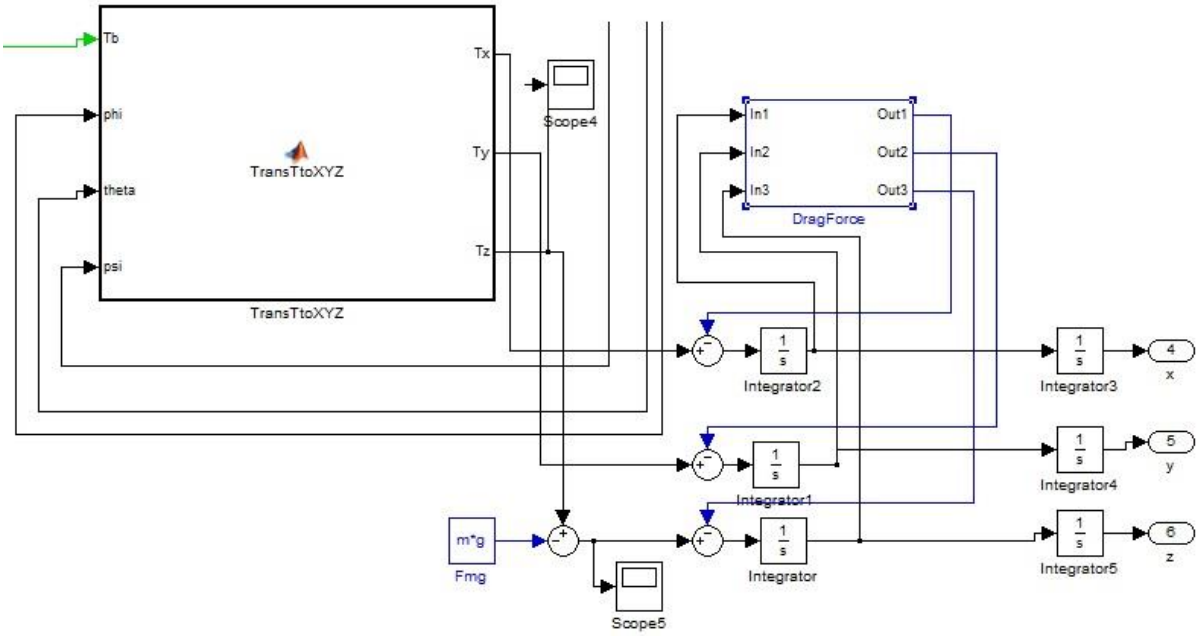

fig. 2.17 Representation of TF for translation motion in Matlab/Simulink

Block 'TransTtoXYZ' (fig. 2.17) has as inputs the thrust force $T(s)$ and orientation angles ψ,θ,φ. These inputs shoud be calculated for finding a quadrocopter's position. The TFs for orientation angles are calculated based on eq. (2.18) and eq. (2.28) as:

$$s\begin{pmatrix}\phi(s)\\ \theta(s)\\ \psi(s)\end{pmatrix}=\begin{pmatrix}1 & sn\phi(s)*t\theta(s) & c\phi(s)*t\theta(s)\\ 0 & c\phi(s) & -sn\phi(s)\\ 0 & \dfrac{sn\phi(s)}{c\theta(s)} & \dfrac{c\phi(s)}{c\theta(s)}\end{pmatrix}*\begin{pmatrix}\omega_{x_b}(s)\\ \omega_{y_b}(s)\\ \omega_{z_b}(s)\end{pmatrix}, \quad (2.42)$$

$$s\omega_{x_b}(s)=\frac{l}{J_{x_bx_b}}*\left(\vec{T}_2(s)-\vec{T}_4(s)\right)+\frac{-\left(J_{z_bz_b}-J_{y_by_b}\right)}{J_{x_bx_b}}*\omega_{y_b}(s)*\omega_{z_b}(s)$$

$$s\omega_{y_b}(s)=\frac{l}{J_{y_by_b}}*\left(\vec{T}_3(s)-\vec{T}_1(s)\right)+\frac{-\left(J_{x_bx_b}-J_{z_bz_b}\right)}{J_{y_by_b}}*\omega_{x_b}(s)*\omega_{z_b}(s) \quad ,(2.43).$$

$$s\omega_{z_b}(s)=\frac{1}{J_{z_bz_b}}*\left(H_1(s)+H_3(s)\right)-\left(H_2(s)+H_4(s)\right)+\frac{-\left(J_{y_by_b}-J_{x_bx_b}\right)}{J_{z_bz_b}}*\omega_{x_b}(s)*\omega_{y_b}(s)$$

These TFs are nonlinear since they have trigonometric functions and multiplying of the outputs. They can be represented in Matlab/Simulink as shown on fig. (2.18).

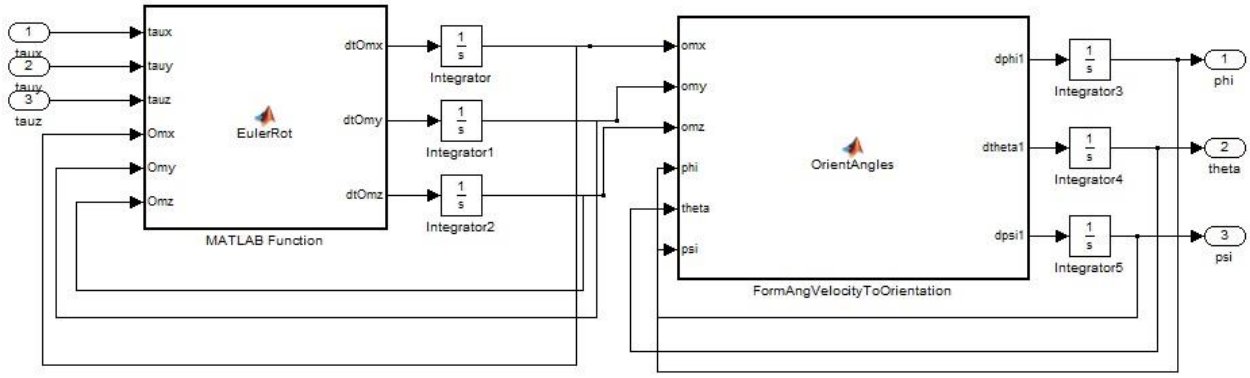

fig. 2.18 Representation of TF for angular rotation in Matlab/Simulink

The input of this block is hub torque $H(s)$. The inputs $T(s)$ and $H(s)$ can be calculated based on eq., (2.24), (2.25), (2.26), (2.27). The TFs are:

$$T_1(s)=b_T*{\omega_{bl1}}^2(s);T_2(s)=b_T*{\omega_{bl2}}^2(s);T_3(s)=b_T*{\omega_{bl3}}^2(s);T_4(s)=b_T*{\omega_{bl4}}^2(s), \quad (2.44)$$

$$H_1(s)=b_H*{\omega_{bl1}}^2(s);H_2(s)=b_H*{\omega_{bl2}}^2(s);H_3(s)=b_H*{\omega_{bl3}}^2(s);H_4(s)=b_H*{\omega_{bl4}}^2(s) \quad , (2.45)$$

$$T(s) = b_T * \left({\omega_{bl1}}^2(s) + {\omega_{bl2}}^2(s) + {\omega_{bl3}}^2(s) + {\omega_{bl4}}^2(s) \right), \text{ (2.46)}$$

$$\begin{pmatrix} H_{x_b}(s) \\ H_{y_b}(s) \\ H_{z_b}(s) \end{pmatrix} = \begin{pmatrix} l * \left(\vec{T}_2(s) - \vec{T}_4(s) \right) \\ l * \left(\vec{T}_3(s) - \vec{T}_1(s) \right) \\ \left(\vec{H}_1(s) + \vec{H}_3(s) \right) - \left(\vec{H}_2(s) + \vec{H}_4(s) \right) \end{pmatrix}, \text{ (2.47).}$$

These TFs are nonlinear since they have squaring outputs. They can be represented in Matlab/Simulink as it shown on fig. (2.19).

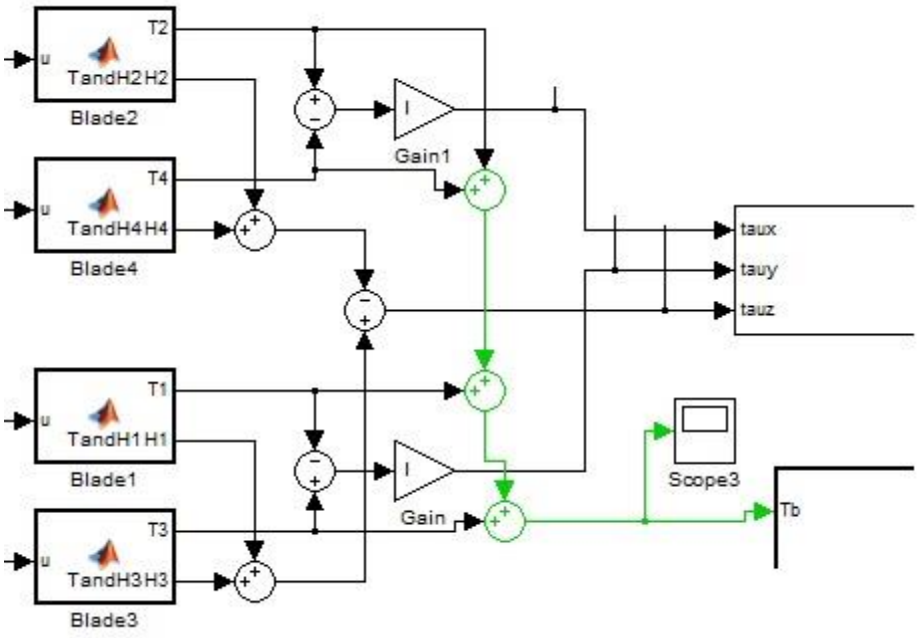

fig. 2.19 Representation of TFs in Matlab/Simulink for creating $T(s)$ and $H(s)$

MM of BLDC is designed based on eqs. (2.33 - 36). Applying Laplace transformation to eq. (2.33), (2.34) leads to:

$$\frac{U(s) - k_\omega * \omega(s)}{R + sL} = I(s) \text{ , (2.48)}$$

and to eq. (2.35), (2.36) leads to:

$$k_m * I(s) = J * s\omega(s) \text{ . (2.49)}$$

Transfer function can be obtained by combining eq. (2.48) and (2.49) as:

$$k_m * \frac{U(s) - k_\omega * \omega(s)}{R + sL} = J * s\omega(s) \Rightarrow U(s) = \frac{J}{k_m} * \left(R + sL\right) * s\omega(s) + k_\omega * \omega(s)$$

$$\Rightarrow U(s) * \frac{1}{\frac{J * L}{k_m} s^2 + \frac{J * R}{k_m} * s + k_\omega} = \omega(s), \text{ (2.50)}$$

where $G(s) = \frac{1}{\frac{J * L}{k_m} s^2 + \frac{J * R}{k_m} * s + k_\omega}$ is 2nd order TF.

More common form of eq. (2.50), where parameters J, L and k_m are substituted, is represented in more appropriate form as:

$$G(s) = \frac{1}{\frac{J * L}{k_m} s^2 + \frac{J * R}{k_m} * s + k_\omega} = \frac{\frac{1}{k_\omega}}{\frac{J * R}{k_m * k_\omega} * \frac{L}{R} * s^2 + \frac{J * R}{k_m * k_\omega} * s + 1} = \frac{\frac{1}{k_\omega}}{\tau_m * \tau_e * s^2 + \tau_m * s + 1}$$

or in short way

$$G(s) = \frac{\frac{1}{k_\omega}}{\tau_m * \tau_e * s^2 + \tau_m * s + 1}, \text{ (2.51)}$$

where $\tau_m = \frac{J * R}{k_m * k_\omega}$ is mechanical constant and $\tau_e = \frac{L}{R}$ is electrical constant.

Implementation of TF based on eq. (2.51) is shown on fig. 2.20.

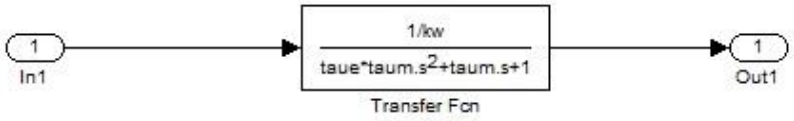

fig. 2.20 Representation of BLDC TF in Simulink

Thereby the TF functions of continuous elements of the quadrocopter can be represented as shown on fig. 2.21.

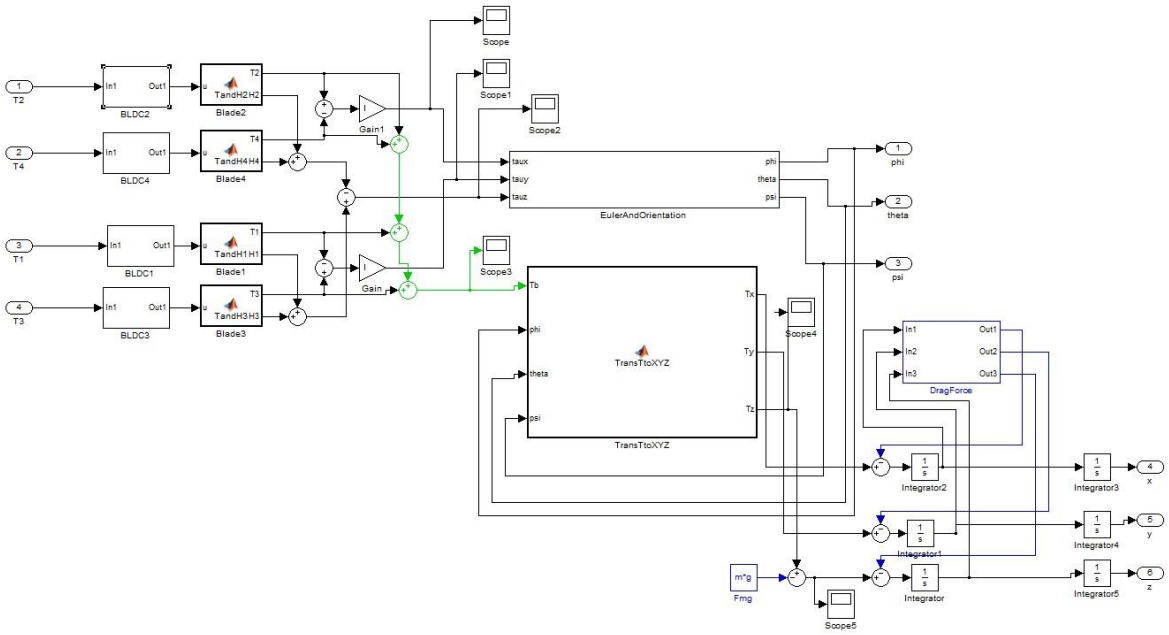

fig. 2.21 Representation of TF of continuous part in Matlab/Simulink

Chapter 3 Design of the Control System

A goal of a controller design is to reach a new plant's behavior, which corresponds to desired quality requirements. Any controller design procedure is based on information how inputs and outputs of the plant are connected. In most cases relation between inputs and outputs is described by mathematical model of the plant. The MM can be created by three methods: mathematical description of physical processes, identification procedure and a sum of two previous methods. MMs obtained based on these procedures are called white box, black box and grey box respectively [12]. Most of design procedures are based on the linear MM of the plant and most popular procedures are root-locus method, pole placement method and by Bode diagrams. As soon as the quadrocopter should have intelligent control system, the pole-placement method from mentioned one is chosen.

3.1 Pole-placement method: Ackermann approach

Pole placement method is a method, where a designed controller should change poles of characteristic equation of the MM to the poles that give the system desired quality such as a settling time, an overshoot, a steady state error.

However, it should be mentioned that this method has serious limitations such as: sensitiveness to how adequate the model is and not observability of particular connected to real physical parameters. Advantage of pole placement method is that the controller designed by this method can be easily expanded to an optimal or adaptive controller.

One of the simplest and direct ways to design a system with chosen poles is using Ackerman equation. This equation transfers state space model in control canonical form and calculate new coefficients for feedbacks [11]. In general the TF look like:

$$G_p(s) = \frac{b_{n-1} * s^{n-1} + \cdots + b_1 * s + b_0}{s^n + a_{n-1} * s^{n-1} + \cdots + a_1 * s + a_0}, \quad (3.1).$$

The control canonical form for a 3rd order system is represented on fig. 3.1.

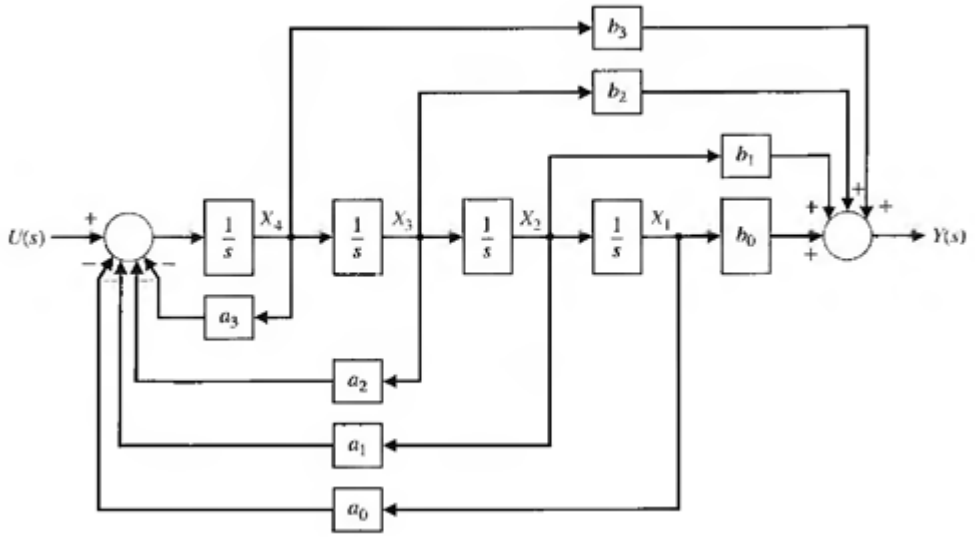

fig. 3.1 Structure diagram of control canonical form [10]

The state equations for n-order system in control canonical form are:

$$\begin{aligned} \dot{x}(t) &= A * x(t) + B * u(t) \\ y(t) &= C * x(t) + D * u(t) \end{aligned}, \quad (3.2)$$

where $A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} b_0 & b_1 & b_2 & \dots & b_{n-1} \end{bmatrix}$,

$D = 0$.

The idea of Ackerman approach is to calculate new coefficients K that supplement existing coefficients to make roots of a closed-loop system equal to desired poles. Consider the case when the input of the system is zero, so called regulator control. In this case $u(t)$ consists only of feedback signals and can be described as:

$$u(t) = -Kx(t), \quad (3.3).$$

So close-loop matrix A_f can be written by substituting result of eq. (3.3) to eq. (3.2):

$$A_f = (A - BK) = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} K_1 & K_2 & \dots & K_{n-1} & K_n \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 - K_1 & -a_1 - K_2 & -a_2 - K_3 & \dots & -a_{n-1} - K_n \end{bmatrix}, (3.4)$$

Matrix K can be calculated as:

$$K = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \end{bmatrix} * \begin{bmatrix} B & A*B & \dots & A^{n-2}*B & A^{n-1}*B \end{bmatrix}^{-1} * \alpha_c(A) \text{ , (3.5),}$$

where $\alpha_c(A)$ is matrix polynomial formed with coefficients of the desired characteristic equation $\alpha_c(s)$. The desired closed-loop characteristic equation can be described as:

$$\alpha_c(s) = s^n + \alpha_{n-1} * s^{n-1} + \dots + \alpha_1 * s + \alpha_0 = (s + \lambda_1) * (s + \lambda_2) \dots (s + \lambda_n) = 0 \text{ , (3.6)}$$

where $-\lambda_1 \dots - \lambda_n$ are desired poles.

Matrix polynomial $\alpha_c(A)$ is described as:

$$\alpha_c(A) = A^n + \alpha_{n-1} * A^{n-1} + \dots + \alpha_1 * A + \alpha_0 * I \text{ , (3.7),}$$

where I is identity matrix with dimension equals to dimension of A.

Thereby, pole placement procedure has three steps. First step is obtaining the MM of the system , where the MM should be linear continuous time invariant model. Next step is calculation desired poles based on requirements to quality of the

system. Last step is calculation feedback coefficients that change poles of the MM to desired ones.

3.2 Linear time-invariant mathematical model of the quadrocopter

The MM developed in chapter 2 cannot be used for controller design by pole placement method and should be simplified. One of the approaches is known as small disturbance theory [13]. Based on the assumption that motion of the flying object consists of small deviations around a steady flight conditions, multiplication of angular velocity components can be omitted and eq. (2.43) can be rewritten as:

$$\begin{aligned} s\omega_{x_b}(s) &= \frac{l}{J_{x_b x_b}} * \left(T_2(s) - T_4(s)\right) \\ s\omega_{y_b}(s) &= \frac{l}{J_{y_b y_b}} * \left(T_3(s) - T_1(s)\right) \\ s\omega_{z_b}(s) &= \frac{1}{J_{z_b z_b}} * \left(H_1(s) + H_3(s)\right) - \left(H_2(s) + H_4(s)\right) \end{aligned} \quad , (3.8).$$

Another simplification which leads from this fact is that changes in orientation angles equal to angular velocity and eq. (2.42) can be rewritten as:

$$s\begin{pmatrix} \phi(s) \\ \theta(s) \\ \psi(s) \end{pmatrix} = \begin{pmatrix} \omega_{x_b}(s) \\ \omega_{y_b}(s) \\ \omega_{z_b}(s) \end{pmatrix}, (3.9).$$

Combination of eq. (3.8) and eq. (3.9) can be shown as:

$$\begin{aligned} \phi(s) &= \frac{l}{s^2 J_{x_b x_b}} * \left(T_2(s) - T_4(s)\right) \\ \theta(s) &= \frac{l}{s^2 J_{y_b y_b}} * \left(T_3(s) - T_1(s)\right) \\ \psi(s) &= \frac{1}{s^2 J_{z_b z_b}} * \left(H_1(s) + H_3(s)\right) - \left(H_2(s) + H_4(s)\right) \end{aligned} \quad , (3.10).$$

Another assumption is that TFs between PWM values and forces/torques can be substituted by proportional coefficient or (if dynamics of the motors should be

taken into account) 1st order TF. In the case of keeping dynamics into account, eq. (2.44), (2.45), (2.51), (2.37) can be rewritten as:

$$\begin{aligned}\frac{T(s)}{k_{PWM}(s)} &= \frac{\frac{1}{k_\omega}}{\tau_m * s + 1} * b_T * u_{\max} = \frac{\frac{b_T * u_{\max}}{k_\omega}}{\tau_m * s + 1} = \frac{k_T}{\tau_m * s + 1} \\ \frac{H(s)}{k_{PWM}(s)} &= \frac{\frac{1}{k_\omega}}{\tau_m * s + 1} * b_H * u_{\max} = \frac{\frac{b_H * u_{\max}}{k_\omega}}{\tau_m * s + 1} = \frac{k_H}{\tau_m * s + 1}\end{aligned}, \quad (3.11)$$

where $k_T = \frac{b_T * u_{\max}}{k_\omega}$ and $k_H = \frac{b_H * u_{\max}}{k_\omega}$.

In the case, if dynamics of the motors is small comparing to dynamics of whole system eq. (3.11) can be simplified to:

$$\begin{aligned}\frac{T(s)}{k_{PWM}(s)} &= k_T \\ \frac{H(s)}{k_{PWM}(s)} &= k_H\end{aligned}, \quad (3.12).$$

Combination of eq. (3.10) and (3.12) gives:

$$\begin{aligned}\frac{\phi(s)}{k_{2PWM}(s) - k_{4PWM}(s)} &= \frac{k_\varphi}{s^2} \\ \frac{\theta(s)}{k_{3PWM}(s) - k_{1PWM}(s)} &= \frac{k_\theta}{s^2} \\ \frac{\psi(s)}{k_{1PWM}(s) + k_{3PWM}(s) - \left(k_{2PWM}(s) + k_{4PWM}(s)\right)} &= \frac{k_\psi}{s^2}\end{aligned}, \quad (3.13)$$

where $k_\varphi = \frac{l * k_T}{J_{x_b x_b}}$, $k_\theta = \frac{l * k_T}{J_{y_b y_b}}$ and $k_\psi = \frac{k_H}{J_{z_b z_b}}$.

Eq. (2.41) that describes quadrocopter's motion in *XYZ* can be simplified by neglected drag forces [5] and can be modified to:

$$r_X(s) = \frac{1}{s^2}\left(\frac{c\psi(s) * sn\theta(s) + c\theta(s) * sn\phi(s) * sn\psi(s)}{m} * T(s)\right)$$
$$r_Y(s) = \frac{1}{s^2}\left(\frac{s\theta(s) * sn\psi(s) - sn\phi(s) * c\theta(s) * c\psi(s)}{m} * T(s)\right) \quad , (3.14).$$
$$r_Z(s) = \frac{1}{s^2}\left(\frac{c\phi(s) * c\theta(s)}{m} * T(s) - g\right)$$

For design purpose this TF should be linearized to:

$$r_X(s) = \frac{1}{s^2}\left(\frac{\theta(s) * c\psi + \phi(s) * sn\psi}{m} * T(s)\right)$$
$$r_Y(s) = \frac{1}{s^2}\left(\frac{\theta(s) * sn\psi - \phi * c\psi}{m} * T(s)\right) \quad , (3.15)$$
$$r_Z(s) = \frac{1}{s^2}\left(\frac{1}{m} * T(s) - g\right)$$

where $T(s) = k_T * \left(k_{PWM1}(s) + k_{PWM2}(s) + k_{PWM3}(s) + k_{PWM4}(s)\right)$.

Thereby, changes in the quadrocopter attitude and altitude can be described by 2nd order functions from eq. (3.13) and eq. (3.15). So an approach for choosing desired poles for 2nd order system should be described and controllers for 2nd order system by Ackerman equation should be designed.

3.3 Desired poles for 2nd order system

Stability and quality of a system are determined by its characteristic equation. Choice of desired poles depends on required quality of the system. The simplest way of choosing poles for 2nd order system is to use standard approach described e.g. in [11]. According to this approach a TF of 2nd order system should be represented as:

$$G(s) = \frac{\omega_n^2}{s^2 + 2 * \zeta * \omega_n + \omega_n^2} \text{ , (3.16)}$$

where ω_n is natural frequency and ζ is damping ratio.

This TF is analyzed in time domain by step input. Based on the step response the quality of the system can be estimated. Typical step response with all specifications is shown on fig. 3.2.

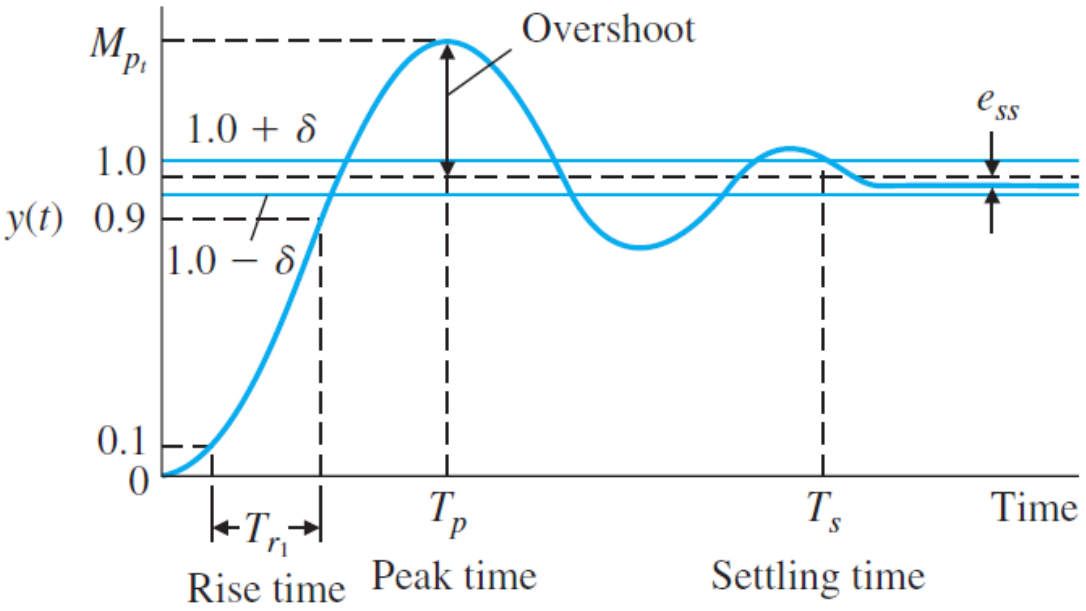

fig. 3.2 Step response of a 2nd order system [10]

The following quality parameters are shown: rise time, peak time, overshoot, settling time, steady state error. Usually for design purpose minimum settling time and overshooting should be specified.

Percent of overshoot $percOvSh$ can be set by varying damping ratio ζ (eq. 3.17) and settling time T_s by varying natural frequency ω_n (eq. 3.18).

$$percOvSh = e^{-\zeta*\pi/\sqrt{1-\zeta^2}} * 100, \text{ (3.17)}$$

$$T_s = \frac{4}{\zeta * \omega_n}, \text{ (3.18).}$$

For finding damping ratio ζ and natural frequency ω_n eq. (3.17) and (3.18) is rewritten as:

$$\zeta = \frac{\left(\frac{1}{\pi} * \ln \frac{percOvSh}{100}\right)}{\sqrt{1 + \left(\frac{1}{\pi} * \ln \frac{percOvSh}{100}\right)^2}}, \quad (3.19)$$

$$\omega_n = \frac{4}{\zeta * T_s}, \quad (3.20).$$

Based on ζ and ω_n the desired poles can be calculated as:

$$s_{1,2} = -\zeta * \omega_n \pm j * \omega_n * \sqrt{1 - \zeta^2}, \quad (3.21).$$

Thereby, by choosing quality parameters: overshooting and settling time T_s, the damping ratio ζ and natural frequency ω_n can be calculated (eq. 3.19 and 3.20). Based on the last ones the desired poles can be found by eq. (3.21).

For control of the quadrocopter, as for most of other systems, the fastest response with minimal overshooting should be provided. The damping ratio ζ for this behavior is well known and equals to 0.707 that corresponds to 4.32% of overshooting. Settling time in the model can be close to zero by choosing ω_n close to infinity, but in real system it is not possible. Based on current experimental result with the quadrocopter T_s should be no more than $0.8s$, so $T_s = 0.8s$ is chosen. Based on chosen quality parameters $T_s = 0.8s$ and $percOvSh = 4.32\%$ by using eq. (3.19, 3.20, 3.21) the desired poles are calculated as:

$$s_{1,2} = \text{-3.5357} \pm \text{j*3.5354} \quad (3.22).$$

The algorithm of calculation is realized as function ‘Dpoles.m’ in Matlab, where inputs are *percOvSh* and settling time T_s and outputs are desired poles (see App. B).

3.4 Attitude control

3.4.1 Design of controllers

Mathematical models from eq. (3.13) is 2nd order and in general can be represented as:

$$G_p(s) = \frac{b_1 * s + b_0}{s^2 + a_1 * s + a_0}, \text{ (3.23).}$$

The state space equations in canonical form for a 2nd order system are:

$$\begin{aligned} \dot{x}(t) &= A * x(t) + B * u(t) \\ y(t) &= C * x(t) \end{aligned}, \text{ (3.24)}$$

where $A = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} b_0 & b_1 \end{bmatrix}$,

that can be represented as shown on fig. 3.3.

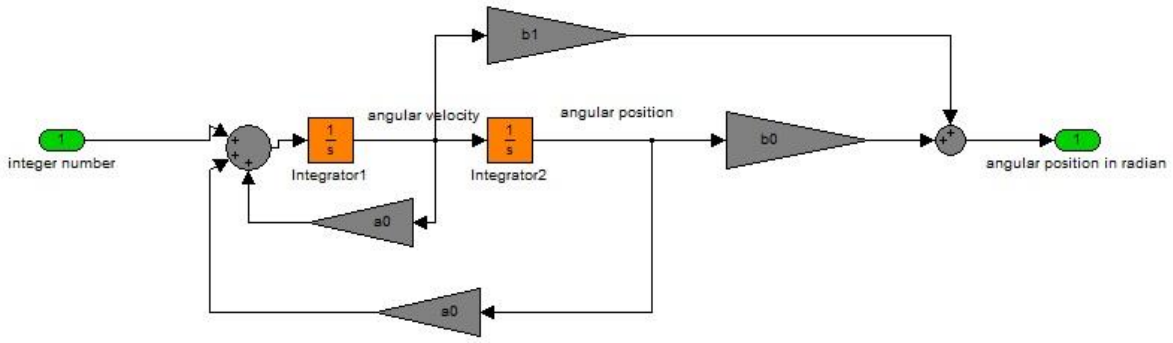

fig. 3.3 Canonical form of 2nd order system

By comparing TF of quadrocopter (eq. 3.1) and 2nd order TF in general form (eq. 3.23) :

$$G_p(s) = \frac{b_1 * s + b_0}{s^2 + a_1 * s + a_0} \lhd\!\!=\!\!\rhd G_p(s) = \frac{k}{s^2}, \text{ (3.25)}$$

where k in general represents k_φ, k_θ or k_ψ (calculation of k_φ, k_θ and k_ψ is in eq. 3.13).

It can be concluded that coefficients b_0 equals to k_θ and b_1, a_0, a_1 equal to 0. So general state space form can be rewritten as:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ k_\theta \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \end{bmatrix},\ (3.26).$$

The structure diagram is shown on fig. 3.4, where k is mentioned in eq. 3.25.

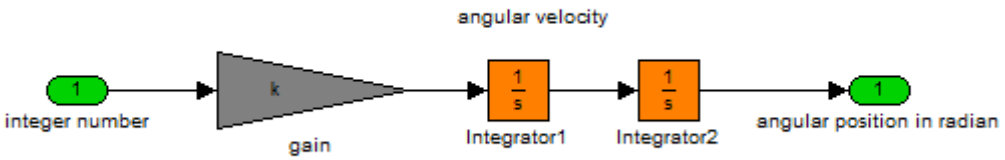

fig. 3.4 Structure diagram of the pitch angle model

For chosen system (eq. (3.25)), desired characteristic equation, from eq. (3.6), can be written as:

$$\alpha_c(s) = s^2 + (\lambda_1 + \lambda_2) * s + \lambda_1 * \lambda_2 = 0,\ (3.27),$$

and with taking into account that $\lambda_{1,2} = 3.5357 \pm j*3.5354$ (eq. (3.22)), it can be stated that:

$$\alpha_c(s) = s^2 + 7.0714 * s + 25 = 0,\ (3.28).$$

By combining eq. (3.26) with eq. (3.28) and eq. (3.7) it can be declared that:

$$\alpha_c(A) = A^2 + 7.0714 * A + 25 * I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + 7.0714 * \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} +$$

$$+25 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 7.0714 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix} =$$

$$= \begin{bmatrix} 25 & 7.0714 \\ 0 & 25 \end{bmatrix},\ (3.29).$$

Calculate the 2nd element from eq. (3.5). For current system it corresponds to the following equation:

$$\begin{bmatrix} B & A*B \end{bmatrix}^{-1}, \text{ (3.30)}$$

where $A*B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ k \end{bmatrix} = \begin{bmatrix} k \\ 0 \end{bmatrix}$, so

$$\begin{bmatrix} B & A*B \end{bmatrix}^{-1} = \begin{bmatrix} 0 & k \\ k & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & k \\ k & 0 \end{bmatrix}, \text{ (3.31).}$$

Thus, from eq. (3.29) and (3.31), eq. (3.5) can be rewritten as:

$$K = \begin{bmatrix} 0 & 1 \end{bmatrix} * \begin{bmatrix} B & A*B \end{bmatrix}^{-1} * \alpha_c(A) = \begin{bmatrix} 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & k \\ k & 0 \end{bmatrix} * \begin{bmatrix} 25 & 7.0714 \\ 0 & 25 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 25*k \\ 25*k & 7.0714*\text{k} \end{bmatrix} = \begin{bmatrix} 25*k & 7.0714*k \end{bmatrix}, \text{ or result in short form is:}$$

$$K = \begin{bmatrix} K_1 & K_2 \end{bmatrix} = \begin{bmatrix} 25*k & 7.0714*k \end{bmatrix}, \text{ (3.32).}$$

Thereby, closed-loop system matrix A_f (eq. (3.4)) can be represented as:

$$A_f = \begin{bmatrix} 0 & 1 \\ -a_0 - K_1 & -a_1 - K_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -25*k & -7.0714*\text{k} \end{bmatrix}, \text{ (3.33)}$$

The state-space equations for closed-loop system are:

$$\begin{aligned} \dot{x}(t) &= A_f * x(t) + B * u(t) \\ y(t) &= C * x(t) \end{aligned}, \text{ (3.34)}$$

where $A_f = \begin{bmatrix} 0 & 1 \\ -25*k & -7.0714*\text{k} \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

Structure diagram that corresponds to closed-loop system with matrix A_f is shown on fig. 3.5.

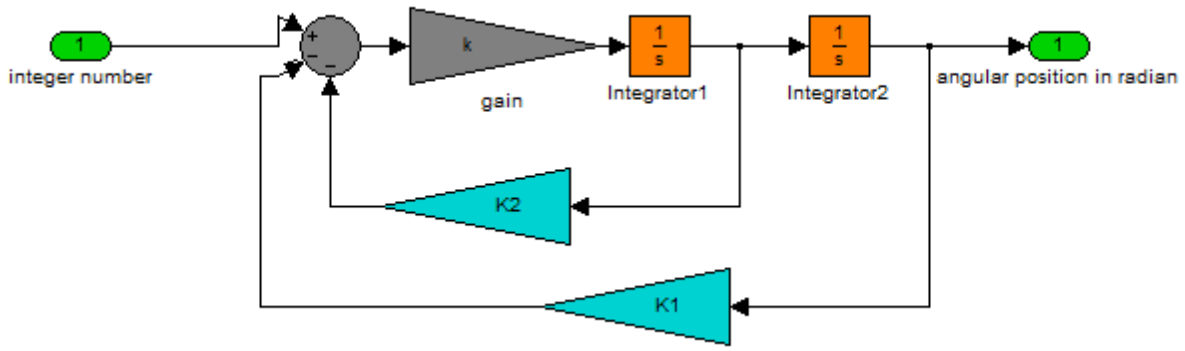

fig. 3.5 Feedback system with desired poles

For implementation of this approach a function ‘AckCont’ in Matlab was written (see App. B).

By regulators the system keeps its position around zero. However, the attitude controllers should follow desired values of pitch, roll and yaw. An output of a closed-loop TF corresponds to an input, if a gain of the closed-loop TF equals to one. So TF of the system from fig. 3.5 should be found. Represent the system as shown on fig. 3.6.

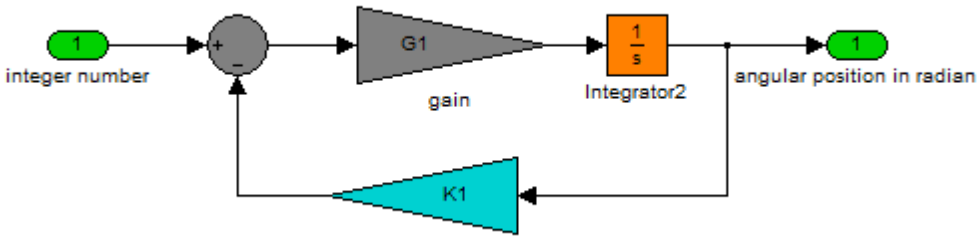

fig. 3.6 Feedback system with desired poles: finding TF

In this case inner TF $G_1(s)$ can be described as:

$$G_1(s) = \frac{\frac{k}{s}}{1 + \frac{k}{s} * k_2} = \frac{k}{s + k * k_2}, \text{ (3.35).}$$

The TF of the regulator $G(s)$ with taking into account eq. (3.35) can be written as:

$$G(s)=\frac{\frac{k}{s+k*k_2}*\frac{1}{s}}{1+\frac{k}{s+k*k_2}*\frac{1}{s}*k_1}=\frac{k}{s^2+k*k_2*s+k*k_1}, \text{ (3.36).}$$

Based on eq. (3.36) the gain of $G(s)$ can be found as:

$$G(0)=\frac{1}{k_1}, \text{ (3.37).}$$

Result of eq. (3.37) means that numerator of $G(s)$ should be multiplied by k_1. The final TF of the controller is:

$$G(s)=\frac{k*k_1}{s^2+k*k_2*s+k*k_1}, \text{ (3.38).}$$

3.4.2 Simulation results

Control of orientation is implemented in Simulink based on eq. (3.13) and eq. (2.32) (fig. 3.7).

Blocks ‘roll_cont’, ‘pitch_cont’ and ‘yaw_cont’ are represented control laws that are designed based on eq. (3.32). With taking into account eq. (3.38) controllers for roll, pitch and yaw are created in Simulink for incorporation in whole system, shown on fig. 3.7. Example for roll control is shown on fig. (3.8).

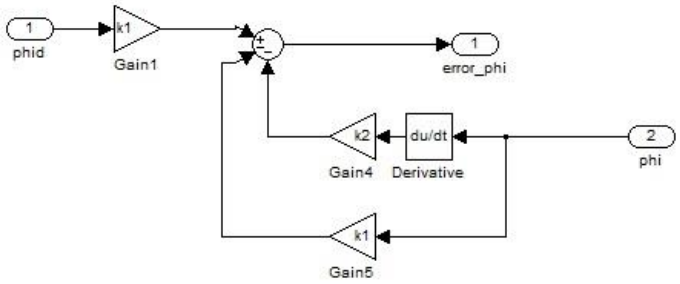

fig. 3.8 Implementation of attitude controllers as a block for incorporation in whole model: Simulink

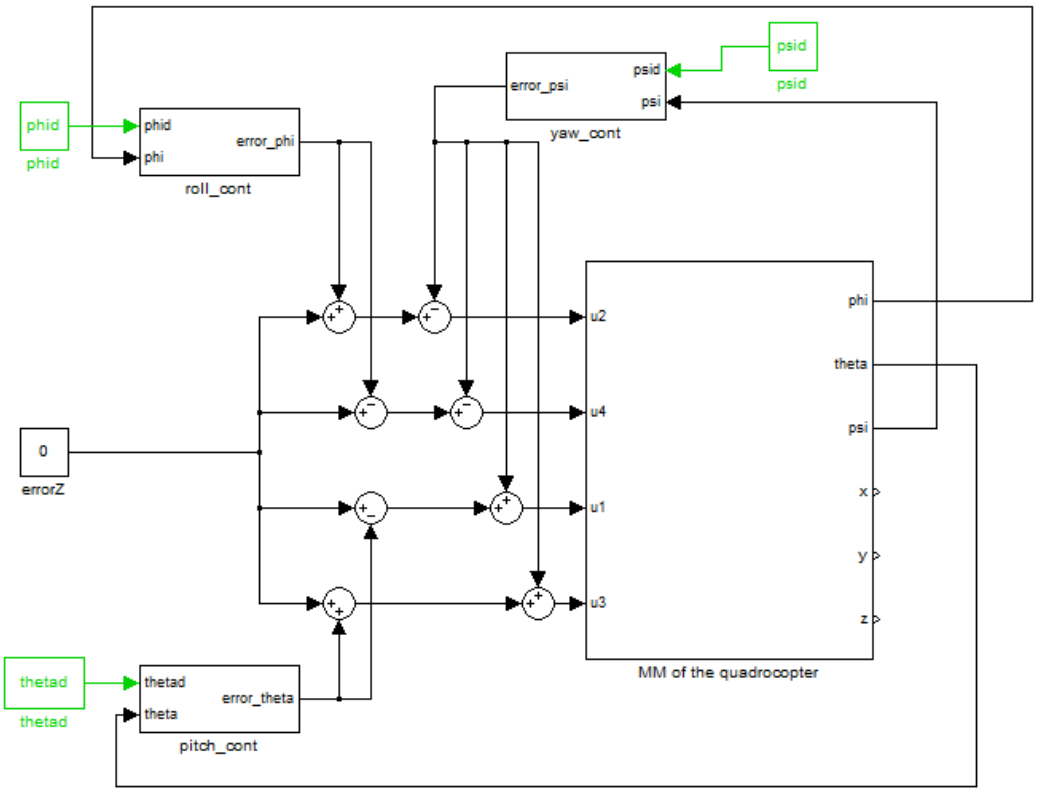

fig. 3.7 Orientation control scheme: Simulink

The system was analyzed with different step signals. Theoutput for unit step signal is shown on fig. 3.9 for pitch angle and on fig. 3.10 for yaw angle (step response for roll angle is omitted, since it is identical to the results on fig. 3.9).

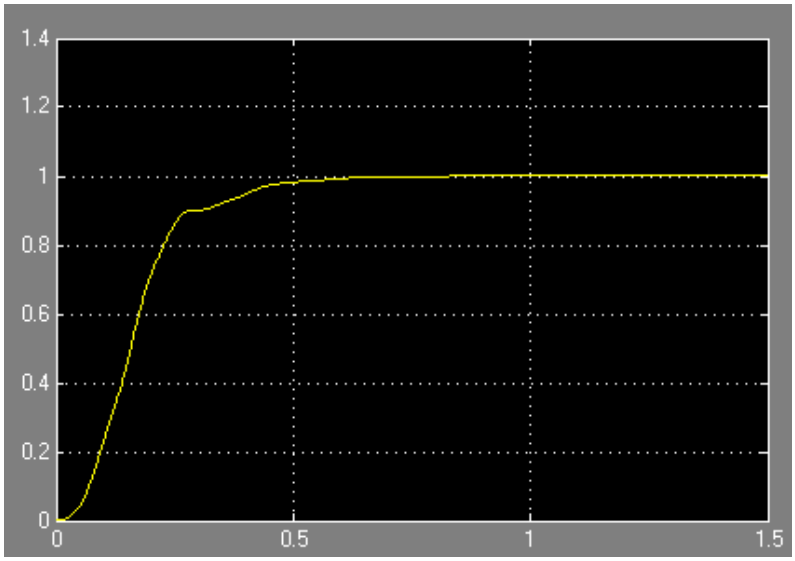

fig. 3.9 Step response: pitch angle

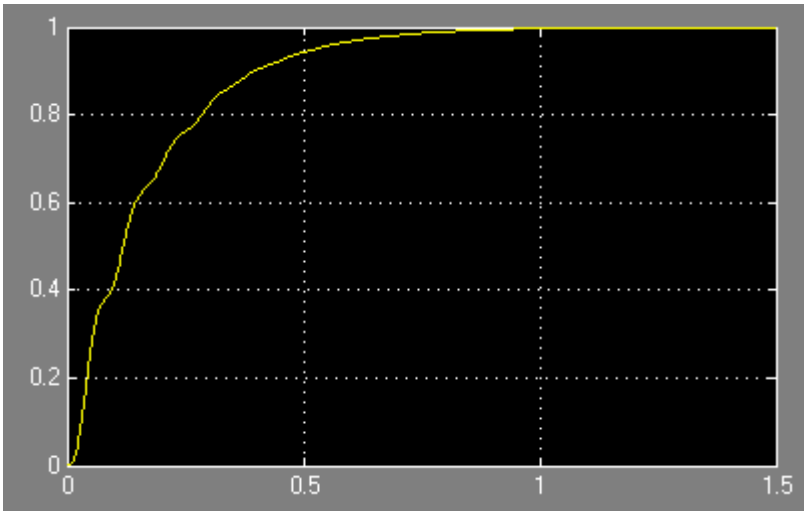

fig. 3.10 Step response: yaw angle

It has clearly seen that controllers provide quality requirements as $T_s = 0.8s$ and $percOvSh = 4.32\%$. Additionally, it should be mentioned that steady state error for this model is zero, because TFs of the MM (eq. 3.13) has 2nd order integration

As soon as designed controllers show required result, they should be implemented in a microcontroller and checked through experiments with real quadrocopter.

3.5 Altitude control

3.5.1 Design of controllers

Position control is divided into two parts: height control along Z axis and 2D control in XY plane.

Height control is based on 3rd equation from linearized model (eq. 3.15) and can be calculated based on Ackerman approach (eq. 3.32), where $k = \frac{k_T}{m}$.

For control law in XY plane the algorithm from [23] is chosen. Based on eq. (3.15) required acceleration can be calculated as:

$$\begin{aligned}\phi^d(s) &= g*\left(s^2 r_X(s)*sn\psi - s^2 r_Y(s)*c\psi\right)\\ \theta^d(s) &= g*\left(s^2 r_X(s)*c\psi + s^2 r_Y(s)*sn\psi\right)\end{aligned}, \quad (3.39)$$

where g is gravitational acceleration, ϕ^d and $\theta^d(s)$ are desired values. In case if yaw angle equals to zero and constant during whole flight it can be even simplified to:

$$\begin{aligned}\phi^d(s) &= -g*s^2 r_Y(s)*c\psi\\ \theta^d(s) &= g*s^2 r_X(s)*c\psi\end{aligned}, \quad (3.40)$$

3.5.2 Simulation results

Control law based on eq. (3.39) is realized in Matlab/Simulink as shown on fig. (3.11).

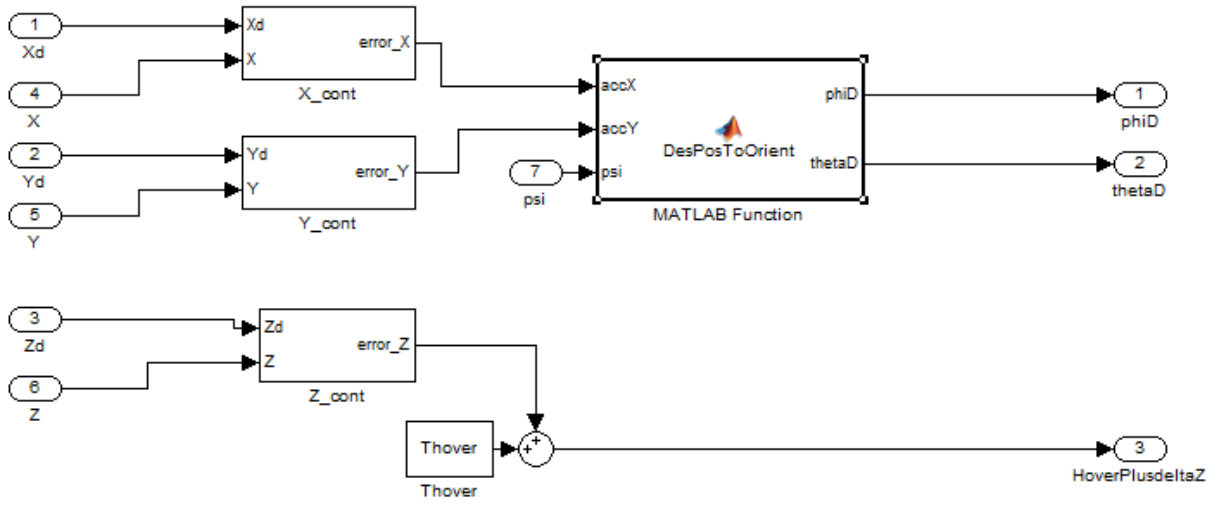

fig. 3.11 Altitude Control: Matlab/Simulink

Controllers ‘X_cont’, ‘Y_cont’, ‘Z_cont’ have the same structures as attitude controls (fig. 3.8). Coefficients for ‘Z_cont’ are calculated based on Ackerman approach; coefficients for ‘X_cont’, ‘Y_cont’ controllers are adjusted manually.

Hereby, the structure of the whole system is implemented as shown on fig. (3.12).

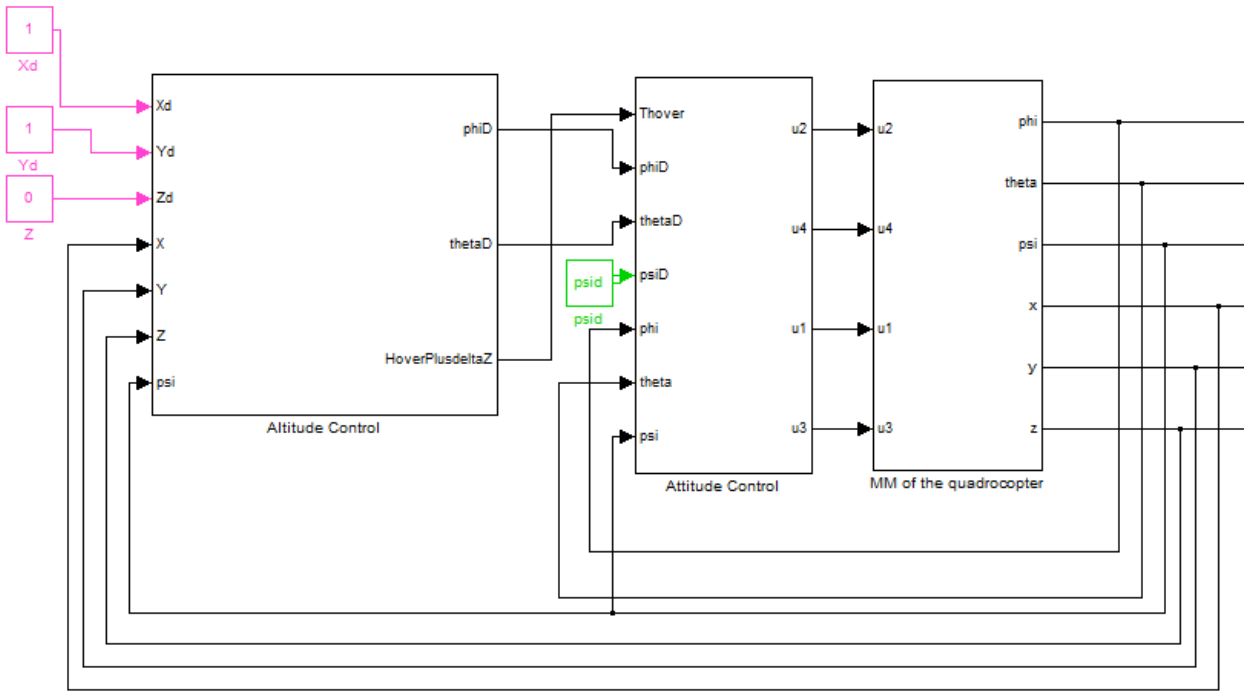

fig. 3.12 Altitude Control: Matlab/Simulink

Simulation results for $X^d = 1, Y^d = 1, Z^d = 1, \psi^d = \frac{\pi}{6}$ are shown on fig. (3.13).

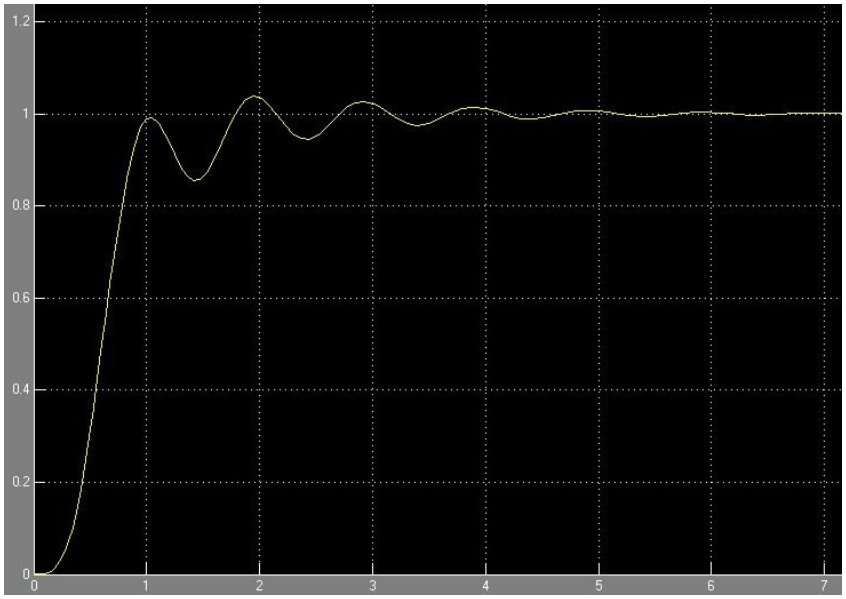

fig. 3.13 a X position

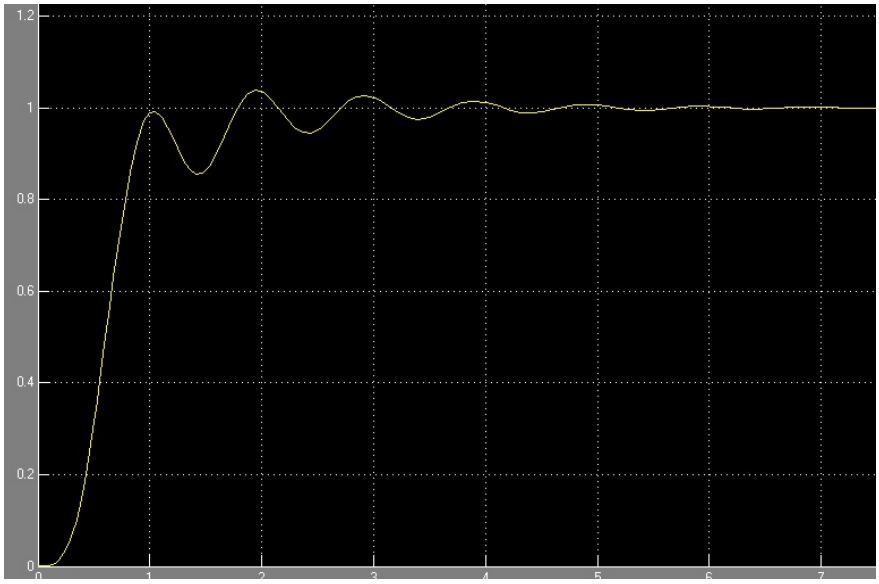

fig. 3.13 b Y position

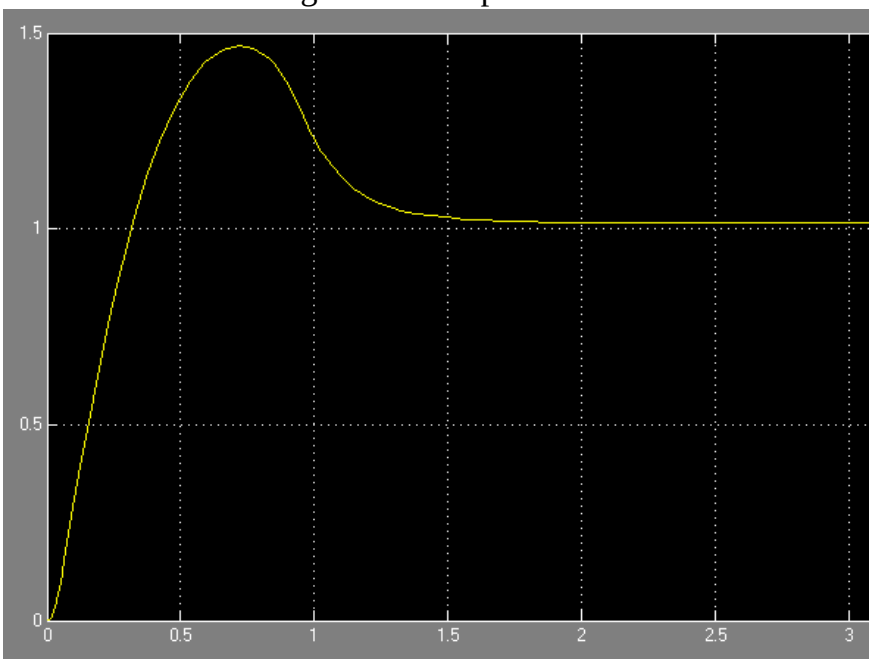

fig. 3.13 c Z position

There are no overshoots along X and Y axes and settling time about 3s. The overshoot along Z axis is about 50% and settling time about 2s. These ones can be accepted for some application, but in general it should be improved. For example if a quadrocopter should record visual information during its movement from one desired point to another one, the behavior shown on fig. 3.13 is unacceptable and should be improved. On the other hand, if a quadrocopter should make several photos in a stable state, this behavior is acceptable since the positioning by itself is precise enough. Additionally, a steady state error along Z axis is about 0.14 m. It can be eliminated by using integral component in controller for Z axis.

Chapter 4 Implementation of the Control System

The MM of the quadrocopter, attitude and position control systems were developed in previous chapters. To adapt the MM to the real quadrocopter and to check controllers, several experiments should be fulfilled. First of all the MM should be validated, afterwards control system should be implemented and tested. For the MM validation and adjusting attitude control two test benches are used. Test bench 1 has 2 DOF and used for validating and adjusting pitch orientation. Test bench 2 has 3DOF and used for validating and adjusting yaw orientation. Afterwards, position control system is checked in flying version.

The control algorithms were implemented as a script in a frame of software created inside Aerospace Information Technology Department, Würzburg University. This software also was used though experiments for data sending and recording.

4.1. Transfer functions for pitch and roll orientation

Transfer function for pitch and roll was obtained in previous chapter. However, structure of test bench 1 is different from free motion of the object; Because of this fact the TF for pitch and roll should be modified for test bench 1.

4.1.1 Elements of the system

For adjusting pitch controller test bench 1 is used (fig. 4.1). This one consists of a cross-frame, four motors with propellers fixed on their shafts, four power bridges for motor control, a gyro sensor and a microcontroller. It has 2DOF: pitch (an axis of rotation shown as black line) and yaw orientation.

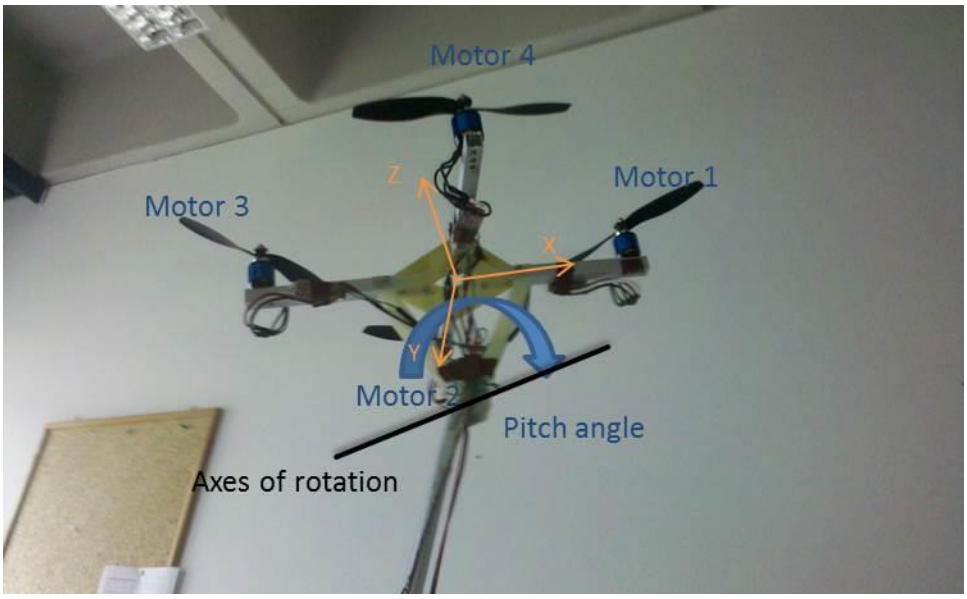

fig. 4.1 Test bench 1: 2DOF

The gyro sensor is set close to the center of the symmetry of the quadrocopter. The orientation of the x and y axes of gyro sensor are parallel to correspondent axes of xy (fig. 4.1). The pitch angle is limited by construction in the range of $-18 \le \theta \le 18$ degree.

Each actuator consists of power bridge BL-Ctrl1.2 [14], BLDC motor KA20-22L [15], a blade EPP0845 [16]. The power bridge is controlled by the microcontroller AT32UC3A0512-0ESAL fixed on evaluation board EVK1100 [17]. Relation between pitch angle and force and torque from a blade is shown on fig. 4.2. An integer number in a range $0\ldots255$ should be send by the microcontroller though I2C to the power bridge. The power bridge produces control signals to rotate motor shaft with angular velocity which is proportional to integer value , where 0 corresponds to stop and 255 to rotation with maximal velocity. The blade which is fixed on the motor's shaft begins to rotate and produce thrust force and hub torque.

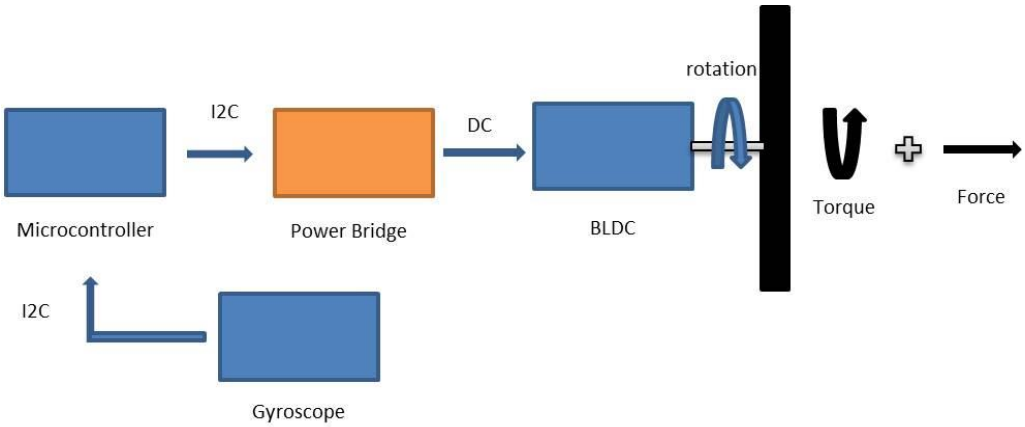

fig. 4.2 Structure diagram of actuator and feedback: test bench 1

The forces from propellers move quadrocopter around axes of rotation and change pitch angle. The angular velocity around pitch angle is measured by gyro IMU3000 [18] in deg/sec.Integration of this value gives changing of pitch angle in deg. Error between desired and current values of pitch angle should be changed according to the design law. This output is transferred through the power bridges, motors and blades to produce forces and torques. This motion corresponds to 2nd equation from eq. (3.13), but since the axis of rotation is not in the center of symmetry of the quadrocopter additional force occurs. Thereby, 2nd equation from eq. (3.13) should be modified for applying to test bench 1.

4.1.2 Linear model for pitch angle on test bench 1

Consider forces acting on the quadrocopter: thrust force from motor 1, thrust force from motor 3 and gravitational force (fig. 4.3).

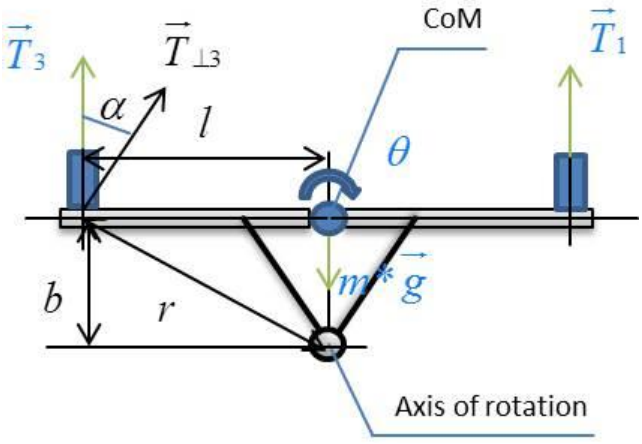

fig. 4.3 Forces applied to the quadrocopter: test bench 1

2nd equation from eq. (3.10) can be rewritten as:

$$\dot{\omega}_{y_b} = \frac{\left(\vec{T}_{\perp 3} - \vec{T}_{\perp 1}\right) * r + m * \vec{g} * b * \sin\theta}{J_{y_b y_b}}, \quad (4.1)$$

where $\vec{T}_{\perp 3}$ and $\vec{T}_{\perp 1}$ are projections of thrust forces $\vec{T}_3$ and $\vec{T}_1$ on perpendiculars to radius vector $\vec{r}$; $\vec{r}$ is a shortest distance between a point, where a force $\vec{T}_3$ or $\vec{T}_1$ applied, and axis of rotation; $b * \sin\theta$ is the displacement vector for gravitational force . These projections can be calculated as:

$$\begin{aligned} \vec{T}_{\perp 1} &= \vec{T}_1 * \cos\alpha \\ \vec{T}_{\perp 3} &= \vec{T}_3 * \cos\alpha \end{aligned}, \quad (4.2)$$

where α is a constant angle between vector of force $\vec{T}_3/\vec{T}_1$ and $\vec{T}_{\perp 3}/\vec{T}_{\perp 1}$. This angle and radius vector $\vec{r}$ can be calculated as:

$$\begin{aligned} \alpha &= \arctan\frac{b}{l} \\ r &= \sqrt{b^2 + l^2} \end{aligned}, \quad (4.3)$$

where b is a shortest distance from the center of the quadrocopter's symmetry to axis of rotation, l is a shortest distance from force $\vec{T}_3/\vec{T}_1$ to the center of the quadrocopter's symmetry.

Forces $\vec{T}_3$ and $\vec{T}_1$ can be calculated by 1st equation from eq. (3.12) in rewritten form:

$$\begin{aligned} \vec{T}_1 &= u_1 * k_T \\ \vec{T}_3 &= u_3 * k_T \end{aligned}, \quad (4.4)$$

where u_1 and u_3 are values in the range $0\ldots255$.

After combining eq. (4.1), (4.2), (4.3), (4.4), the relation between u_1, u_3 and $\dot{\omega}_{y_b}$ can be written as:

$$\dot{\omega}_{y_b} = \frac{(u_3 - u_1) * k_T * \sqrt{b^2 + l^2} * \cos\left(\arctan\frac{b}{l}\right) + m * g * b * \sin\theta}{J_{y_b y_b}}, \quad (4.5)$$

or in a form of TF as:

$$s^2\theta(s) = k_\theta * (u_3(s) - u_1(s)) + k_{mg} * \sin\theta, \quad (4.6)$$

where $k_\theta = \frac{k_T * \sqrt{b^2 + l^2} * \cos\left(\arctan\frac{b}{l}\right)}{J_{y_b y_b}}$, $k_{mg} = \frac{m * g * b}{J_{y_b y_b}}$, $u_3(s) - u_1(s)$ is input and $\theta(s)$ is output.

Hereby, the TF of pitch angle for test bench 1 can be stated as:

$$\theta(s) = \frac{k_\theta}{s^2} * (u_3(s) - u_1(s)) + \frac{1}{s^2} k_{mg} * \sin\theta(s), \quad (4.7)$$

This TF is nonlinear continuous one and for linearization, element $\sin\theta$ should be substituted by a linear element. With taking into account that fixed construction has pitch angle range approximately $-18 \le \theta \le 18$ degree, this element can be substituted by θ in radians [10]. So TF from eq. (3.13) can be rewritten as:

$$\frac{\theta(s)}{u_3(s) - u_1(s)} = G_{pitch}(s) = \frac{k_\theta}{s^2 - k_{mg}}, \quad (4.8).$$

The TF based on eq. (4.8) is created in Simulink and shown on fig. 4.4.

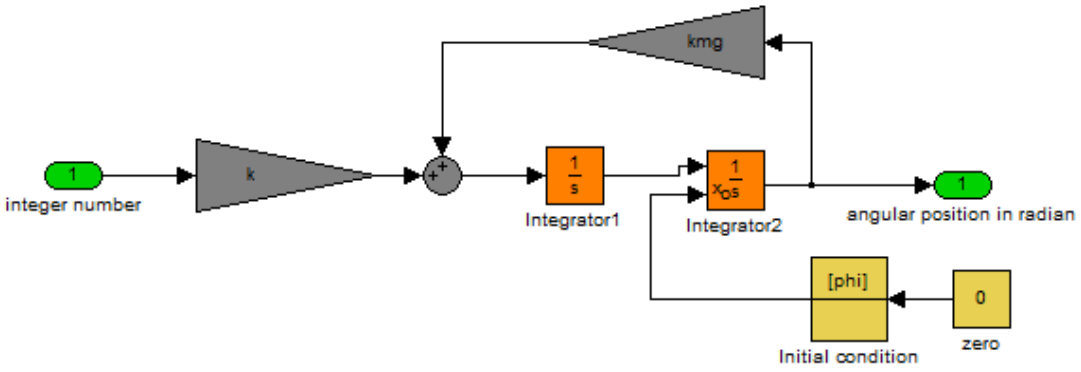

fig. 4.4 Open-loop TF of pitch orientation: test bench 1

4.1.3 Coefficients calculation and verification

Coefficients calculation

To calculate coefficients k, k_{mg} the length and weight of the quadrocopter should be measured. Initial data for calculation are:

$$b = 0.11m, \quad l = 0.205m, \quad m = 0.49kg, \quad g = 9.8\frac{m}{s^2}.$$

Moment of inertia component $J_{y_b y_b}$ according to eq. (2.19) and parallel axis theorem can be calculated as:

$$J_{y_b y_b} = \frac{2 * M * R^2}{5} + 2 * l^2 * m_M + m * b^2 = 0.0151, \text{ (4.9)}$$

where R is chosen equal to l.

Coefficients k_{mg} and k based on eq.(4.6) are:

$$k_\theta = \frac{k_T * \sqrt{b^2 + l^2} * \cos\left(\arctan\frac{b}{l}\right)}{J_{y_b y_b}} = 0.4487, \text{ (4.10)}$$

$$k_{mg} = \frac{m * g * b}{J_{y_b y_b}} = 34.9258$$

where $k_T = 0.0331$ coefficient for the motor taken from here [19].

So based on theoretical model $k_{mg} = 34.9258\frac{1}{s^2}$ and $k = 0.4487\frac{1}{s^2}$.

Coefficients verification

Feedback coefficients for designed controller fit the MM. The more precise the MM is, the more real behavior of the quadrocopter with designed controller coefficients relates to simulation results. It has been done several simplifications during creation MM and verification coefficients k and k_{mg} through experiment can improve MM. The verification procedure includes two types of experiments.

Verification of coefficient k_{mg}

First coefficient k_{mg} was verified. This one corresponds to movements in gravitational field without influence from blade's forces. The quadrocopter fixed in test bench 1 is inclined four times to random angular position and released to fall till the limited angle. Changes in angular position were recorded and as graph represented on fig. 4.5.

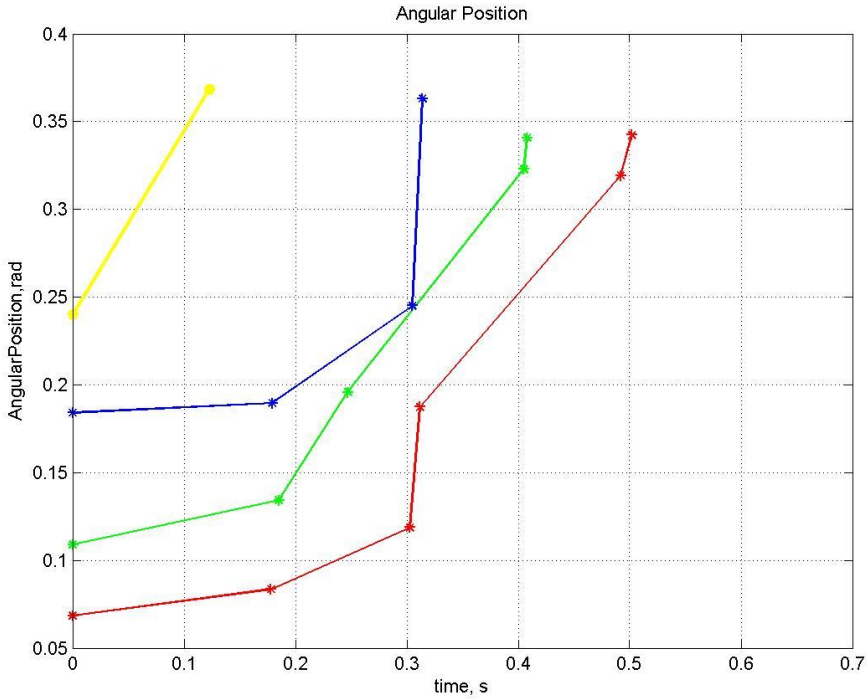

fig. 4.5 Changes in angular position during quadrocopter's free falling: test bench 1

Four initial angles are taken from these records and used as initial positions for the MM from fig. 4.4. The same trajectories based on MM with $k_{mg} = 34.9258\frac{1}{s^2}$ are shown on fig. 4.6 (colorful trajectories indicate experiment data and black ones simulation). Comparing the sets of trajectories shows that current value of k_{mg} does not describe the behavior of the quadrocopter adequate. It can be seen that model is too fast, so it means that the calculated inertia momentum is less than real.

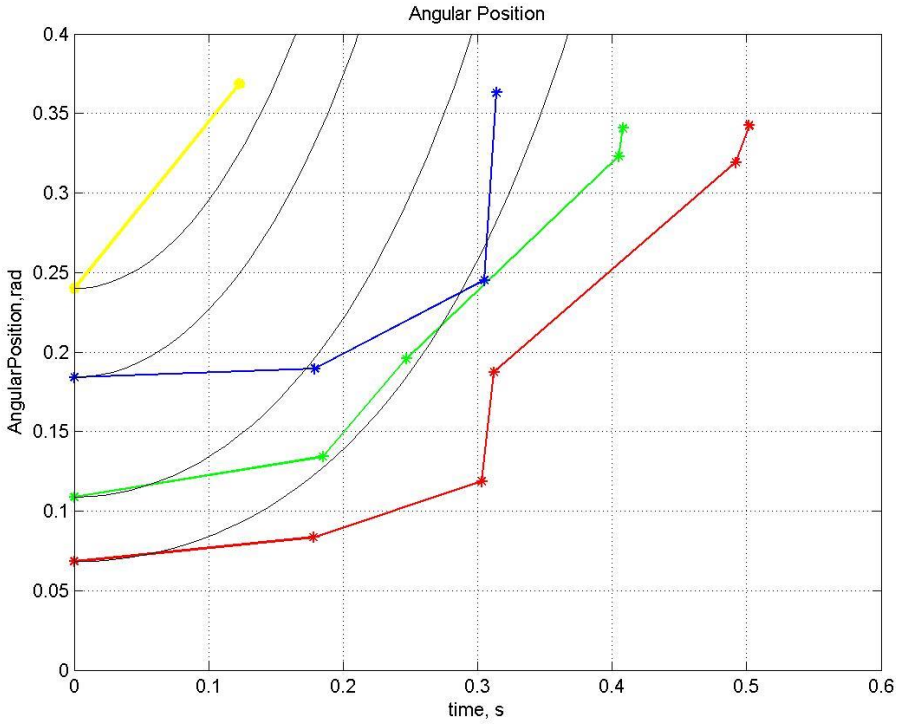

fig. 4.6 Experimental and theoretical trajectories: $k_{mg} = 34.9258 \frac{1}{s^2}$

To make the model adequate $k_{mg}$was decreased until the longest trajectories from both sets became as close as possible to each other's (fig. 4.7).

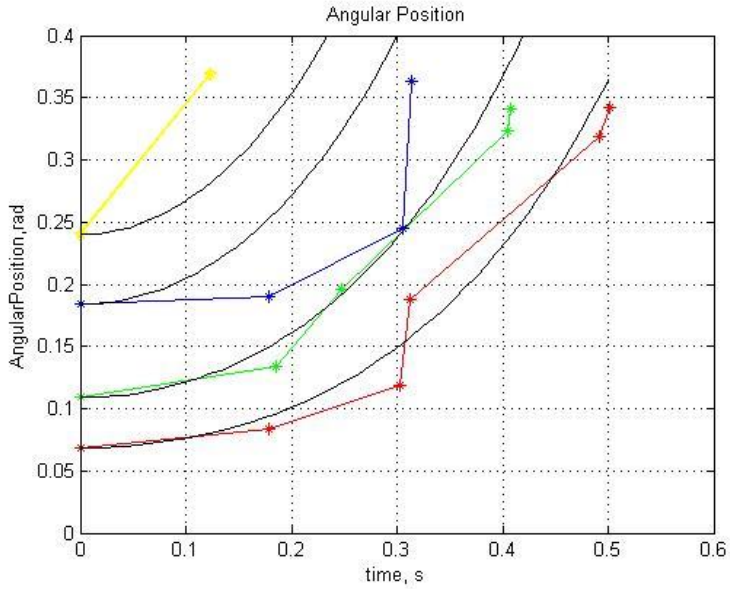

fig. 4.7 Experiment and theoretical trajectories: $k_{mg} = 22.3006 \frac{1}{s^2}$

The new k_{mg} is renewed as $k_{mg} = 22.3006\frac{1}{s^2}$. For making theoretical result close to real one, the new value $R = 1.72 * l$ is chosen.

Verification of coefficient k

The idea of verification is to find equilibrium conditions between torques from motor 1 and from gravitation force for various angles. Values of angular position and for motor control were recorded (see table 4.1).

Table 4.1

angle, rad	0.3491	0.3316	0.2897	0.2548	0.2217	0.1658	0.1117
value	65	60	50	40	30	20	10

Based on the eq. (4.7), with taking into account that angular acceleration $s^2\theta = 0$, the coefficient k is calculated. It is value in the range $0.1198 \ldots 0.2491$. The average value $k = 0.1844$ is chosen.

The script 'DOF2_ver.m' (see App. B) was written and used for processing and representation experiment data, for simulation TF from fig. 4.4.

After analyses and validation the MM of pitch angle (test bench1) was specified. The mathematical model is represented in a form of linear continuous time invariant transfer function (eq. 4.8) and can be used for controllers design.

4.2 Control design for pitch and roll orientation

By comparing TF of 2nd order function and TF of pitch orientation quadrocopter (eq. 4.8) coefficients for state space model are calculated as :

$$G(s) = \frac{b_1 * s + b_0}{s^2 + a_1 * s + a_0} \lhd\!\!=\!\!\rhd G_{pitch}(s) = \frac{k}{s^2 - k_{mg}}, \quad (4.11).$$

It can be concluded that b_0 equals to k, a_0 equals to $-k_{mg}$, b_1 and a_1 equal to zero.

Using the same logic as in section 3.4 and eq. (3.24 -3.29) feedback coefficients can be obtained as (details see in App. A):

$$K = \begin{bmatrix} K_1 & K_2 \end{bmatrix} = \begin{bmatrix} k_{mg} + 25*k & 7.0714*\mathrm{k} \end{bmatrix}, \text{ (4.12).}$$

Thereby, closed-loop system matrix A_f (eq. (3.4)) can be represented as:

$$A_f = \begin{bmatrix} 0 & 1 \\ -a_0 - K_1 & -a_1 - K_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ k_{mg} - k_{mg} - 25*k & -7.0714*\mathrm{k} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -25*k & -7.0714*\mathrm{k} \end{bmatrix}$$
, (4.13)

The state-space equations for closed-loop system are:

$$\begin{aligned} \dot{x}(t) &= A_f * x(t) + B * u(t) \\ y(t) &= C * x(t) \end{aligned}, \text{ (4.14)}$$

where $A_f = \begin{bmatrix} 0 & 1 \\ -25 & -7.0714 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

Structure diagram that corresponds to closed-loop system with matrix A_f is shown on fig. 4.8.

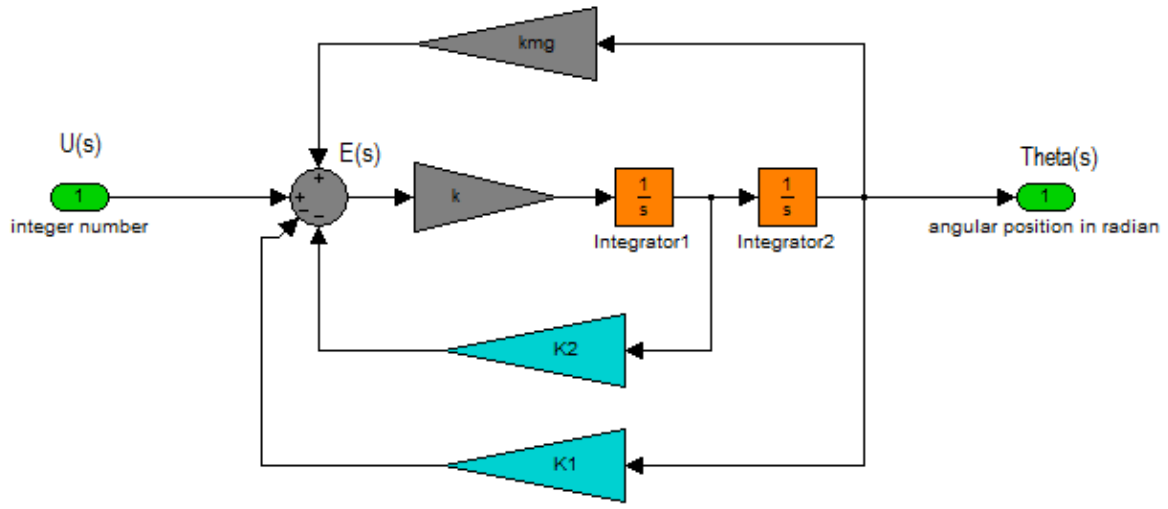

fig. 4.8 Feedback system with desired poles

Based on idea and equations mentioned above the script in Control System Toolbox (CST) was written (see App. B 'AckContSim.m'). The response of the closed-loop with desired poles and initial conditions $\begin{bmatrix}1 & 0\end{bmatrix}$ (means initial position is 1 radian and initial velocity is zero) is shown on fig. 4.9. It is clearly seen that for the model the overshooting is minimal (less than 4.32%), settling time is about 0.8 seconds, steady-state error equals to zero.

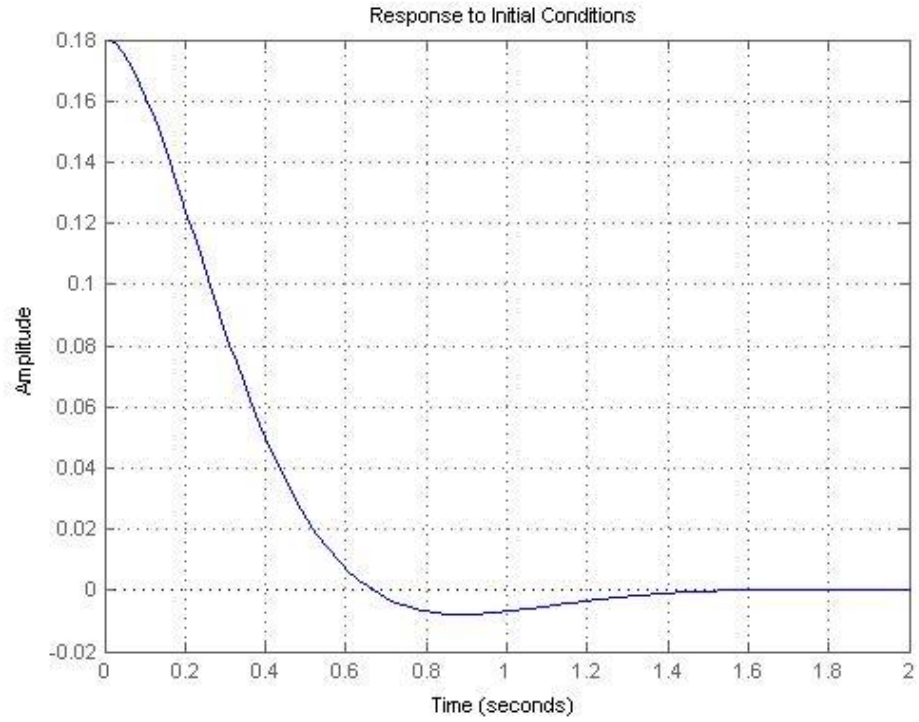

fig. 4.9 Response for initial conditions

4.2.1 Implementation of the regulator

Control law corresponds to closed-loop TF (fig. 4.8) and, with taking into account that the gyroscope generates data in degrees, this law can be represented as:

$$\varepsilon(t) = \theta^d(t) + \frac{1}{k} * \left(-K_1\right) * DtR * \theta(t) + \frac{1}{k} * (-K_2) * DtR * \dot{\theta}(t), \quad (4.15)$$

where $\theta^d(t) = u_3(t) - u_1(t)$ and $DtR = \frac{\pi}{180}$.

In form applied to motor's values it can be rewritten as:

$$u_1(t) = \varepsilon_\theta(t) = \frac{1}{k} * (-K_1) * DtR * \theta(t) + \frac{1}{k} * (-K_2) * DtR * \dot{\theta}(t)$$
$$u_3(t) = -\varepsilon_\theta(t) = -\left(\frac{1}{k} * (-K_1) * DtR * \theta(t) + \frac{1}{k} * (-K_2) * DtR * \dot{\theta}(t) \right), \quad (4.16).$$

Additionally, initial values of $u_3(t)$ and $u_1(t)$ should not be 0, because of two reasons. First reason is a simplification of the model for which this controller was designed. Time delays in changing velocities of quadrocopter's blades were neglected. The most important delay is during increasing the blade angular velocity from zero to some value. So the blades should always have some non-zero velocities. Moreover, there is some value of speed that provides hovering of the quadrocopter and all changes in control should be around this value. With taking into account these facts eq. (4.16) should be rewritten as:

$$u_1(t) = u_{\min} + u_{hov} + \varepsilon_\theta(t)$$
$$u_3(t) = u_{\min} + u_{hov} - \varepsilon_\theta(t), \quad (4.17),$$

where $\varepsilon_\theta(t)$ is defined by eq. (4.16), $u_{\min}$ is a minimal angular velocity of the blades, u_{hov} is an angular velocity for hovering.

The designed control law (eq. (4.17)) is incorporated in the software. Fulfilled experiments consist of two simple steps:

- change the quadrocopter pitch angle from 0 by some external force
- record system response

Results are shown on fig. 4.10, the data are recorded with discretization of 100 Hz.

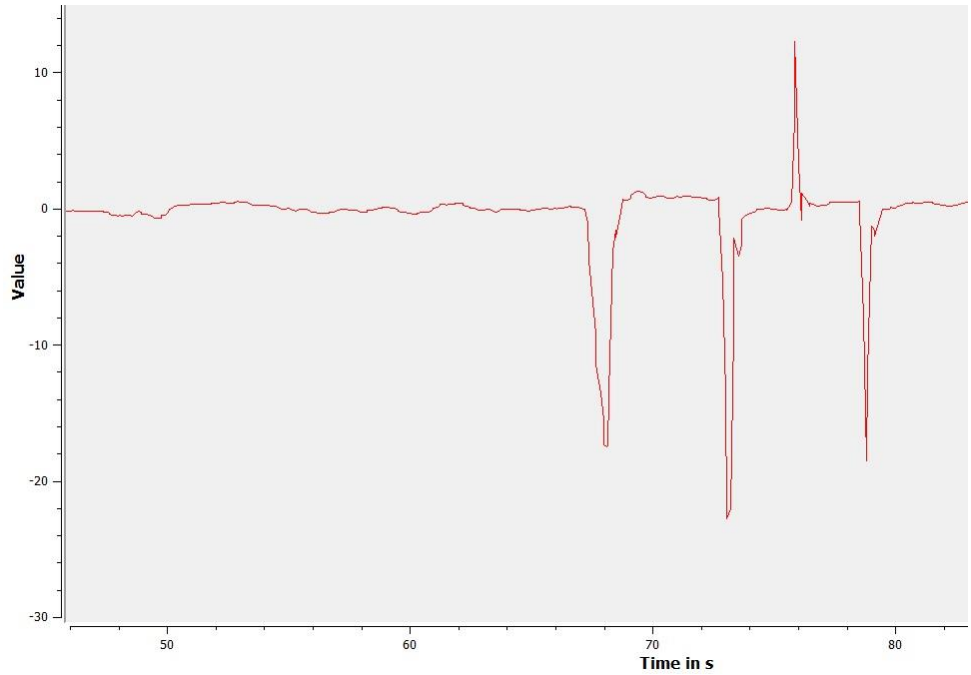

fig. 4.10 Responses of regulator on changing of initial conditions

The system should return its pitch to zero. It is inclined four times and returns to initial zero. For settling time estimation an area from fig. 4.10 is zoomed (fig. 4.11).

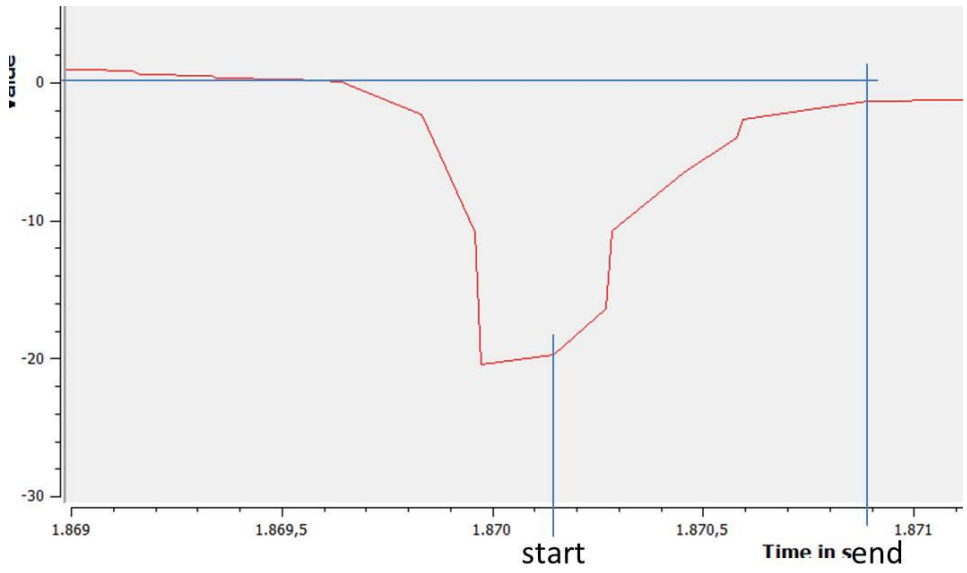

fig. 4.11 Response signal (zoomed part from fig. 4.10)

It can be seen that the settling time is about 0.6 second and a static error about 0.3 degree. Settling time in experiment is close to settling time from model. There is no static error in the model, since the model does not take into account all facts,

e.g. some friction in test bench joint that can be the source of this error. This error can be compensate by an integral component.

4.2.2 Implementation of the controller

To provide control for general input, the gain of the TF should be set to one. TF of the system from fig. 4.8 is:

$$G_{pitch\_f}(s) = \frac{k}{s^2 + k_2 * s + (k_1 - k * k_{mg})}, \text{ (4.18),}$$

and it's gain is:

$$G_{pitch\_f}(0) = \frac{k}{k_1 - k * k_{mg}}, \text{ (4.19).}$$

So a coefficient, which is inversed to $G_{pitch\_f}(0)$, should be set between the reference signal that represents desired pitch angle and input of feedback TF. As soon as a desiredt angle is in degree and the model in radian, transfer coefficients: $\frac{\pi}{180}$ between reference signal and input signal and $\frac{180}{\pi}$ between output signal and results should be added (fig. 4.12).

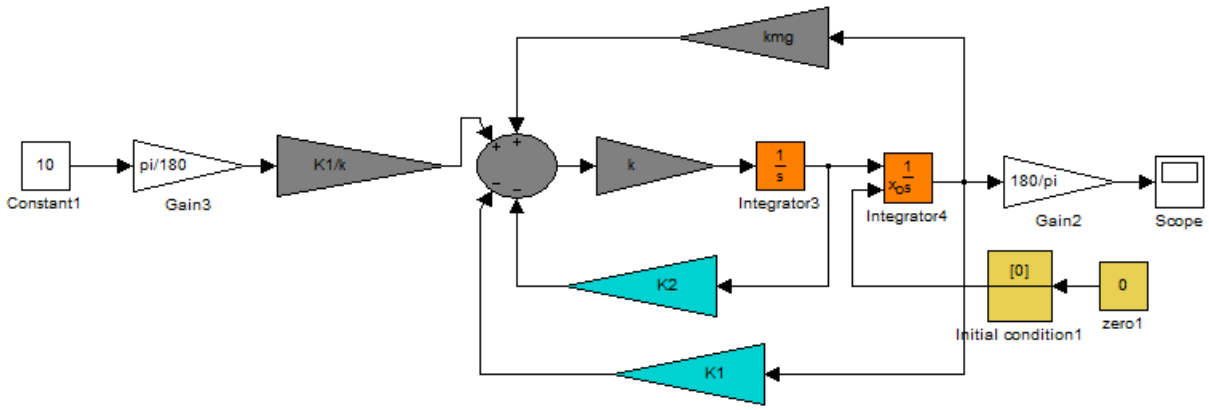

fig. 4.12 Pitch angle: control system for non-zero input

It means that for experiment the control law is:

$$\varepsilon_m(t) = \theta^d(t) * DtR * \left(\frac{K_1}{k}\right) + DtR * \left(\frac{1}{k} * (-K_1) * \theta(t) + \frac{1}{k} * (-K_2) * \dot{\theta}(t)\right), \text{ (4.20).}$$

After corresponding changes in microcontroller one more experiment was fulfilled. Response function is shown on fig. 4.13.

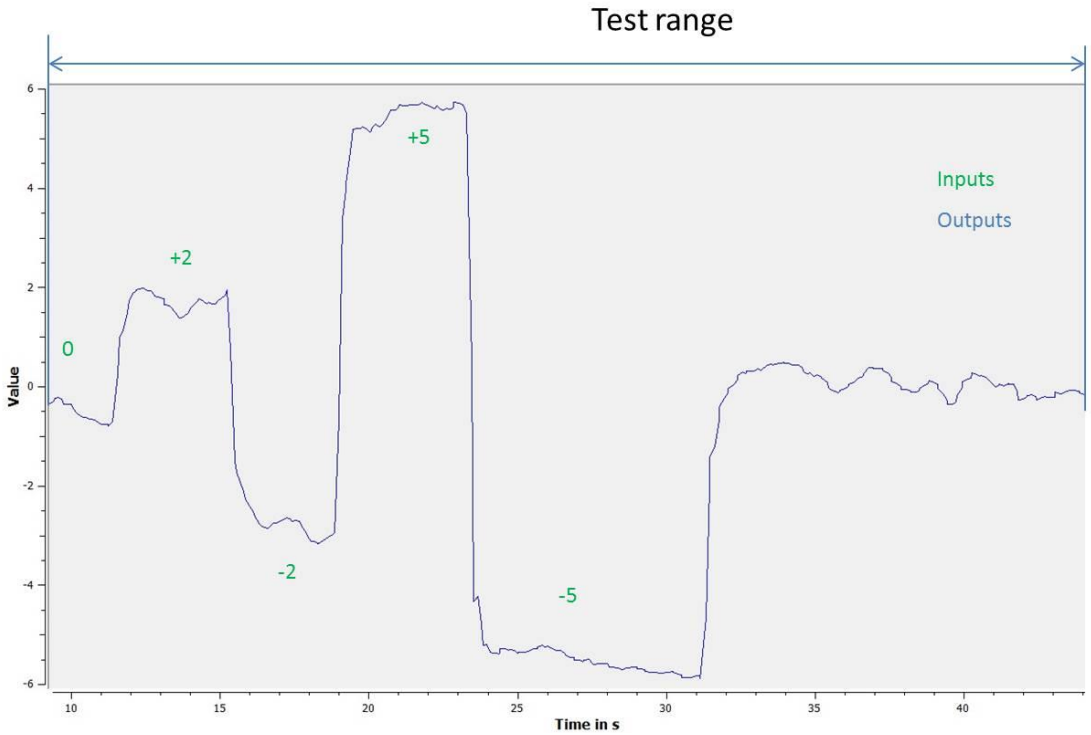

fig. 4.13 Pitch angle: control system for non-zero input

Quality parameters are summarized in table 4.2.

Table 4.2

input signals, *deg*	output signal, *deg*	settling time, *sec*	overshooting, *deg*
0	0.12	--	0.3
2	2.31	0.6	0.4
-2	-2.21	0.6	0.4
5	5.9	0.65	0.4
-5	-5.6	0.65	0.4

Quality parameters are inside required range, so pitch and roll attitude control is designed.

Results of experiments show that system works with required quality (overshooting and settling time) for both cases: zero input and non-zero input. Static error for both cases is inside 0.5 degree. With taking into account many approximations, this error is relatively small. To reduce steady state error integration should be used.

Roll controller by using the same logic can be stated as the following equations:

$$\begin{aligned} u_2(t) &= u_{\min} + u_{hov} - \varepsilon_{\varphi}(t) \\ u_4(t) &= u_{\min} + u_{hov} + \varepsilon_{\varphi}(t) \end{aligned}, \text{ (4.18),}$$

where $\varepsilon_{\varphi}(t) = \frac{1}{k} * (-K_1) * DtR * \varphi(t) + \frac{1}{k} * (-K_2) * DtR * \dot{\varphi}(t)$.
Results of roll controller are identical to results of pitch controller.

4.3 Controller for yaw orientation

Controller for yaw orientation is checked on test bench 2, which has 3DOF. Test bench 2 has a structure close to test bench 1, but fix point of test bench 2 is close to CoM of the quadrocopter.

The TF of yaw rotation based on 3rd line from eq. (3.13) can be stated as an equation:

$$\frac{\psi(s)}{u_1(s) + u_3(s) - \left(u_2(s) + u_4(s)\right)} = \frac{k_{\psi}}{s^2}, \text{ (4.21)}$$

where $k_{\psi} = \frac{k_H}{J_{z_b z_b}}$.

Component $J_{z_b z_b} = 0.0226$ is calculated by eq. (2.19). Coefficient k_{ψ} is measured by an experiment. Inputs $u_1(s)$ and $u_3(s)$ were set to value of hover and yaw rotation was recorded (fig. 4.14).

In the range $10\ldots12\sec$ released and stop manually. Angular velocity is about $1.084 rad/\sec$. Coefficient k_H was adjusted until angular rate of yaw in the model (file ‘’) became $1.084 rad/\sec$; new $k_H = 0.5 * 10^{-3}$ and $k_{\psi} = 0.0211$.

Controller for yaw orientation is calculated based on eq. (3.23) where $k = k_{\psi}$. Settling time is chosen equal to $3.5\sec$ instead of $0.8\sec$.It is done to make feedback coefficients lower.

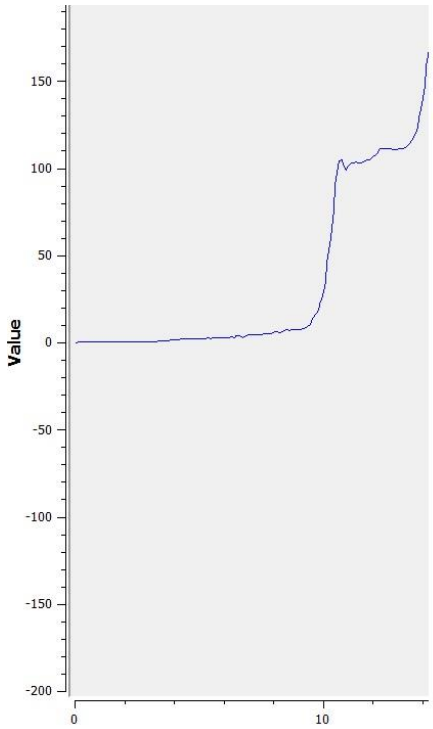

fig. 4.14 Yaw rotation open-loop: test bench 2

Implementation of yaw controller can be described as the following equations:

$$\begin{aligned} u_1(t) &= u_{\min} + u_{hov} - \varepsilon_\psi(t) \\ u_2(t) &= u_{\min} + u_{hov} + \varepsilon_\psi(t) \\ u_3(t) &= u_{\min} + u_{hov} - \varepsilon_\psi(t) \\ u_4(t) &= u_{\min} + u_{hov} + \varepsilon_\psi(t) \end{aligned}, \quad (4.22),$$

where $\varepsilon_\psi(t) = \frac{1}{k} * (-K_1) * DtR * \psi(t) + \frac{1}{k} * (-K_2) * DtR * \dot{\psi}(t)$.

Experiments showed that when all three controllers for attitude control have high coefficients the system has very thin linear zone, which leads to big oscillations. Results are shown on fig. 4.15. It can be seen that's settling time is about 3.5sec and precision from 1 to 2 degree.

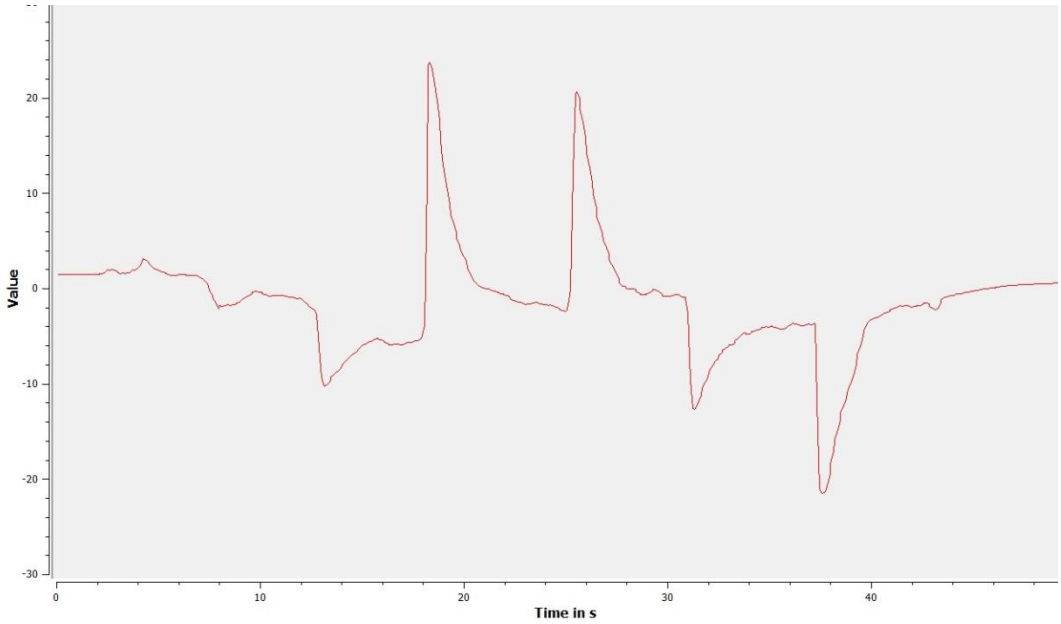

fig. 4.15 Yaw rotation closed-loop: test bench 2

Thereby attitude control can be described by following equations:

$$\begin{aligned} u_1(t) &= u_{\min} + u_{hov} + \varepsilon_\theta(t) - \varepsilon_\psi(t) \\ u_2(t) &= u_{\min} + u_{hov} - \varepsilon_\varphi(t) + \varepsilon_\psi(t) \\ u_3(t) &= u_{\min} + u_{hov} - \varepsilon_\theta(t) - \varepsilon_\psi(t) \\ u_4(t) &= u_{\min} + u_{hov} + \varepsilon_\varphi(t) + \varepsilon_\psi(t) \end{aligned}, \quad (4.23).$$

Results of attitude controller are received by test bench 2 and shown on fig. 4.16. The quadrocopter was inclined several times and one returned to original positions. Moments of inclination are marked as: roll1, roll2, pitch1, pitch2, yaw1, yaw2.

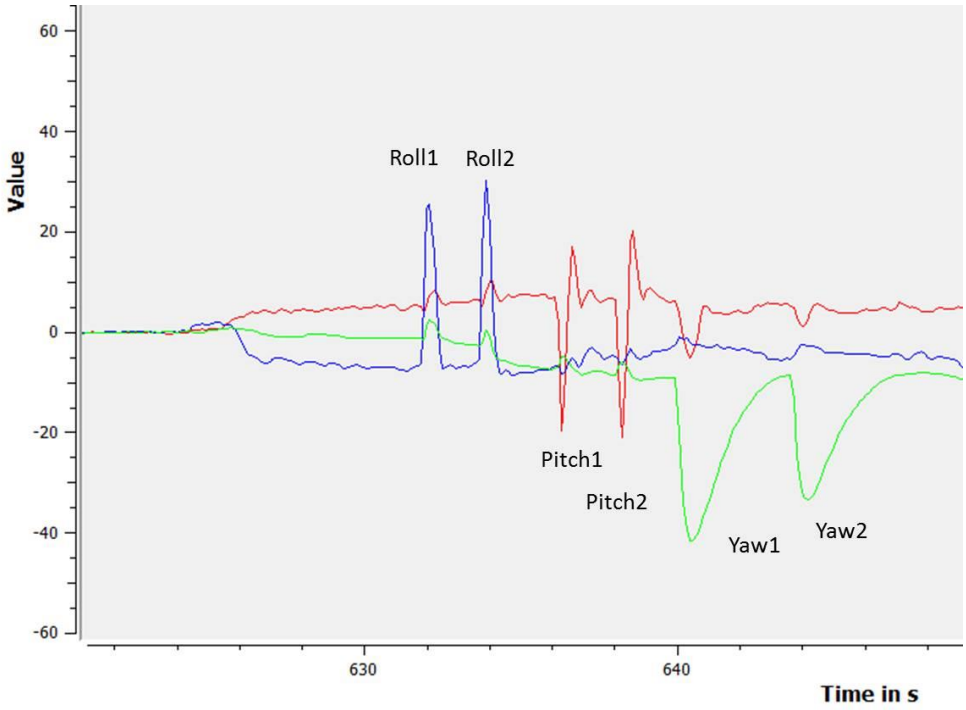

fig. 4.16 Attitude Controller: test bench 2

Dynamics of the system corresponds to required quality. However original positions are changed during the time from 0 to 5 degree. This disadvantage can be eliminated by using integral component and by separating in time pitch-roll and yaw controllers.

4.4 The Altitude Control System

The altitude control system consists of the controller for quadrocopter's hover and position controllers (see section 3.5.1).

Implementation of the hover controller can be described as the following equations:

$$\begin{aligned} u_1(t) &= u_{\min} + u_{hov} + \varepsilon_z(t) \\ u_2(t) &= u_{\min} + u_{hov} + \varepsilon_z(t) \\ u_3(t) &= u_{\min} + u_{hov} + \varepsilon_z(t) \\ u_4(t) &= u_{\min} + u_{hov} + \varepsilon_z(t) \end{aligned}, \quad (4.23),$$

where $\varepsilon_z(t) = \frac{1}{k} * (-K_1) * z(t) + \frac{1}{k} * (-K_2) * \dot{z}(t)$ and $k = \frac{k_T}{m}$.

Result of experiment with a flying prototype for the hover control is shown on fig. 4.17, where a green line is desired height and a red line is a current height.

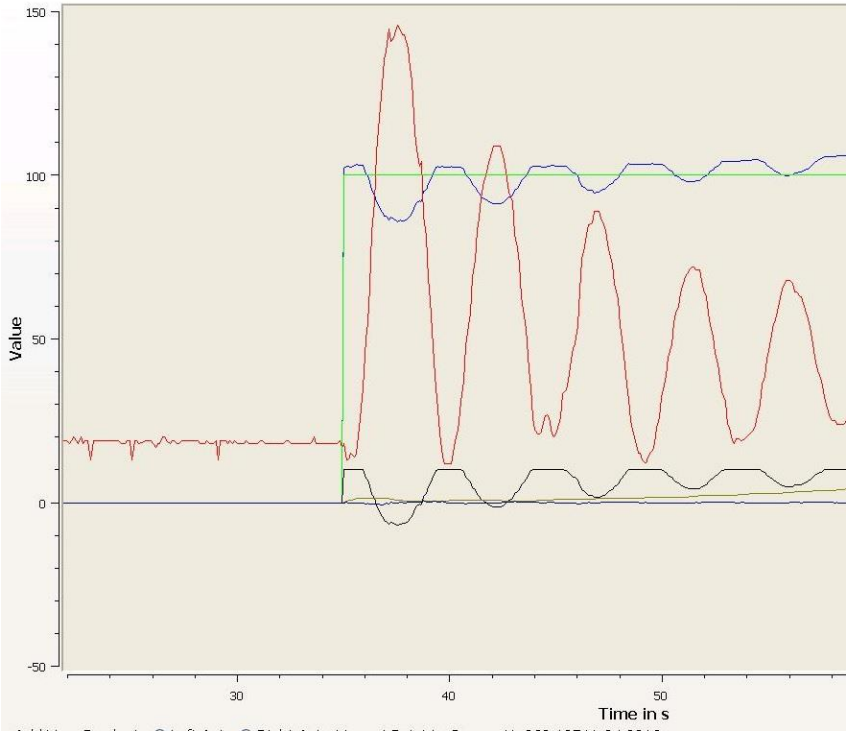

fig. 4.17 The hover Controller with calculated coefficients: flying mode

In the case when coefficients of the hover controller are calculated perfectly, the behavior of current height (red line) should correspond to the modeling result (fig. 3.13c). However it does not since proportional coefficient (K_1) is low and velocity/derivative coefficient (K_2) is extremely low. After slightly increasing of K_1 and big increasing of K_2 (about ten times) the new experiment was conducted (fig. 4.18). The behavior of the system is better, but still is not acceptable. For better results the K_2 was increased (in total about 30 times comparing to original K_2) and an integral component for elimination steady state error was added. Appropriate result is shown on fig. 4.19.

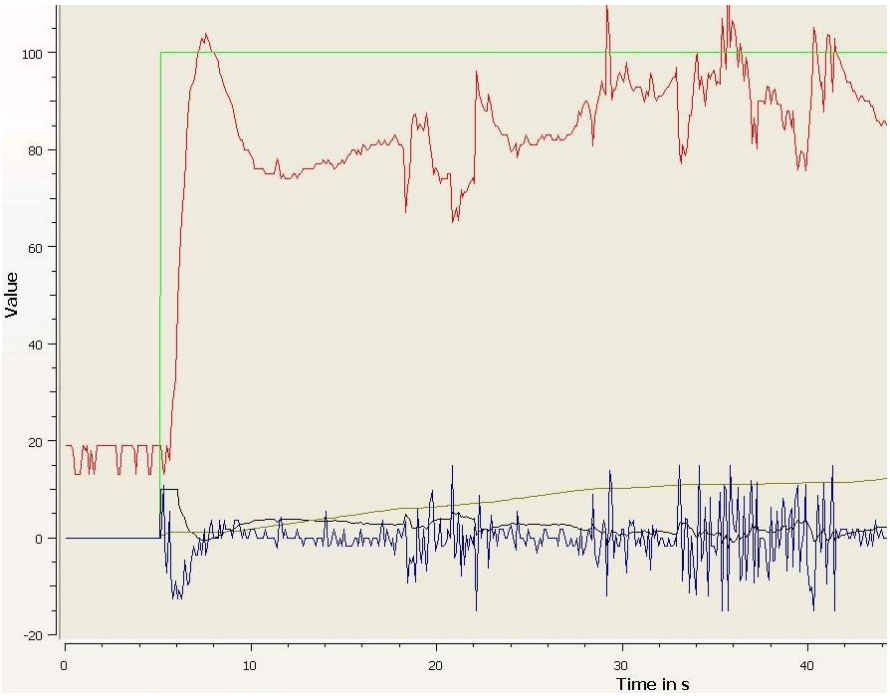

fig. 4.18 The hover Controller with $K_2 = 10$: flying mode

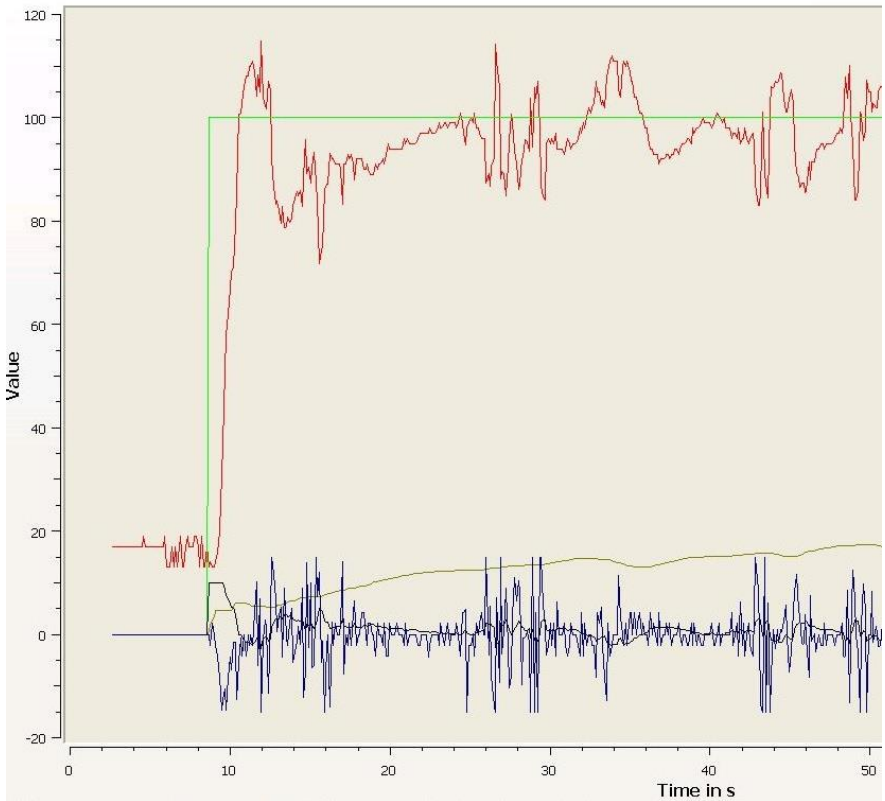

fig. 4.19 The hover Controller with $K_2 = 35$: flying mode

Implementation of position controller corresponds to eq. (3.40). The behavior of the quadrocopter in the flying mode with coefficients used in the simulation is shown on fig. 4.20, where green line corresponds to the desired height and other ones to changing along X and Y.

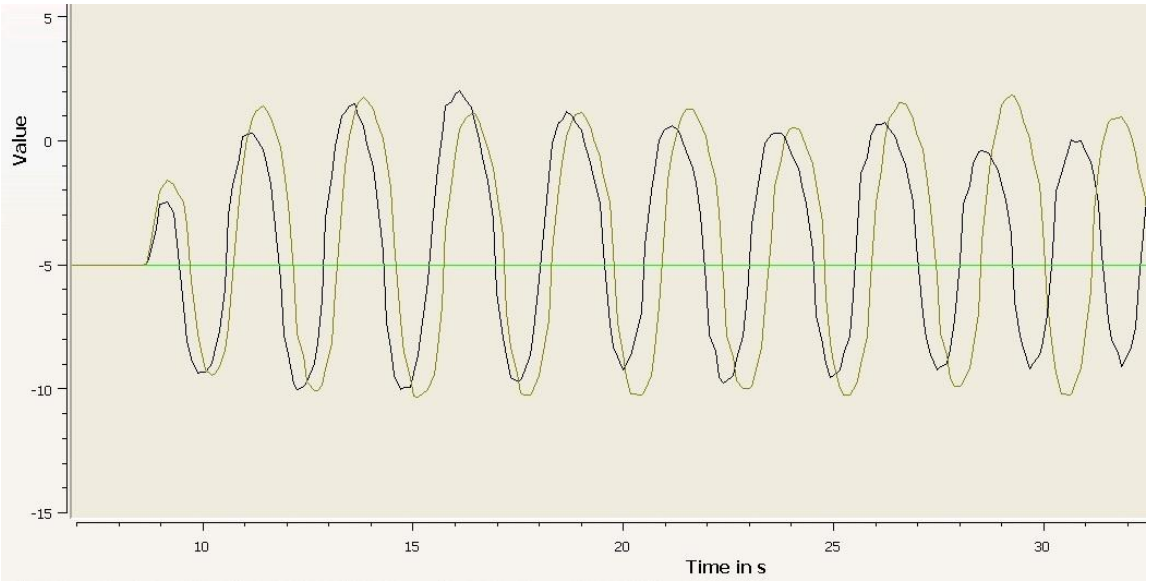

fig. 4.20 Position Controller with calculated coefficients: flying mode

It can be clearly seen that the system is unstable and simulation results (fig. 3.13a and fig. 3.13b) do not match to experimental ones. New coefficients for position controllers were found though adjusting, results are shown on fig. 4.21, where green line is desired position along Y axis and red line is current position of the quadrocopter along Y axis. Also it should be mentioned that new K_1 is ten times less than K_1 from simulation and new K_2 is thirty times more than original K_2. Position error of the system is inside 10 centimeters.

Thereby, it can be concluded that results of the simulation are different from results from real experiment. For the hover controller K_1 from the simulation is close to real one, but K_2 should be increased in thirty times. For the position control, both coefficients obtained from the simulation should be changed, K_1 should be decreased in ten times and K_2 should be increased in thirty times.

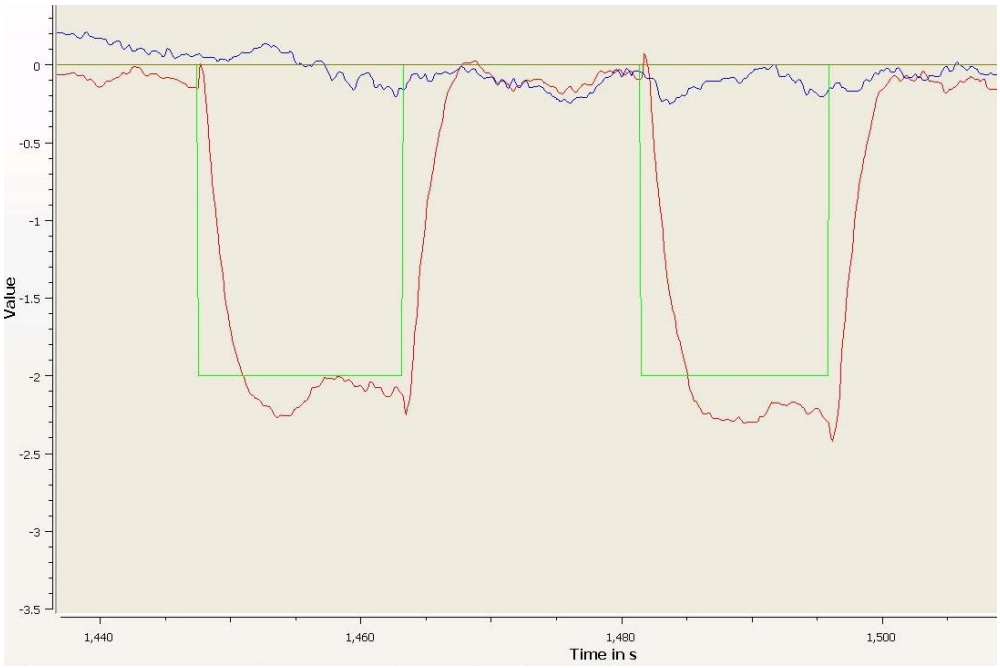

fig. 4.21 Position Controller with adjusted coefficients: flying mode

5 Conclusion and Recommendations

5.1 Conclusion

In this work all stages of a control system development were done. Starting from the analysis of the quadrocopter as a flying object, the mathematical model was created and modified for using in design purposes. After choosing an appropriate design method, the 6DOF control system for the quadrocopter was developed and implemented for using on the real quadrocopter. Experiments for estimation theoretical results were fulfilled.

Process of calculation coefficients for the 6DOF control system is organized as one script in Matlab. The input parameters are the mass of the quadrocopter, the dimensions of the quadrocopter and the gains of the actuators. Based on these inputs coefficients for all 6 controllers (pitch one, roll one, yaw one, hover one and two for the position) are calculated.

The validating procedures for the attitude part of the mathematical model are shown in sections 4.1.3 and 4.3. These procedures are necessary part of the modeling, since they helped to improve the coefficients of the model. Based on the experimental results the parameter R (eq.2.19) for calculation MoI was improved (p. 63). However, the equations for roll and pitch from the mathematical model (1^{st} one and 2^{nd} one from 3.13) cannot be used directly for applying to test bench1and should be modified as it was described in section 4.1.2. The attitude controller, designed based on the improved model, shows in simulation the behavior close to required one. The same controller after implementation shows experimental results closed to the simulation ones.

The control system for hover and position control obtained from simulation shows not adequate results. Only the proportional coefficient for hover control is close to the real one. The proportional coefficient for position control is ten times more that real one and all derivative coefficients for altitude control are about thirty times less than real ones.

Chosen structure for all controllers in general corresponds to PD control. According to the theory all systems that are 1^{st} order and higher (they have pure integration component in their TFs) have no steady state error [11]. However in reality the quadrocopter has steady state error in the attitude (e.g. fig. 4.16) and the

attitude (fig. 4.21), it means that international components should be added to all controllers.

5.2 Recommendations for a future work

Several improvements for the mathematical model and the control system can be done.

First of all, a time delay of the actuator (BLDC and the blade) should be taken into account (eq. 3.11). In this work settling time for the system is about 0.8 seconds and in general typical time for a mechanical constant is about 0.3 seconds. If e.g. a required settling time should be less than 0.3 seconds, a model without mentioned time delay cannot show adequate behavior of the system. Moreover, because of the blade the time delay is different during increasing and decreasing angular velocity of the blade. This factor also should be taken into account.

Another important improvement is about the model by itself. In the work it is continuous one, but it should be transferred to discrete one. The discrete model helps to estimate influence of time delays of the sensors and microcontrollers.

Additionally, the structure of chosen controllers should be changed and supplemented by integral components. In case of making first mentioned improvement, the structure of the controllers should be expanding to three coefficients. Also, a third order polynomial for desired poles should be chosen.

One more improvement can be done with the control system. In current work it is a real time one, but with fixed coefficients. A more profound control system can calculate/recalculate coefficients depend on the current quadrocopter behavior. Also another control algorithm such as e.g. back stepping algorithm or LQR can be used.

Bibliography

1. S. Windnall, J. Peraire, *Relative Motion using Rotating Axes*, Lecture Notes in Dynamics, MIT, Boston, USA, 2008 http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec08.pdf

2. D. Kleppner, R.J. Kolenkow, *An Introduction to Mechanics*, McGraw-Hill, New York, USA, 1973

3. *Drag coefficient*, http://en.wikipedia.org/wiki/Drag_coefficient

4. J.B. Kuipers, *Quaternions and Rotation Sequences*, Princeton University Press, Princeton,USA, 1999

5. R. W. Beard, *Quadrotor dynamics and control*, Lecture notes, Brigham Young University, Provo, USA, 2008, http://rwbclasses.groups.et.byu.net/lib/exe/fetch.php?media=quadrotor:beardsquadrotornotes.pdf access 01.07. 2013

6. O. J. Oguntoyinbo, *PID control of brushless DC Motor and robot trajectory planning and simulation in MATLAB/Simulink*, Thesis, University of Applied Sciences, Vaasa, Finland 2009, access 22.07.2013 available: http://publications.theseus.fi/bitstream/handle/10024/7467/Oludayo%20Oguntoyinbo.pdf

7. E.B. Mikirtumov, *Aircraft Models*, DOSAAF, Moscow, Russia, 1956 (in Russian)

8. G. Miklashevski, *Handbook of Young Aircraft Builder*, ONTI, Moscow, Russia, 1936 (in Russian)

9. E.L. Nikolai, *Theory of gyroscopes*, OGIZ, Moscow, Russia, 1948 (in Russian)

10. R.C. Dorf, R.H. Bishop, *Modern Control Systems*, 11th ed., Prentice Hall, Bergen, USA, 2007

11. C.L. Phillips, R. D. Harbor, *Feedback Control Systems*, 4th ed., Prentice Hall, Bergen, USA, 2000

12. L. Ljung, *System Identification: Theory for the User*, 2nd ed., Prentice Hall, Bergen, USA, 1999

13. R. Nelson, *Flight Stability and Automatic Contro*l, 2nd ed., McGraw-Hill, New-York, USA, 1997

14. Power Bridge BL-Ctrl 1.2, access 15.08.2013 available: http://mikrokopter.de/ucwiki/en/BL-Ctrl_V1.2

15. Motor BLDC KA20-22L, access 15.08.2013 available: http://www.quadroufo.com/product_info.php?cPath=2_9&products_id=47&osCsid=kf0uvl36ffqnkkaefvi9plufg5

16. Blades EPP1045, access 15.08.2013 available: http://www.quadroufo.com/product_info.php?products_id=43&osCsid=dh0c5ejh8ss2a7q63l4bl0jp94

17. Evaluation board EVK1100, access 15.08.2013 available: http://www.atmel.com/tools/EVK1100.aspx

18. Gyroscope IMU3000 combo, access 15.08.2013 available: http://www.invensense.com/mems/gyro/imu3000.html

19. F. Kämpf , *Praktikumsbericht zur Erstellung einer Messstation zwecks Bestimmung der Ubertragungsfunktion zwischen Neigungswinkel und Regelereinstellung einer Quadrokopterachse*, Würzburg University, Germany , 2012

20. T. Luukkonnen, *Modelling and control of quadrocopter, Independent Research project in applied mathematics*, Aalto University, Espoo, Finland 2011

21. S. Bouabdallah, *Design and control of quadrotors with application to autonomous flying*, EPFL, Lausanne, Switzerland, 2007

22. S. Raza and Wail Gueaieb, *Intelligent Flight Control of an Autonomous Quadrotor*, Motion Control, InTech, University of Ottawa, Canada, 2010

23. N. Michael and etc., *The GRASP Multiple Micro UAV Testbed,* Robotics and Automation Magazine, IEEE (Volume: 17 Issue: 3), 2010

24. I. Sonnevend, *Analysis and model based control of a quadrotor helicopter*, Pazmany Peter Catholic University, Budapest, Hungary, 2010

25. *Description of Oemichen's helicopter*, access 12.08.2013 available: http://www.aviastar.org/helicopters_eng/oemichen.php

26. *Description of De Bothezat helicopter*, access 12.08.2013 available: http://www.aviastar.org/helicopters_eng/bothezat.php

27. *Fire-fighter quadrocopter prototype*, access 13.08.2013 available: http://www.jurmol.com/uav.html

28. *Ocean rescue drone*, access 13.08.2013 available: http://www.wired.co.uk/news/archive/2013-03/27/iranian-rescue-robot

29. *A swarm of nano quadrocopters*, access 13.08.2013 available: http://www.youtube.com/watch?v=YQIMGV5vtd4

30. V. Kumar, *Robots that fly and cooperate*, GRASP, University of Pennsylvania, access 13.08.2013 available:

http://www.youtube.com/watch?v=4ErEBkj_3PY

31. R. D'Andrea, *The astounding athletic power of quadrocopters*, Zürich University, access 13.08.2013 available: http://www.youtube.com/watch?v=w2itwFJCgFQ

Appendix A Calculations

1. *Calculation feedback coefficients for pitch orientation: test bench 1*

State-space form from eq. (4.10) looks like:

$$A=\begin{bmatrix} 0 & 1 \\ k_{mg} & 0 \end{bmatrix},\ B=\begin{bmatrix} 0 \\ k \end{bmatrix},\ C=\begin{bmatrix} 1 & 0 \end{bmatrix},\ \text{(a.1)}$$

Characteristic equation can be obtained by combining eq. (a.1) with eq. (3.18) and eq. (3.7) as:

$$\alpha_c(A)=A^2+7.0714*A+25*I=\begin{bmatrix} 0 & 1 \\ k_{mg} & 0 \end{bmatrix}*\begin{bmatrix} 0 & 1 \\ k_{mg} & 0 \end{bmatrix}+7.0714*\begin{bmatrix} 0 & 1 \\ k_{mg} & 0 \end{bmatrix}+$$

$$+25*\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} k_{mg} & 0 \\ 0 & k_{mg} \end{bmatrix}+\begin{bmatrix} 0 & 7.0714 \\ 7.0714*k_{mg} & 0 \end{bmatrix}+\begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}=$$

$$=\begin{bmatrix} k_{mg}+25 & 7.0714 \\ 7.0714*k_{mg} & k_{mg}+25 \end{bmatrix},\ \text{(a.2)}.$$

The 2nd element from eq. (3.5) can be calculated as:

$$\begin{bmatrix} B & A*B \end{bmatrix}^{-1},\text{(a.3)}$$

where $A*B=\begin{bmatrix} 0 & 1 \\ k_{mg} & 0 \end{bmatrix}*\begin{bmatrix} 0 \\ k \end{bmatrix}=\begin{bmatrix} k \\ 0 \end{bmatrix}$, so

$$\begin{bmatrix} B & A*B \end{bmatrix}^{-1}=\begin{bmatrix} 0 & k \\ k & 0 \end{bmatrix}^{-1}=\begin{bmatrix} 0 & k \\ k & 0 \end{bmatrix},\ \text{(a.4)}.$$

Thus, from eq. (a.2) and (a.4), eq. (3.5) can be rewritten as:

$$K=\begin{bmatrix} 0 & 1 \end{bmatrix}*\begin{bmatrix} B & A*B \end{bmatrix}^{-1}*\alpha_c(A)=\begin{bmatrix} 0 & 1 \end{bmatrix}*\begin{bmatrix} 0 & k \\ k & 0 \end{bmatrix}*\begin{bmatrix} k_{mg}+25 & 7.0714 \\ 7.0714*k_{mg} & k_{mg}+25 \end{bmatrix}=$$

$$=\begin{bmatrix}0 & 1\end{bmatrix}*\begin{bmatrix}7.0714*k_{mg}*k & \left(k_{mg}+25\right)*k\\ \left(k_{mg}+25\right)*k & 7.0714*\mathrm{k}\end{bmatrix}=\begin{bmatrix}\left(k_{mg}+25\right)*k & 7.0714*\mathrm{k}\end{bmatrix}$$, or result in short form is:

$$K=[K1 \quad K2]=\begin{bmatrix}\left(k_{mg}+25\right)*k & 7.0714*\mathrm{k}\end{bmatrix}$$, (a.5).

Appendix B. Scripts

List of the files.

	Name	Description
	JforQuad	Function for calculation ToI (eq. 2.19)
01	fullMM.mdl	Implementation of the MM (fig. 2.16).
01	fullMM_Int.m	Initial conditions for simulation the MM
02	linMM.mdl	Implementation of the MM after linearization (eq. 3.13 and 3.39)
02	linMM_Int.m	Initial conditions for simulation linearized MM
	Dpoles.m	Function for calculation desired poles of 2^{nd} order system, based on settling time and damping ration coefficients
	AckCont.m	Function for calculation feedback coefficients of 2^{nd} order system, based on TF in state-space.
03	Att.mdl	Implementation of attitude control for linearized MM
03	Att_Int.m	Initial conditions for attitude control
04	Pos.mdl	Implementation of position control
04	Pos_Int.m	Initial conditions for position control
	DOF2_Ver.m	Script for verification coefficients k and k_{kmg}
	UnresMotRad.mdl	Simulink model for test bench 1 pitch angle
	2013_04_24_processed.xlsx	file with data from experiment

Functions of blocks incorporated in the mathematical model in Simulink: file 'fullMM.mdl'.

function for finding angular acceleration of the quadrocopter (eq. 2.17)

```
function [dtOmx, dtOmy, dtOmz]  = EulerRot(taux, tauy,tauz, Omx, Omy, Omz, J)
%#codegen
Jx = J(1,1); Jy = J(2,2); Jz = J(3,3);
```

```
dtOmx = (taux-(Jz-Jy)*Omy*Omz)/Jx;
dtOmy = (tauy-(Jx-Jz)*Omx*Omz)/Jy;
dtOmz = (tauz-(Jy-Jx)*Omx*Omy)/Jz;
```

function 2 changing in orientation angles based on angular velocity (eq. 2.18)

```
function [dphi1, dtheta1, dpsi1] = OrientAngles(omx, omy,omz,phi, theta, psi )
%#codegen

dphi1   = [1 sin(phi)*tan(theta) cos(phi)*tan(theta)]*[omx omy omz]'
dtheta1 = [0 cos(phi)            -sin(phi)          ]*[omx omy omz]'
dpsi1   = [0 sin(phi)/cos(theta) cos(phi)/cos(theta)]*[omx omy omz]'
```

function 3 TF for blades (example for 2nd blade eq. 2.24, 2.25)

```
function [T2,H2] = TandH2(u,b)
%#codegen

T2 = b(1)*u;
H2 = b(2)*u;
```

function 4 transfer thrust force from $x_b y_b z_b$ to xyz (eq. 2.31)

```
function [Tx,Ty,Tz] = TransTtoXYZ(Tb, phi,theta,psi )
%#codegen

Tx   = (sin(theta)*cos(psi) + sin(phi)*cos(theta)*sin(psi))*Tb;
Ty   = (sin(theta)*sin(psi) - sin(phi)*cos(theta)*cos(psi))*Tb;
Tz   = cos(phi)*cos(theta)*Tb;
```

function 2.1 *Dpoles*
calculation desired poles for 2nd order system, based on required settling time and overshoot

```
% function for calculation desired poles for 2nd order system
% - inputs: settling time, sec; overshooting, in %;
%
% - outputs: two poles
% 09.07.2013 Alex

function [ poles ] = Dpoles1( SetTime , OverSh)
% set input arguments
 if ~exist ('SetTime','var'),     SetTime = 0.8;  end
 if ~exist ('OverSh','var'),      OverSh  = 4.32; end

 % transfer overshooting from % to real value
 overshD=OverSh/100; % overshooting in %

 % calculate parameters for 2nd order TF
```

```
  % damping ratio
    zeta=abs(log(overshD)*1/pi*sqrt(1/((log(overshD)/pi)^2+1)));
  % natural frequency
    wn=4/(SetTime*zeta);

 % calculate desired poles
  RealPart=-zeta*wn; ImPart=wn*sqrt(1-zeta^2);
  poles = [RealPart+ImPart*1i RealPart-ImPart*1i]; % desired poles
```

function 2.2 AckContSim
calculation feedback coefficients for 2nd order system based on Ackerman equation

```
% function for calculation feedback coefficients for 2nd order system
% - inputs: Ts, a0, b0, a1, b1
%
% - outputs: vector K with two coefficients k1 and k2
% 09.08.2013 Alex

function [ K ] = AckContSim( Ts, b0, a0, a1 )
% set input arguments
 if ~exist ('Ts','var'),    Ts = 0.8; end
 if ~exist ('b0','var'),    b0 = 1; end
 if ~exist ('a0','var'),    a0 = 0; end
 if ~exist ('a1','var'),    a1 = 0; end

%***********
% state -space form
% *************
  A=[0 1; a0 a1];
  B=[0; b0];
  C=[1 0];
  D=[0];

% find desired poles
 Dp = Dpoles(Ts);
% find feedback coefficients
 K=acker(A,B,Dp)

% simulation
 SysF = ss((A-B*K), B,C, D);
figure(1)
initial(SysF, [1;0])
grid on
hold on
```

Script 3.1 '*DOF2_ver.m*'

```
% 2013-08-05
% Task
 % unrestricted motion
```

```
 % The quadro was inclined and released. The data during the falling
(unrestricted motion)
 % were recorded. Records are saved in file 2013_04_24_processed.xlsx.

% Description
 % Code of this file
 % - step 1 processes data and plots data from file
2013_04_24_processed.xlsx
 % - step 2 calculates unrestricted motion based on model
 % - step 3 find optimal coefficient kmg for the model

clear all
close all
% STEP 1
% put data from file in variable 'data'
 test_data = importdata('2013_04_24_processed.xlsx');
 data=test_data.data.Tabelle1
% transfer degree in radian
dTr=pi/180;

% data from test 1
 % create time and angle vectors
 t0=data(1,1);
 time1=zeros(6,1);
 for i=1:6
  time1(i,1)=data(i,1)-t0;
 end
 vector_angularPosition1=data(1:6,2)*dTr;

 % plot time and angle vectors
 figure(1)
 p1=plot(time1,vector_angularPosition1,'-*')
   ylabel('AngularPosition,rad')
   xlabel('time, s')
   grid on
 set(p1,'Color','red','LineWidth',1)

% test 2
 % create time and angle vectors
 t0=data(1,4);
 time2=zeros(5,1);
 for i=1:5
  time2(i,1)=data(i,4)-t0;
 end
 vector_angularPosition2=data(1:5,5)*dTr;

 % plot time and angle vectors
 hold on
 p2=plot(time2,vector_angularPosition2,'-*')
 set(p2,'Color','green','LineWidth',1)


% test 3
 % create time and angle vectors
 t0=data(1,7);
 time3=zeros(4,1);
```

```
 for i=1:4
  time3(i,1)=data(i,7)-t0;
 end
 vector_angularPosition3=data(1:4,8)*dTr;

 % plot time and angle vectors
 hold on
 p3=plot(time3,vector_angularPosition3,'-*')
 set(p3,'Color','blue','LineWidth',1)

% test 4
 % create time and angle vectors
 t0=data(1,10);
 time4=zeros(2,1);
 for i=1:2
  time4(i,1)=data(i,10)-t0;
 end
 vector_angularPosition4=data(1:2,11)*dTr;

 % plot time and angle vectors
 hold on
 p4=plot(time4,vector_angularPosition4,'-*')
 set(p4,'Color','yellow','LineWidth',2)

% STEP 2
% intial conditions for kmg calculation
 m=730*10^-3     ; % kg
 mM = 57*10^-3; % mass of one motor
 l = 21*10^-2; % m
 J = JforQuad(m, mM, l, l);
 Jy = J(1,1); % 0.0139
 b= 0.11; % m
  g=9.8;
 Jtb1=Jy+m*(b)^2; % 0.0227
 kmg=m*g*b/Jtb1; % 34.64

amountOfTests=4; % how many times experiments were made
%prepare initial condition for each experiment
phiInitial=zeros(amountOfTests,1);
 phiInitial(1,1)=vector_angularPosition1(1,1);
 phiInitial(2,1)=vector_angularPosition2(1,1);
 phiInitial(3,1)=vector_angularPosition3(1,1);
 phiInitial(4,1)=vector_angularPosition4(1,1);
% for comparing data simulation time = 0.5 seconds is enough
 Ts = 0.01;
 n=50;
 integrator= 'ode45';
 sim_model = 'UnresMotRad';

 % input and output signals
 y_data      = zeros(n,amountOfTests);
 t_data      = zeros(n,amountOfTests);


 % Simulation
  for jj=1:amountOfTests
```

```
   phi= phiInitial(jj,1)
   simoptions = simset('Solver',integrator,'MaxRows',0);
   eval(['[sizes,x0] = ' sim_model '([],[],[],0);']);
   ref_old   = 0;
   t       = -Ts;
  for i=1:n,
   t = t + Ts;
  %simulation
   utmp=[t-Ts,ref_old;t,ref_old];
   simoptions.InitialState=x0;
   [time,x0,y] = sim(sim_model,[t-Ts t],simoptions,utmp);
   x0 = x0(size(x0,1),:)';
   y  = y(size(y,1),:)';
  % save output and time value for current step
   y_data(i,jj)       = y;
   t_data(i,jj)       = t;
  end
  figure(1);
  axis([0 0.6 0 0.4])
  hold on
  title({'Simulation with theoretical kmg=34.64','Colorful lines are data
from experiment, Black ones from simulation'}) ;
  h=plot(t_data(:,jj)',y_data(:,jj)');
  set(h,'LineWidth',1,{'Color'},{'black'});
  xlabel('time, s'); ylabel('AngularPosition,radian');
  grid on
 end



 % STEP 3 optimization kmg
 % as soon as model with kmg=33.5455, calcualted based on theory,
 % shows the behaviour of the system not precise, the new value of kmg
 % should be found.
 % after several attepts it was found that optimal value is around
 kmg=22.3006;

 % to pove this
 % plot experiment data in new figure (for example fig.2 )
 figure(2)
 p1=plot(time1,vector_angularPosition1,'-*')
  title({'Simulation with optimal kmg=22.3006','Colorful lines are data
from experiment, Black ones from simulation'}) ;
   ylabel('AngularPosition,rad')
   xlabel('time, s')
   grid on
 set(p1,'Color','red','LineWidth',1)
 hold on
 p2=plot(time2,vector_angularPosition2,'-*')
 set(p2,'Color','green','LineWidth',1)
 hold on
 p3=plot(time3,vector_angularPosition3,'-*')
 set(p3,'Color','blue','LineWidth',1)
 hold on
 p4=plot(time4,vector_angularPosition4,'-*')
 set(p4,'Color','yellow','LineWidth',2)
```

```
% simulate the system with new kmg=22.3006

% for comparing data simulation time = 0.5 seconds is enough
Ts = 0.01;
n=50;
integrator= 'ode45';
sim_model = 'UnresMotRad';

% input and output signals
y_data      = zeros(n,amountOfTests);
t_data      = zeros(n,amountOfTests);


%**************
% Simulation
for jj=1:amountOfTests
 phi= phiInitial(jj,1)
 simoptions = simset('Solver',integrator,'MaxRows',0);
 eval(['[sizes,x0] = ' sim_model '([],[],[],0);']);
 ref_old   = 0;
 t       = -Ts;

 for i=1:n,
  t = t + Ts;

 %simulation
  utmp=[t-Ts,ref_old;t,ref_old];
  simoptions.InitialState=x0;
  [time,x0,y] = sim(sim_model,[t-Ts t],simoptions,utmp);
  x0 = x0(size(x0,1),:)';
  y  = y(size(y,1),:)';

 % save output and time value for current step
  y_data(i,jj)       = y;
  t_data(i,jj)       = t;

 end
 figure(2);
 hold on
 axis([0 0.6 0 0.4])
 h=plot(t_data(:,jj)',y_data(:,jj)');
 set(h,'LineWidth',1,{'Color'},{'black'});
 xlabel('time, s'); ylabel('AngularPosition,radian');
 grid on
end
```