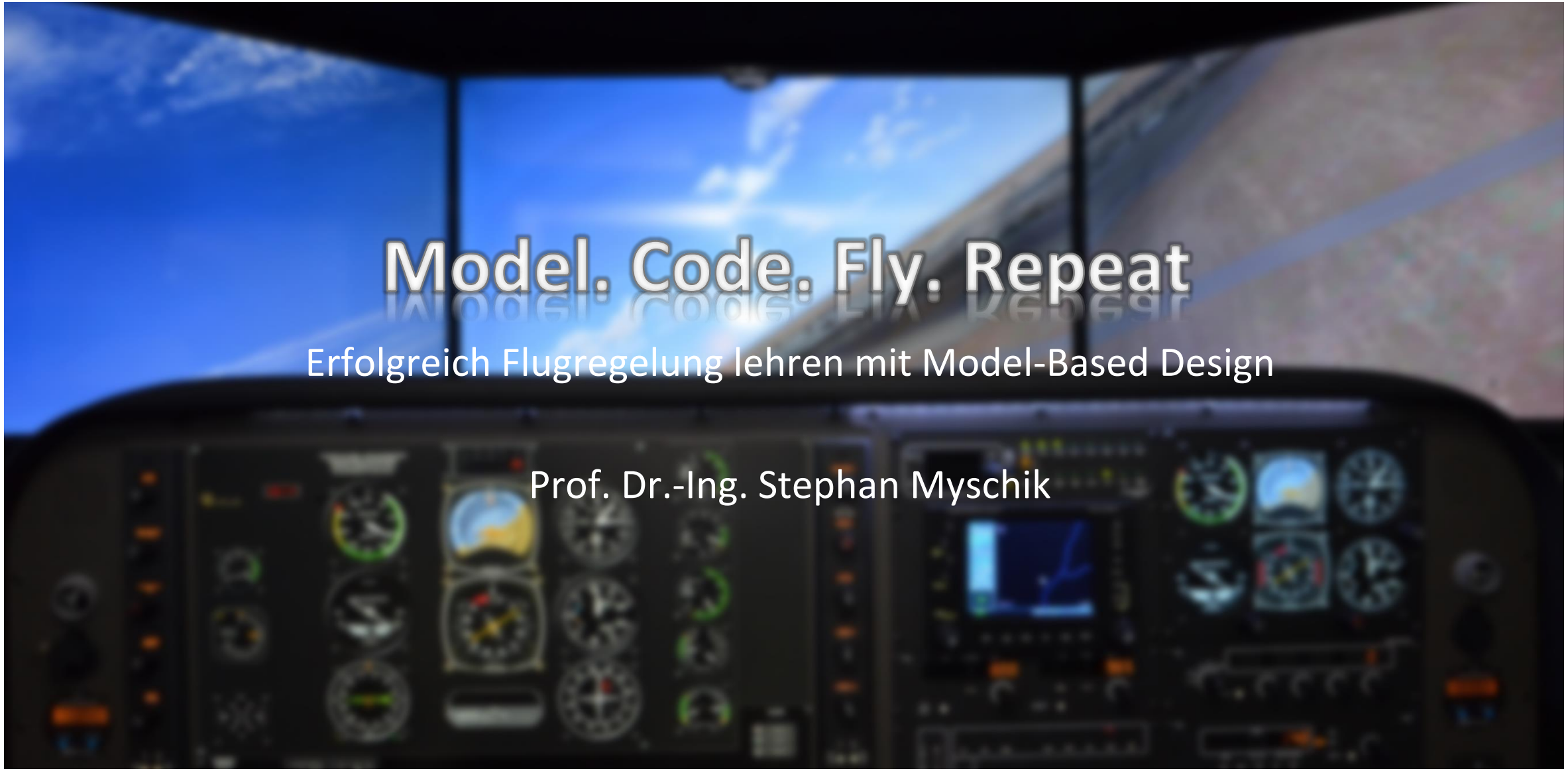
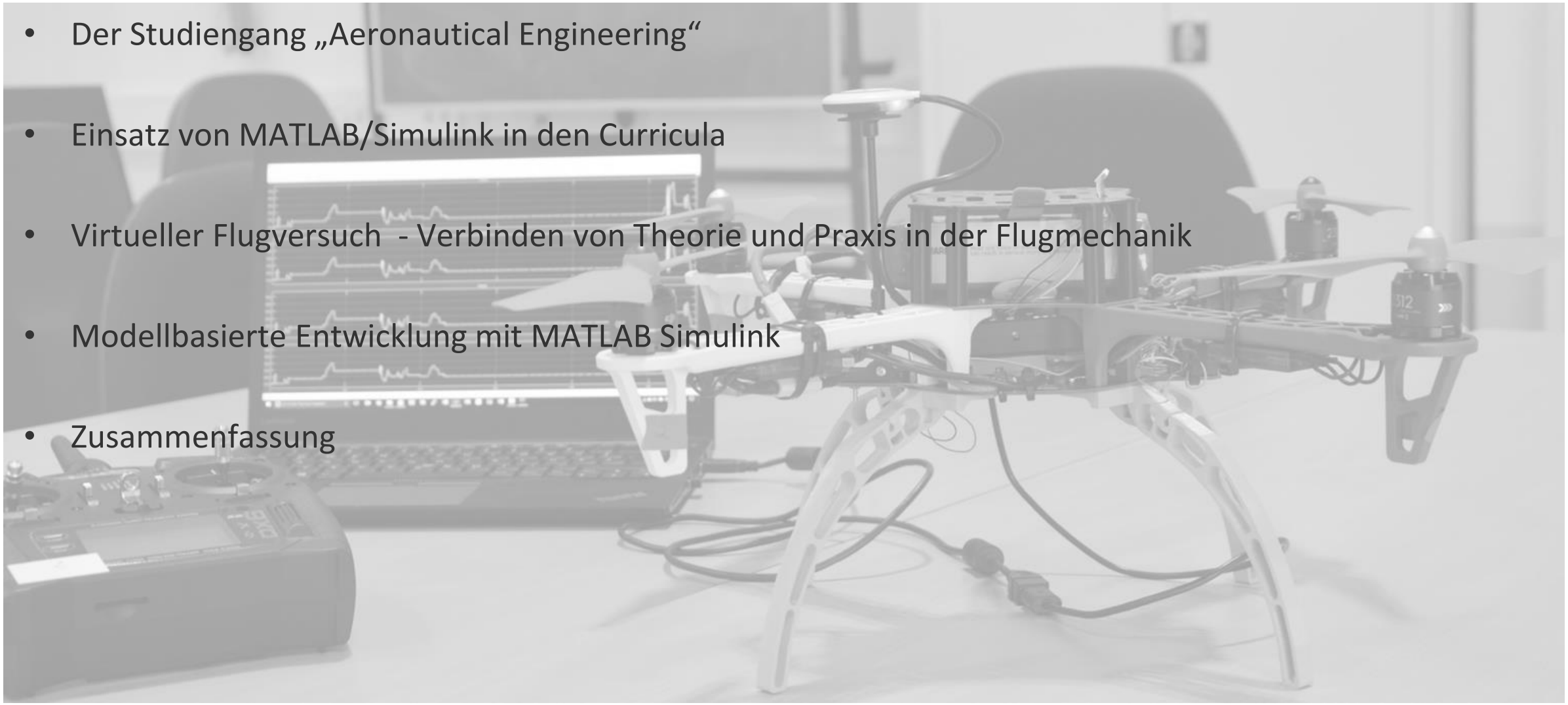


Model. Code. Fly. Repeat

Erfolgreich Flugregelung lehren mit Model-Based Design

Prof. Dr.-Ing. Stephan Myschik



- Der Studiengang „Aeronautical Engineering“
- Einsatz von MATLAB/Simulink in den Curricula
- Virtueller Flugversuch - Verbinden von Theorie und Praxis in der Flugmechanik
- Modellbasierte Entwicklung mit MATLAB Simulink
- Zusammenfassung

Studiengangkonzept

- **Duales Bachelor-Studium am HAW Bereich der UniBw München**
  - Integration von **akademischem Studium** und **fliegerischer Ausbildung** der Piloten der Bundeswehr.
  - Fliegen mit engem Bezug zu den Ingenieurwissenschaften (u.a. Flugphysik, Flugzeugbau, Flugregelung, Flugantriebe)
  - **Ziel** ist eine nutzbringende und motivierende Wechselwirkung von Theorie (Ingenieurwissenschaft) und Praxisbezug (fliegerische Ausbildung)
  - Abschlussgrad: *Bachelor of Engineering* (B. Eng.)
  - zusätzlich: Fluglizenz
- **Hoher Praxisbezug im Studium bei gleichzeitigem fachlichen Tiefgang aufgrund der Zielrichtung der Ausbildung erforderlich!**

Curriculum und Einsatz von MATLAB / Simulink

1. Studienjahr

- Mathematik
- Wissenschaftliches Rechnen

2. Studienjahr

- Regelungstechnik
- Flugmechanik/Flugregelung

3. Studienjahr

- Simulatortechnik und Flugzeugsysteme
- Flugmechanik/Flugregelung
- „Model-Based Design mit MATLAB und Simulink“

4. Studienjahr

Flugausbildung (Jet /Prop/Heli/RPA)

Seminar Aeron. Eng.

Bachelorarbeit

Verleiht Flügel...

... und noch viel mehr!

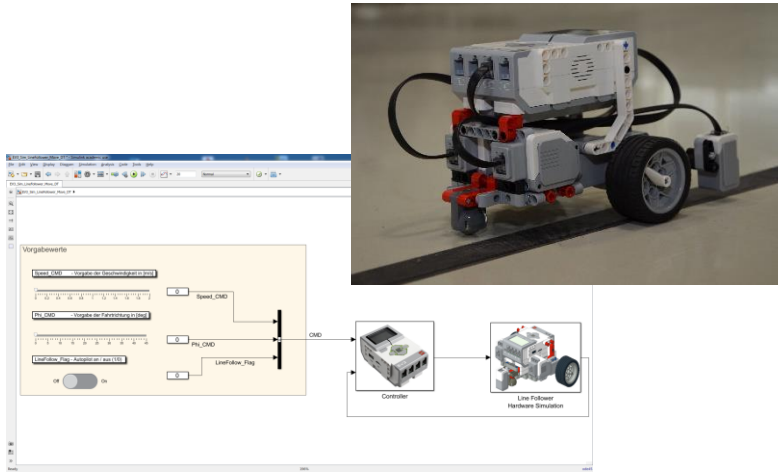
Voraussetzungen:

- Allgemeine oder fachgebundene Hochschulreife: (Fach-)Abitur
- Interesse an der Fliegerei
- Flugtauglichkeit (Assessment durch die Bundeswehr)

Wie setzen wir MATLAB / Simulink ein?

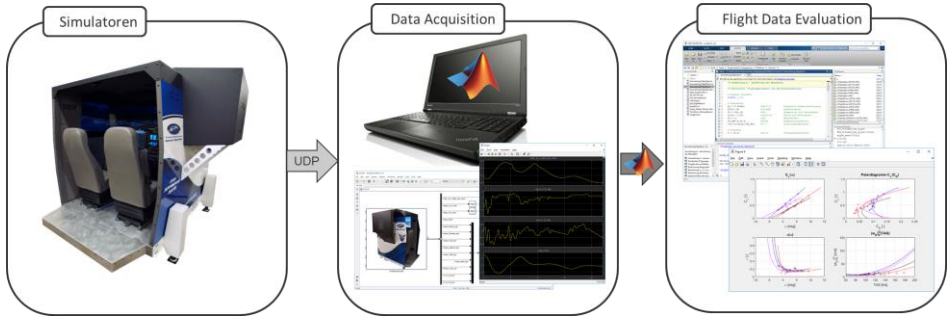
Regelungstechnik

- Einführung in die Grundlagen der Regelungstechnik
- Regelungstechnisches Praktikum zur angewandten Vertiefung des Vorlesungsstoffes
- Erster Kontakt der Studenten mit Simulink *(und mit Regelungstechnik …)*
- Verwendung von LEGO EV3 für praktische Versuche



Flugmechanik (Frühjahrstrimester)

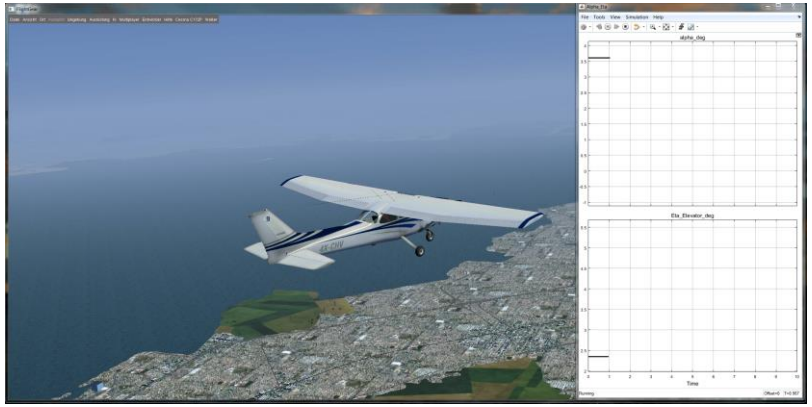
- Flugleistungen, Flugdynamik und Flugeigenschaften
- Simulink zur Verdeutlichung von flugphysikalischen Effekten
- **Virtuelle Flugversuche** im hauseigenen Simulator-Zentrum
- Datenaufnahme /-auswertung mit MATLAB und Simulink



Wie setzen wir MATLAB / Simulink ein?

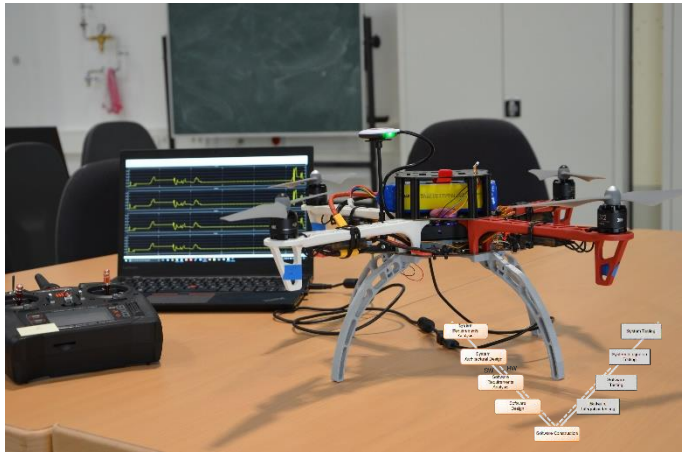
Flugregelung (Herbsttrimester)

- Flugeigenschaftsforderungen, Flugregler und Autopiloten
- Simulink zur Verdeutlichung von flugphysikalischen Effekten
- Entwurf und Evaluation von Flugreglern im Simulator-Zentrum



Model-Based Design mit MATLAB und Simulink

- Prozesskonformer Entwurf, Implementierung und Verifikation von Regelungsalgorithmen am Beispiel eines Quadrocopter
- Hands-On Erlebnis für interessierte Studenten für eine konkrete Applikation
- Verwendung von Off-The-Shelf Komponenten und MathWorks Pilot Support Package für Pixhawk PX4



Virtueller Flugversuch mit MATLAB / Simulink

„Flugmechanik und Flugregelung“

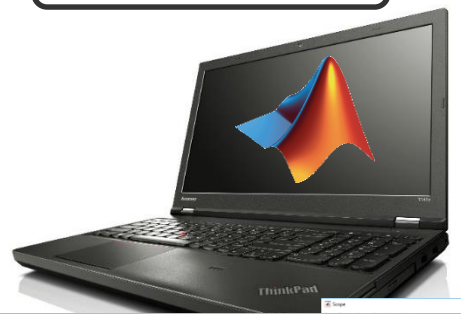
Ziel: Herstellen der Verbindung zwischen flugmechanischer Theorie und Praxis

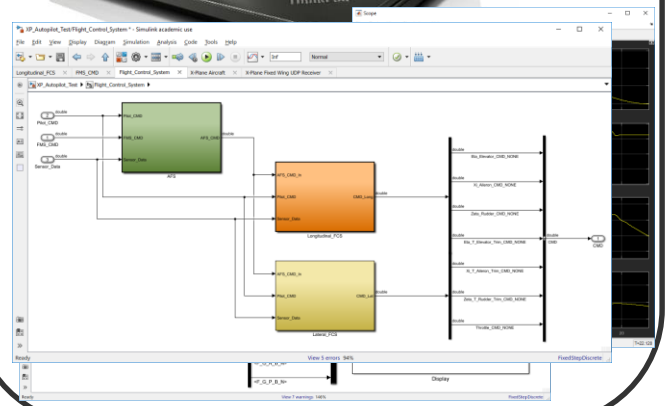
Simulatoren

UDP

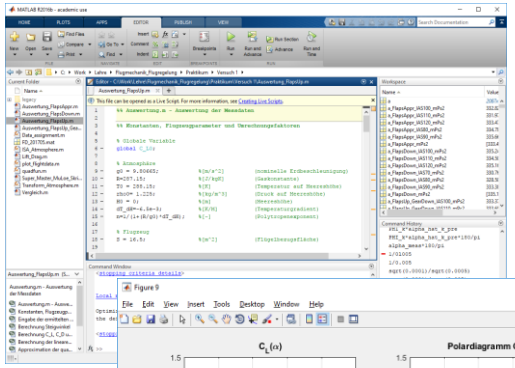
UDP

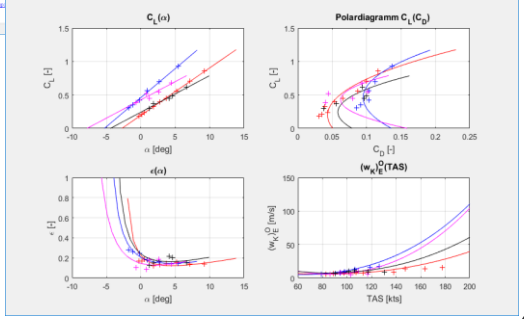
Data Acquisition





Flight Data Evaluation





Ablauf

Mission Briefing / Vorbereitung

- Wiederholung flugmechanische Theorie
- Festlegung der Flugversuchsziele

Start / Flug zum Zielgebiet

- Start des Flugzeugs und Fliegen zum Zielgebiet
- Anwendung der “Normal Procedures” aus der Simulatorausbildung

Flugversuch

- Durchführung der Flugaufgaben ausgehend von einem stationären Flugzustand
- Aufzeichnung der Daten (in wechselnden Rollen)

Landung

- Rückkehr zum Startflughafen ASAP
- Landung

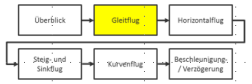
Mission Debriefing und Ausarbeitung

- Finaler Plausibilitätscheck / Auswertung der Daten mit MATLAB Skripten
- Interpretation der Daten und Ausarbeitung durch die Studenten

Durchführung der Flugaufgabe

Gleitflug

Leistungsparameter für effektivsten Gleitflug



10

Berechnungsansatz :

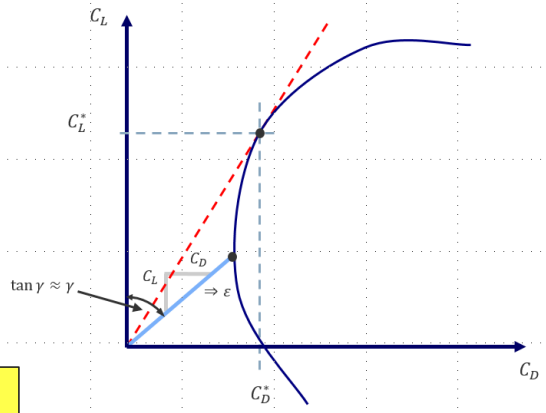
Gleitflug ist am effektivsten, wenn das Verhältnis $\frac{C_D}{C_L}$ minimal wird, also wenn die Gleitzahl ε und damit der Gleitwinkel γ minimal wird. (Tangente an die Polare!)

Annahme:

Symmetrische quadratische Polare:

$C_D = f(C_L) = C_{D0} + k\,C_L^2$

$$\varepsilon = \frac{C_D}{C_L} = \frac{C_{D0} + k\,C_L^2}{C_L} = \frac{C_{D0}}{C_L} + kC_L$$

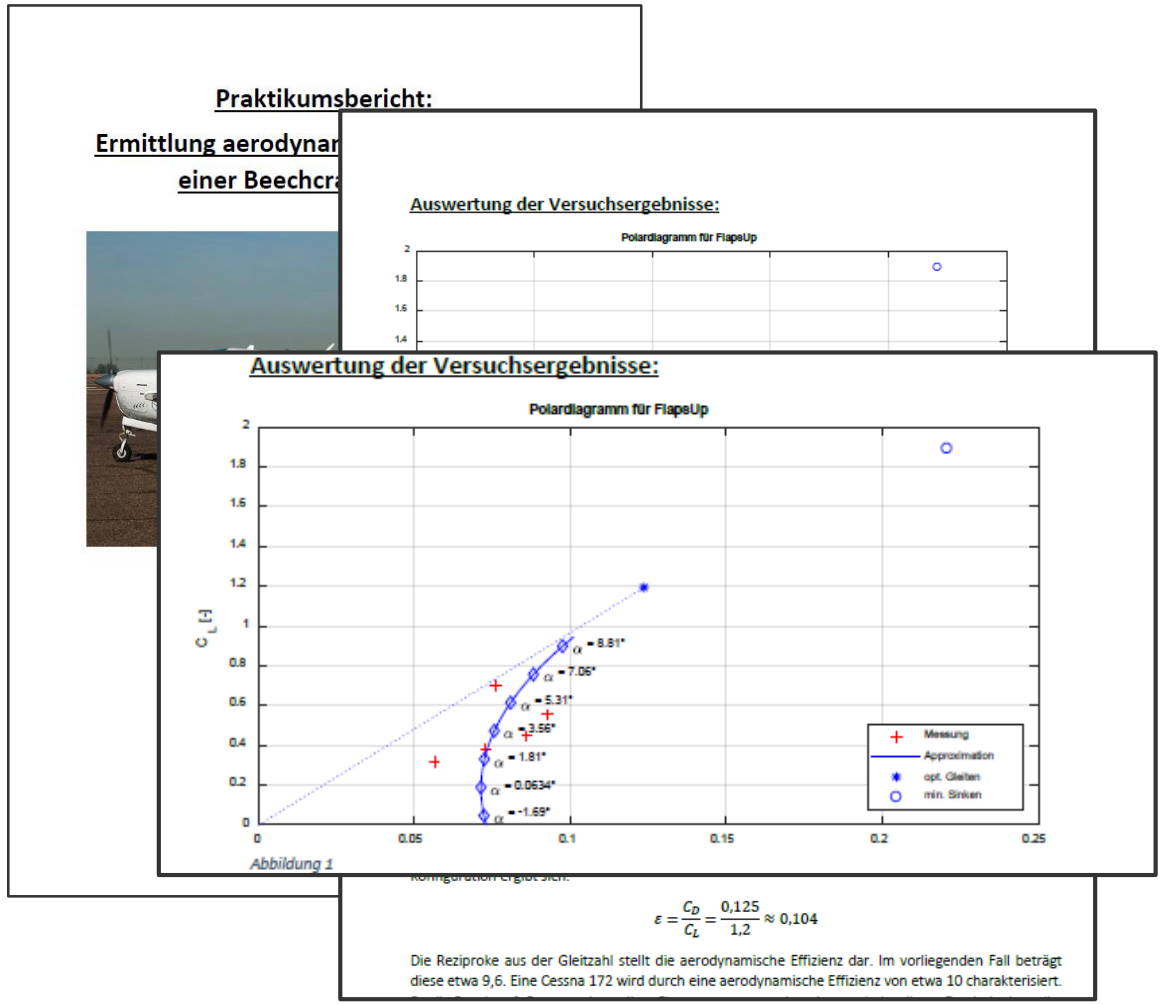


Kurvendiskussion der Gleichung zur Minimumsfindung:

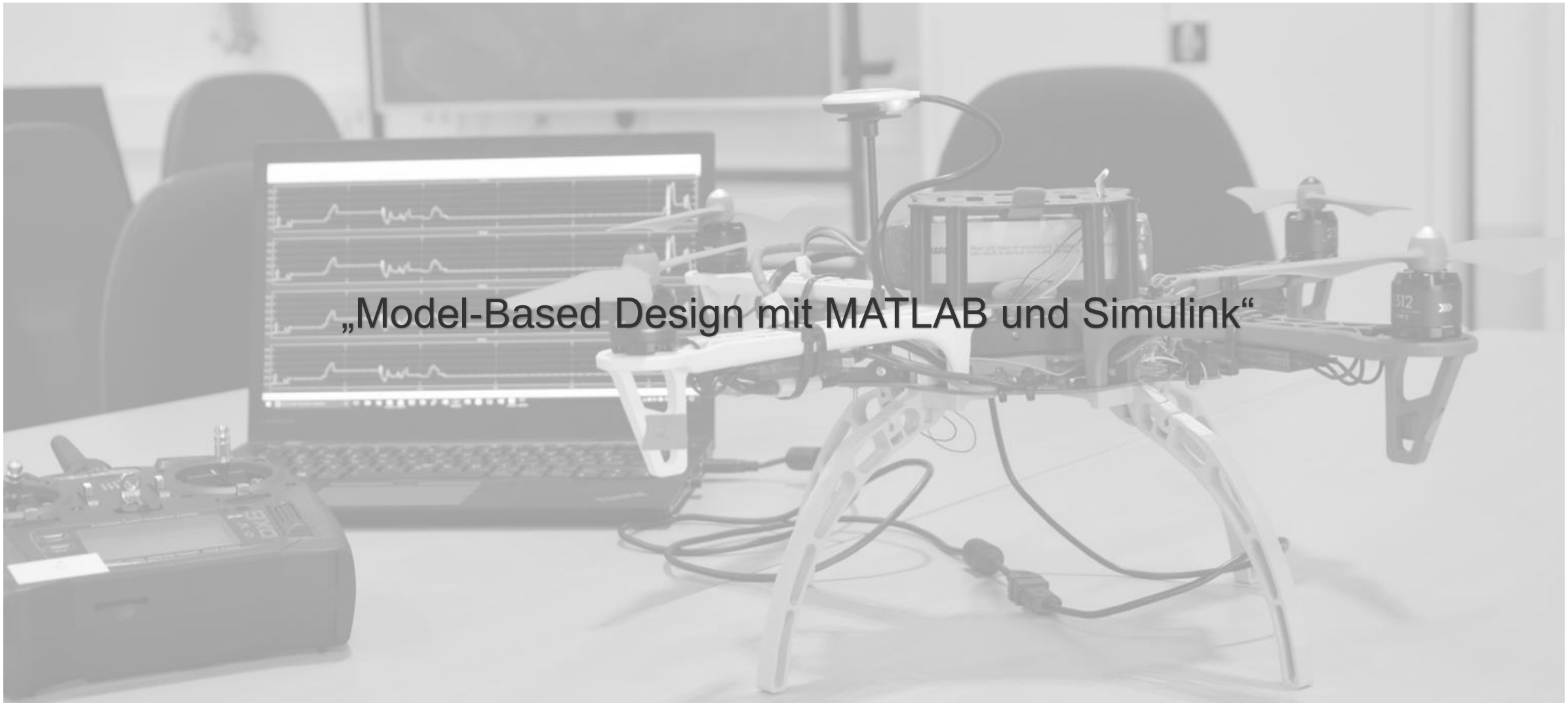
$$\frac{\partial \varepsilon}{\partial C_L} = 0 \quad \Rightarrow -\frac{C_{D0}}{C_L^2} + k = 0$$

Kapitel 6: Stationäre Flugzustände

Theorie aus der Vorlesung



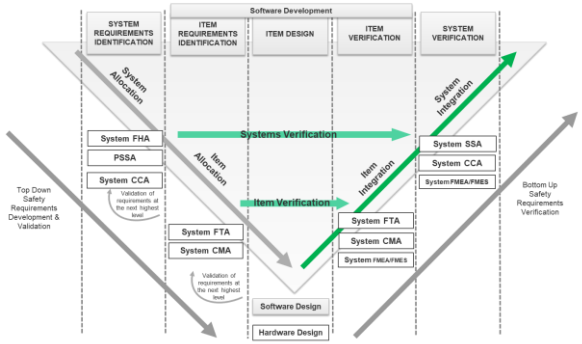
Auswertung der Studenten

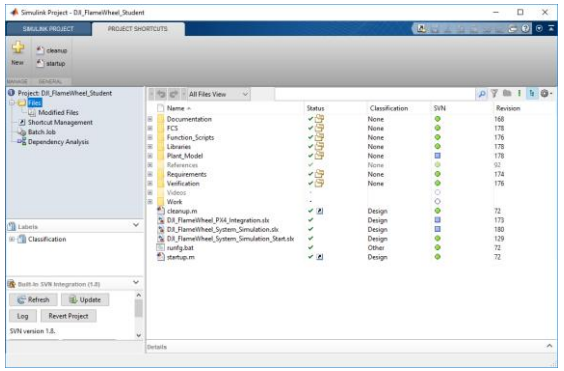


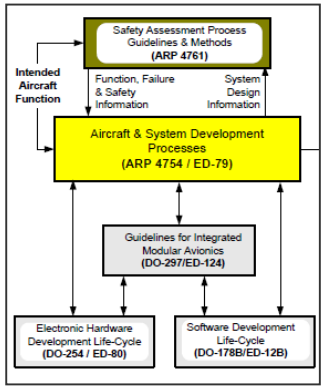
„Model-Based Design mit MATLAB und Simulink“

Übersicht

- Wahlpflichtfach für die Studiengänge „Aeronautical Engineering“ und „Luftfahrttechnik / Wehrtechnik“
- Ziel : Grundlegender Überblick über moderne, industrierelevante Software-Entwicklungsprozesse mit Fokus auf Luftfahrt (DO-178 B/C, ARP 4754, ARP 4761,…)
- Hoher Praxisanteil durch Umsetzung von Regelungsalgorithmen auf fliegende Testplattformen (Quadrocopter, Luftschiff,…)
- Anwendung industrietypischer Entwicklungswerkzeuge (Versionskontrolle, automatische Codegenerierung, Modell-/Codeverifikation, …)



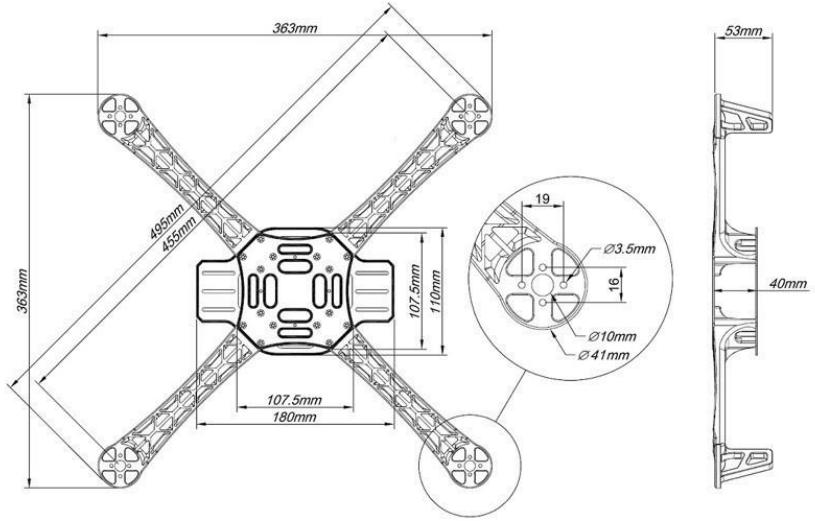




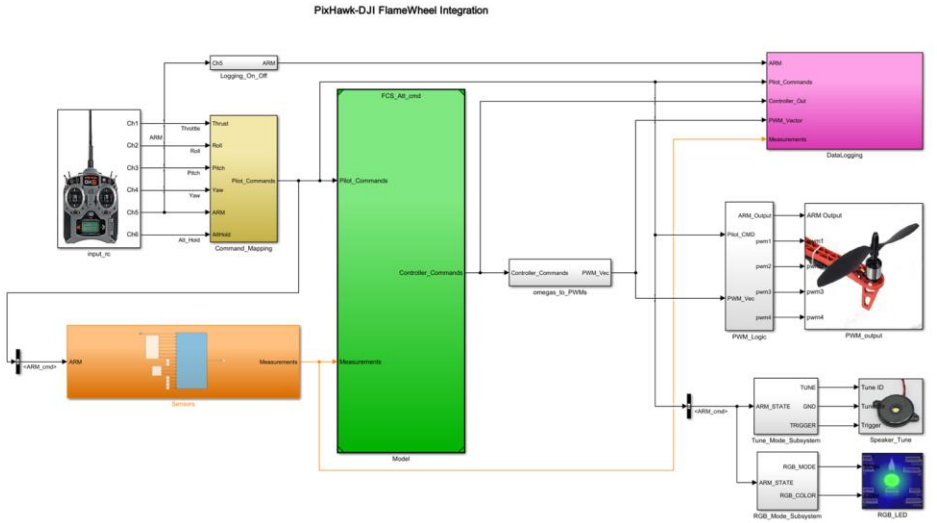
Vorlesungsinhalte

- Einführung in die Modellbasierte Entwicklung
- Entwicklung von System- und Komponentenanforderungen
- Grundlagen der Systemmodellierung mit Simulink
- Methoden zur Analyse von Streckenmodellen
- Umsetzung von Flugreglern und Logikelementen
- Statische Modellanalyse zur Sicherstellung der Softwarequalität
- Automatische Implementierung durch Codegenerierung
- Verifikationsmethoden auf Modellebene zur Funktionsabsicherung
- Absicherungsmethoden auf Codeebene
- Systemintegration und Test

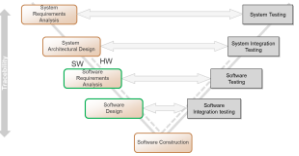
Fliegende Testplattform

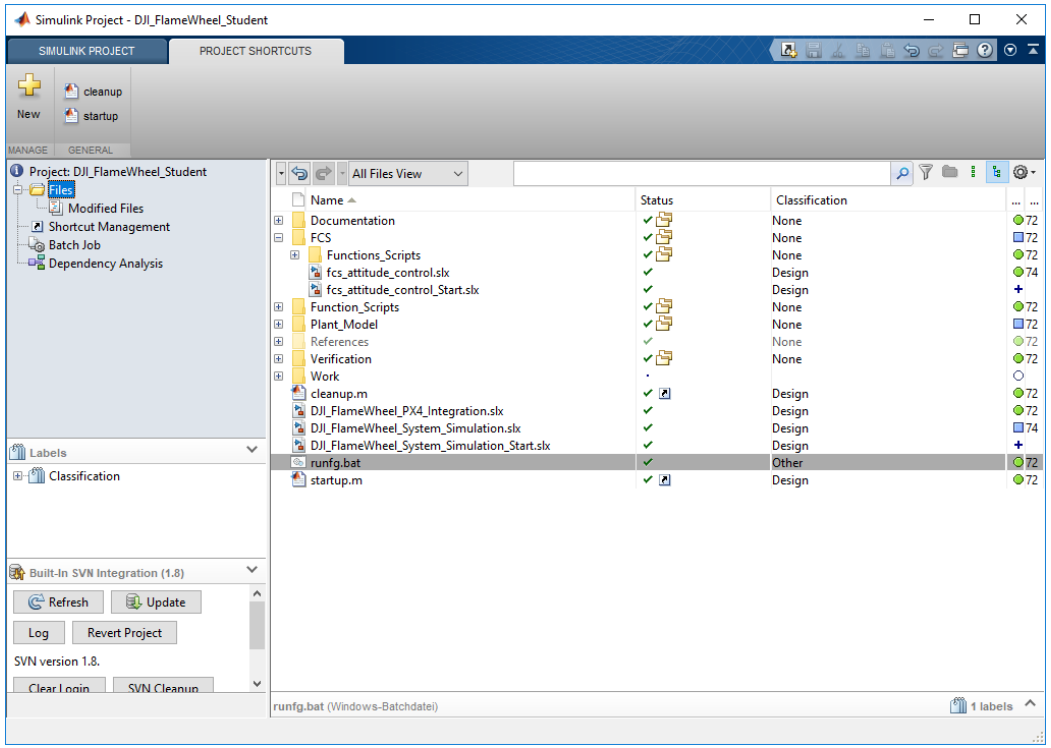


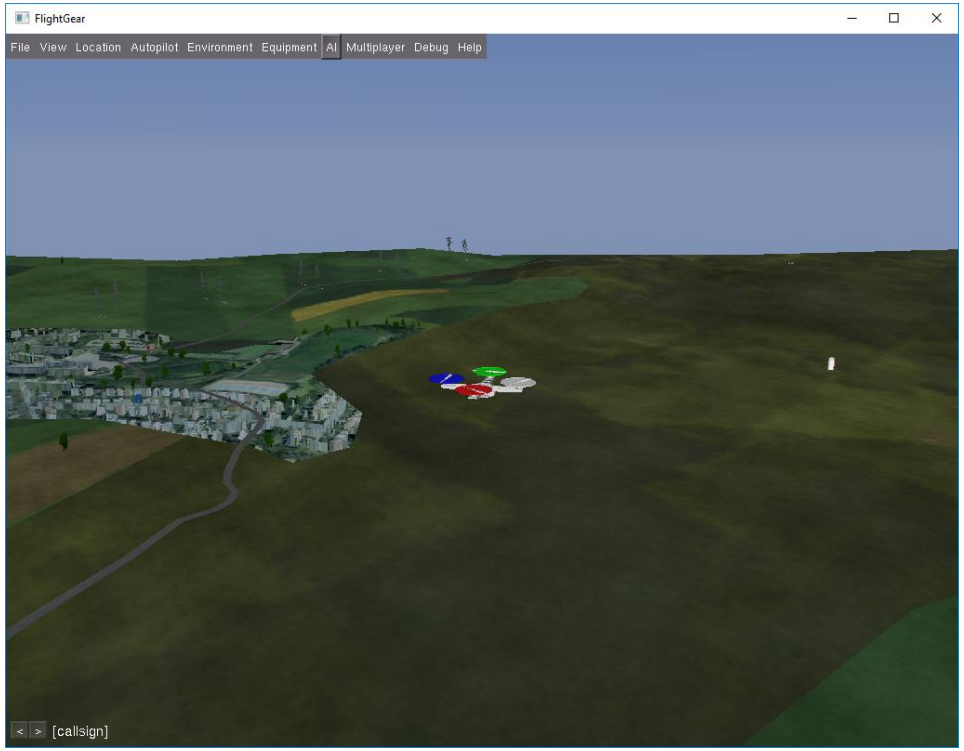
- Quadrotor DJI Flamewheel 450 ARF Kit
- PixHawk Flight Controller basierend auf ARM Cortex M4
- 3 Achsen Beschleunigungs-/Drehratensensoren
- Barometrischer Höhenmesser
- Lage-, Geschwindigkeit- und Positionsinformationen bereitgestellt durch integriertes Navigationssystem
- 14 PWM Ausgänge für Servos



Grundlagen der Systemmodellierung mit Simulink

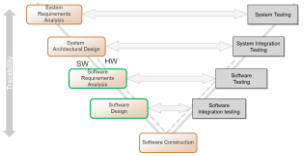


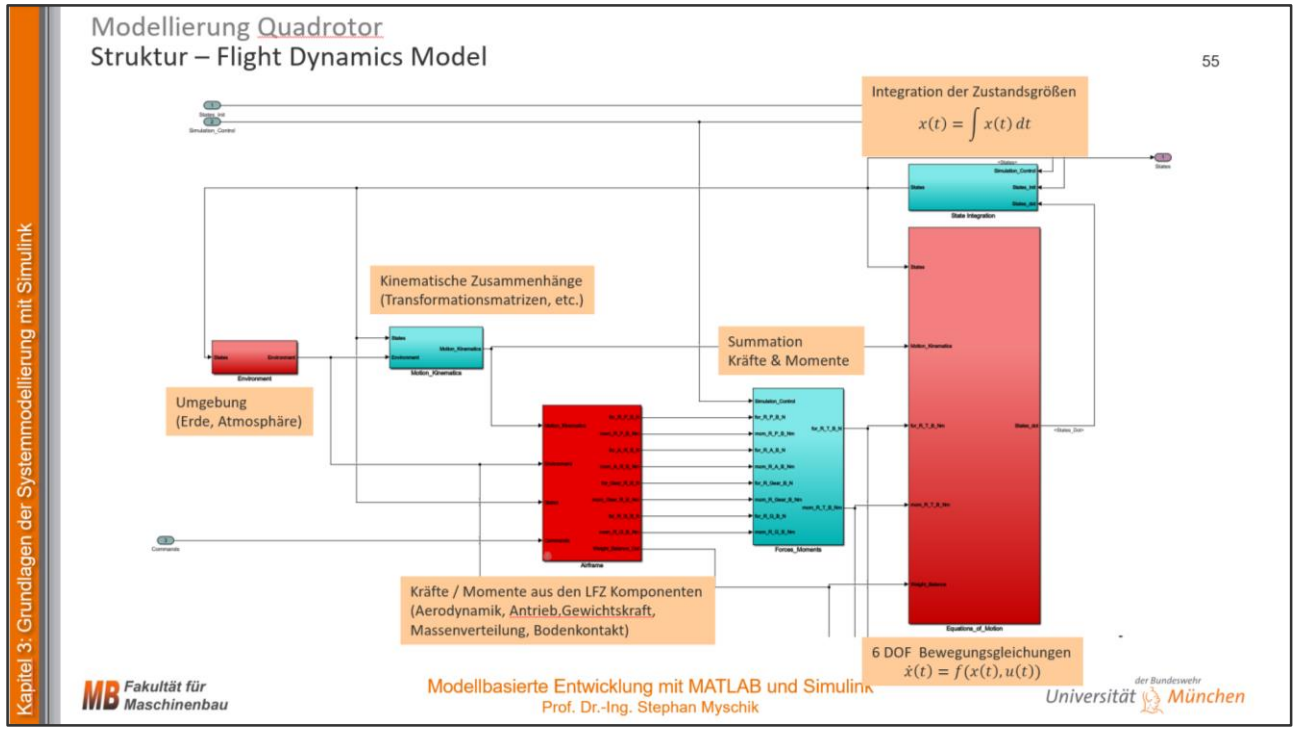


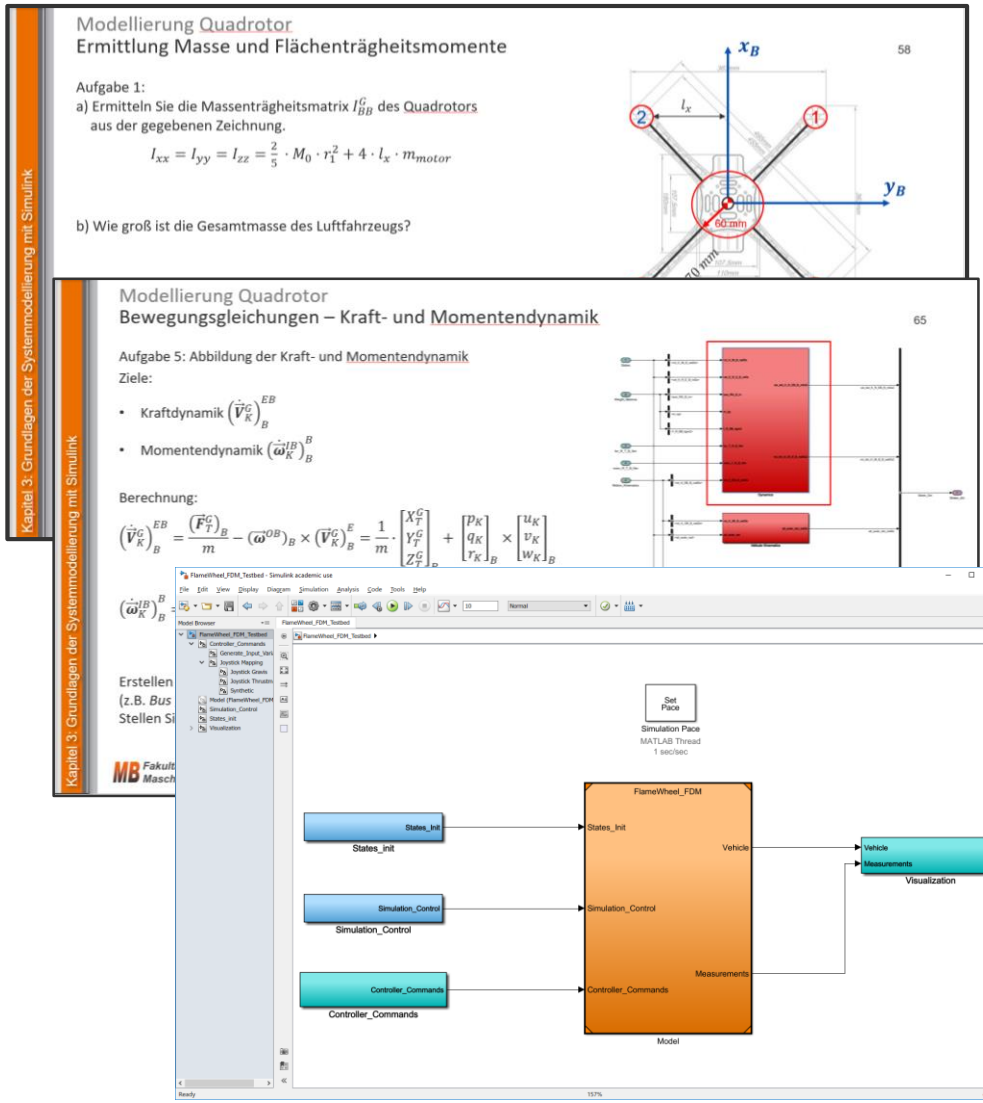


- Organisation der Aufgabenstellung mit Simulink Projects
- Erlaubt das Kennenlernen der Arbeit mit Versionskontrollsystemen (SVN, Git)
- Animation mittels FlightGear (Aerospace Blockset)

Grundlagen der Systemmodellierung mit Simulink

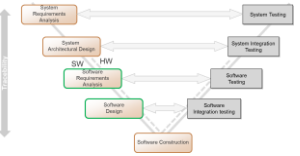




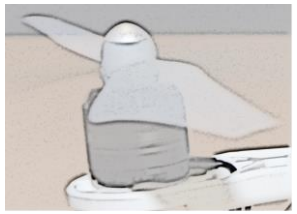


- Ausgangslage ist ein vordefiniertes Modellgrundgerüst mit fehlenden Komponenten und Initialisierungsdaten
- Aufgabe der Studenten ist es, die fehlenden Komponenten umzusetzen und relevante Parameter abzuschätzen
- Ergebnis ist die komplette Streckendynamik des Quadrocopters

Abgleich Modell mit der Realität

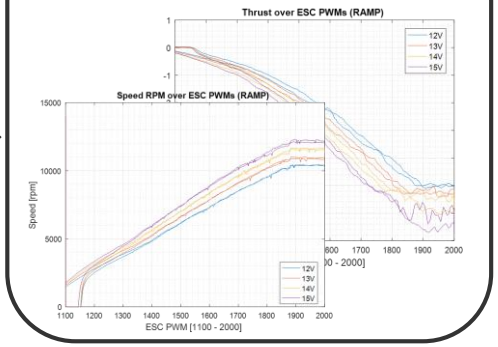


Analytisches Schubmodell

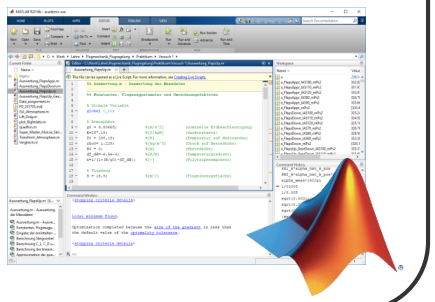


$$\left(\vec{F}_P^G\right)_B = f(\omega, V, C_L, S, \ldots)$$

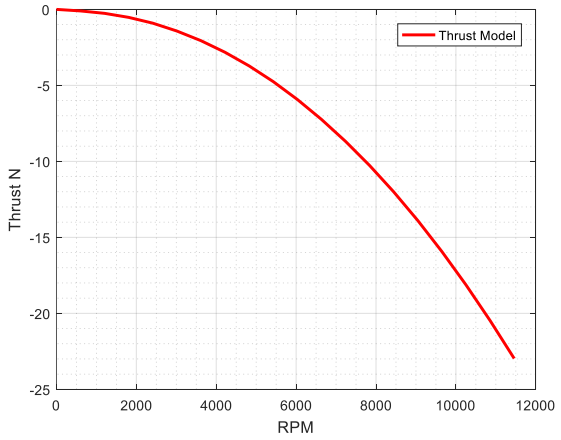
Schubmessung

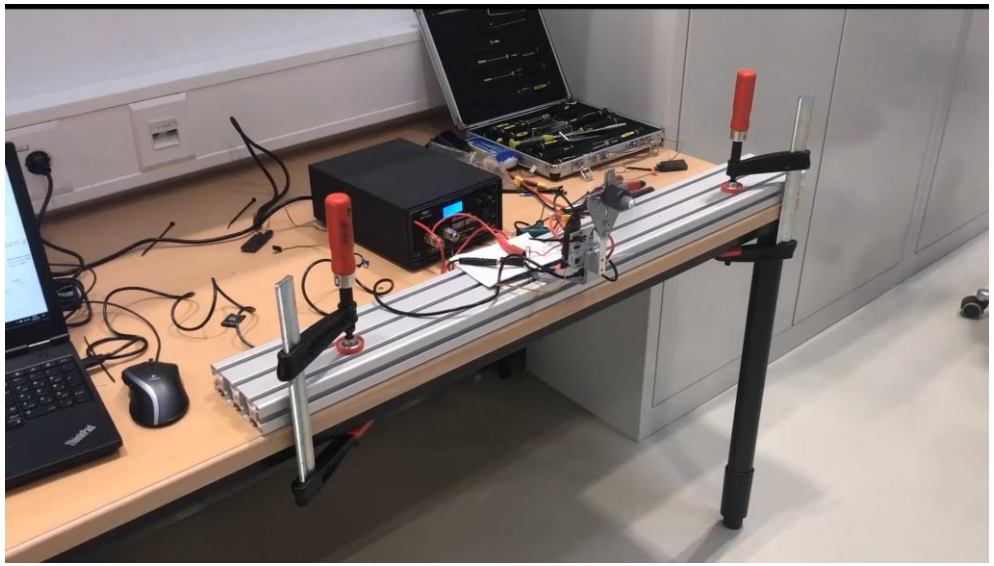


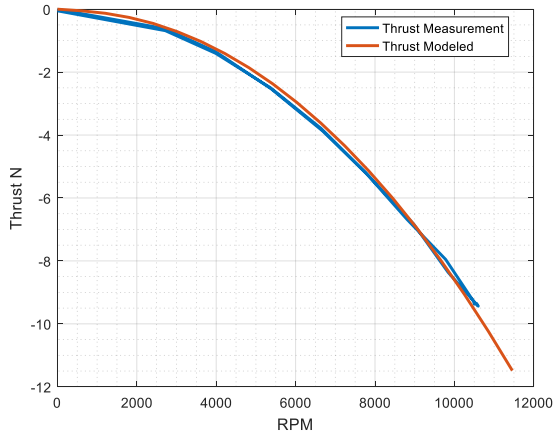
Curve Fitting



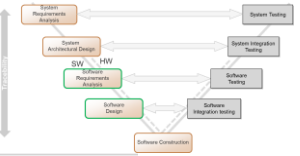
Reales Schubmodell







Analyse der Streckendynamik
Trimmen



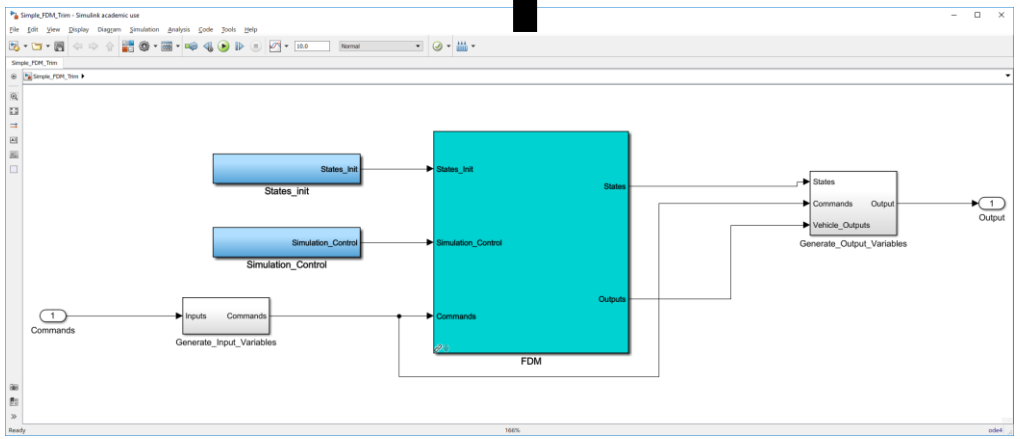
Trim_Template_Hover.m | Run_Trim.m

```
%% Run Trimming Algorithm
init_FDM

% Get trim definitions
[State, Input, Output, Param] = Trim_Template_Hover;

%[State, Input, Output, Param] = Trim_Template_Hover_RollPitchYaw(500);
%[State, Input, Output, Param] = Trim_Template_ForwardFlight(1,180*pi/180,10);

% Get operspecs
OpSpec = operspec('Simple_FDM_Trim_Start');

% Set operspecs to trim definitions
OpSpec = setOperspec(OpSpec, State, Input, Output);

%% Run trim algorithm
% Optimization options
opti_options = optimset;
opti_options.Algorithm = 'interior-point';
opti_options.Display = 'iter';
options = findopOptions('OptimizationOptions', opti_options);

% Find trim point
OP = findop('Simple_FDM_Trim_Start', OpSpec, options);
[State, Input] = getTrimStates(OP, State, Input);
```

findop

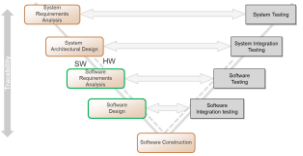
Command Window

```
      x:      -1.34e-15      dx:             0 (0)
(4.) pos_x_R_O_m
      x:              0      dx:             0
(5.) pos_y_R_O_m
      x:              0      dx:             0
(6.) pos_z_R_O_m
      x:             -5      dx:             0 (0)
(7.) rot_x_K_IB_B_radDs
      x:              0      dx:      1.45e-14 (0)
(8.) rot_y_K_IB_B_radDs
      x:              0      dx:     -4.82e-15 (0)
(9.) rot_z_K_IB_B_radDs
      x:              0      dx:     -1.19e-15 (0)
(10.) vel_x_K_R_E_B_mDs
      x:              0      dx:      7.02e-16 (0)
(11.) vel_y_K_R_E_B_mDs
      x:              0      dx:      -1.8e-15 (0)
(12.) vel_z_K_R_E_B_mDs
      x:              0      dx:     -6.39e-12 (0)

Inputs:
----------
(1.) Simple_FDM_Trim_Start/Commands
      u:            481    [-Inf Inf]
      u:            481    [-Inf Inf]
      u:            481    [-Inf Inf]
      u:            481    [-Inf Inf]

Outputs:
----------
(1.) Simple_FDM_Trim_Start/Output
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:     -8.88e-16     [-Inf Inf]
      y:     -1.78e-15     [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:              0    [-Inf Inf]
      y:             -5    [-Inf Inf]
```

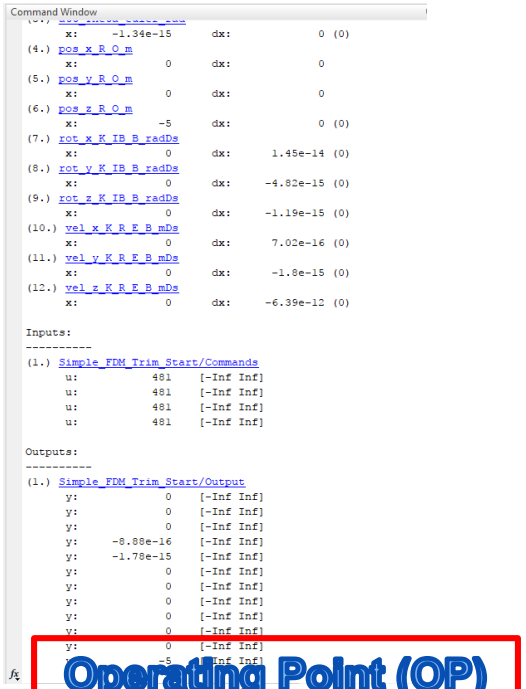


Operating Point (OP)

Linearisierung



Trimmpunkt



Linearisierung (`slLinearizer`)

$$A(:,i) = \frac{\dot{x}|_{x_{p,i}} - \dot{x}_0}{x_{p,i} - x_0},$$

$$B(:,i) = \frac{\dot{x}|_{u_{p,i}} - \dot{x}_0}{u_{p,i} - u_0}$$

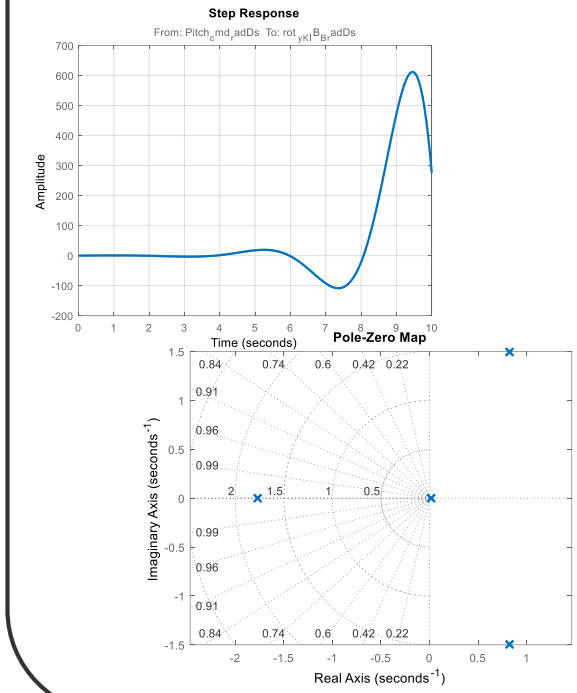
$$C(:,i) = \frac{y|_{x_{p,i}} - y_0}{x_{p,i} - x_0},$$

$$D(:,i) = \frac{y|_{u_{p,i}} - y_0}{u_{p,i} - u_0}$$

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

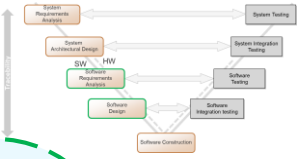
$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathbf{D} \cdot \mathbf{u}$$

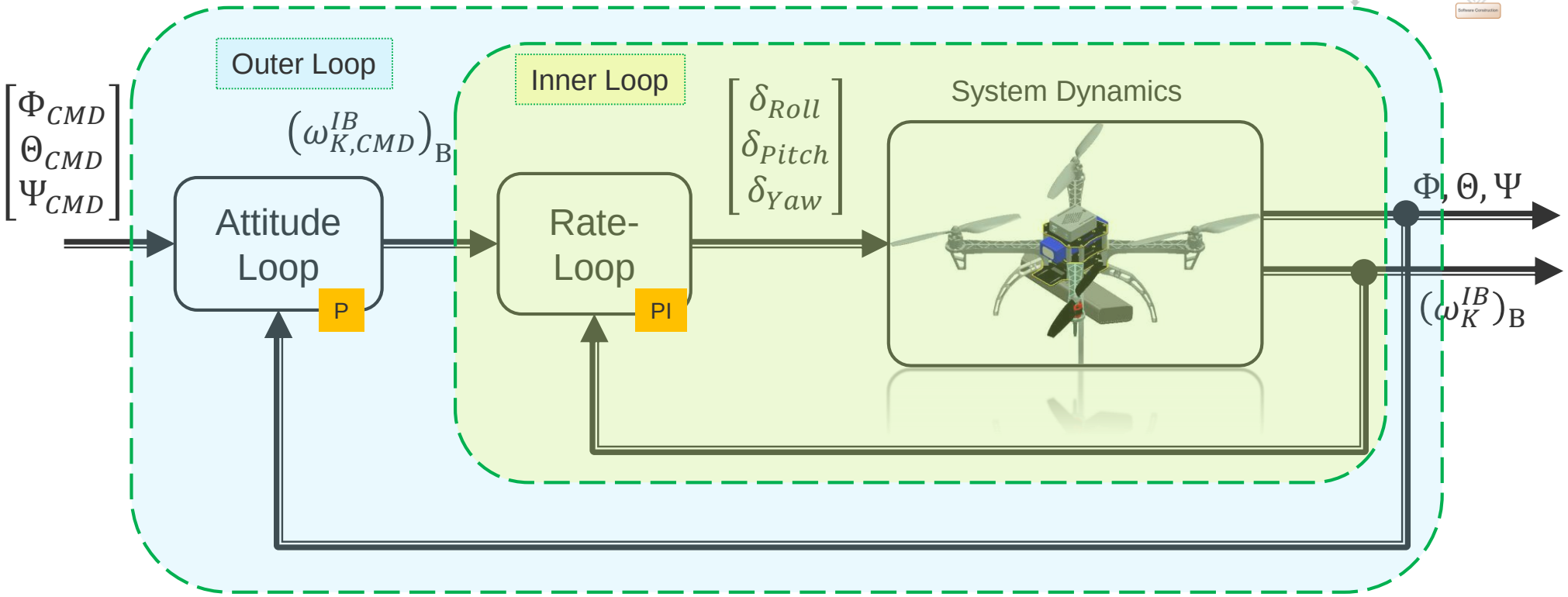
Analyse der Dynamik (`step,initial,eig,tf,…`)



- Anwendung der typischen LTI Analyseverfahren aus der Control System Toolbox und Simulink Control Design

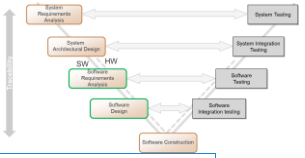
Reglerentwurf

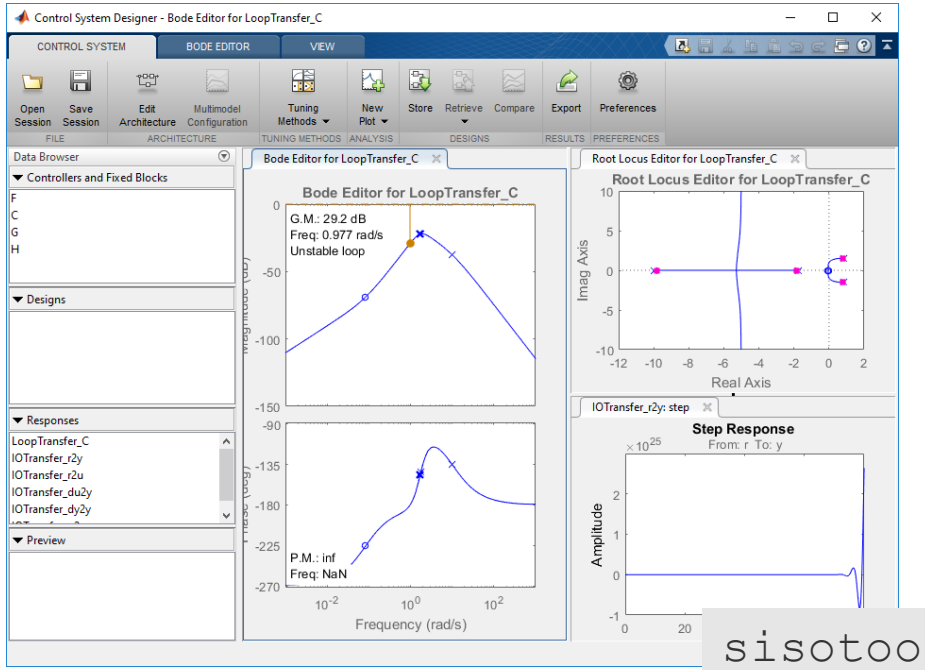




- Typische Kaskadenregelung zur Stabilisierung der Eigendynamik und Realisierung einer Lagekontrolle
- Entwurf des Reglers erfolgt zunächst in MATLAB unter Verwendung des **linearen Zustandsraummodells bzw. abgeleiteter Übertragungsfunktionen**
  (Control System Toolbox: `tf, feedback, series, …`*)*
- Im Anschluss dann Umsetzung in Simulink und Evaluation in nichtlinearen Simulationen

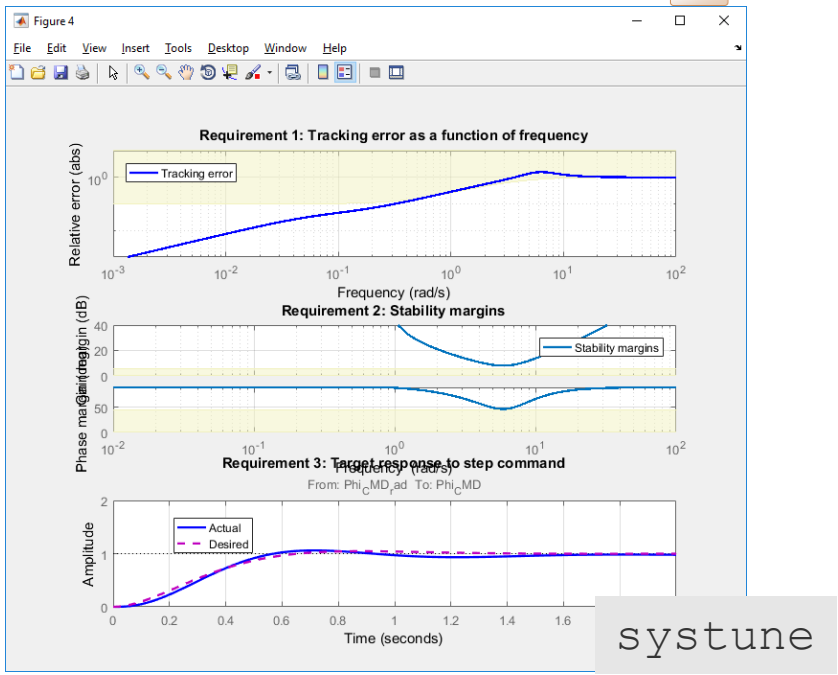
Reglerentwurf und Parametrierung in MATLAB





sisotool

- Grafische Oberfläche zur Parametrierung von SISO Reglern
- Wichtigen Kenngrößen auf einem Blick („Bode Diagramm“, „Wurzelortskurve“,“Sprungantwort“)
- Einfache Veränderung der Parameter mittels Maus/Tastatur
- Anpassung der Strecken- / Reglerdynamik durch Hinzufügen/Entfernen von Polen- und Nullstellen
- Automatisierte Auslegungsverfahren zur Bestimmung von Reglergrößen durch Optimierung, etc.

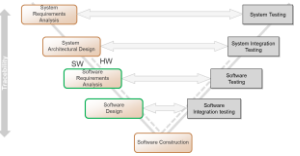


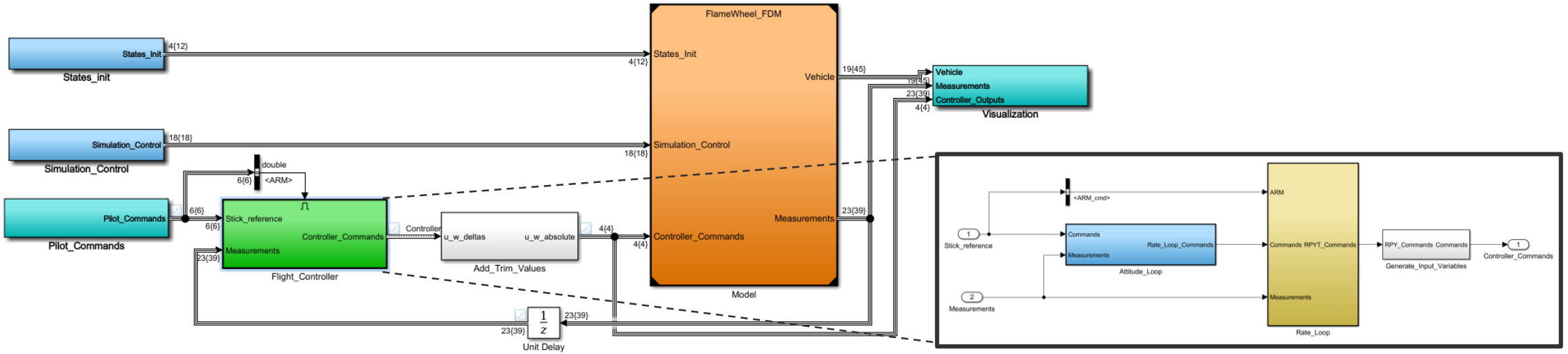
systune

- Automatisierte Ermittlung von Reglerparametern mittels Optimierungsverfahren
- Vorgabe und Gewichtung von Anforderungen (Zeitverhalten, Amplituden-/Phasenreserve, Störverhalten, etc.)

⇨ Studenten ermitteln durch beide Verfahren die Parametrierung der Regelschleifen

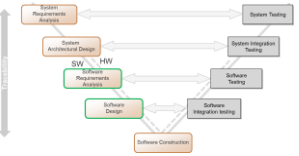
FCS Design Modell

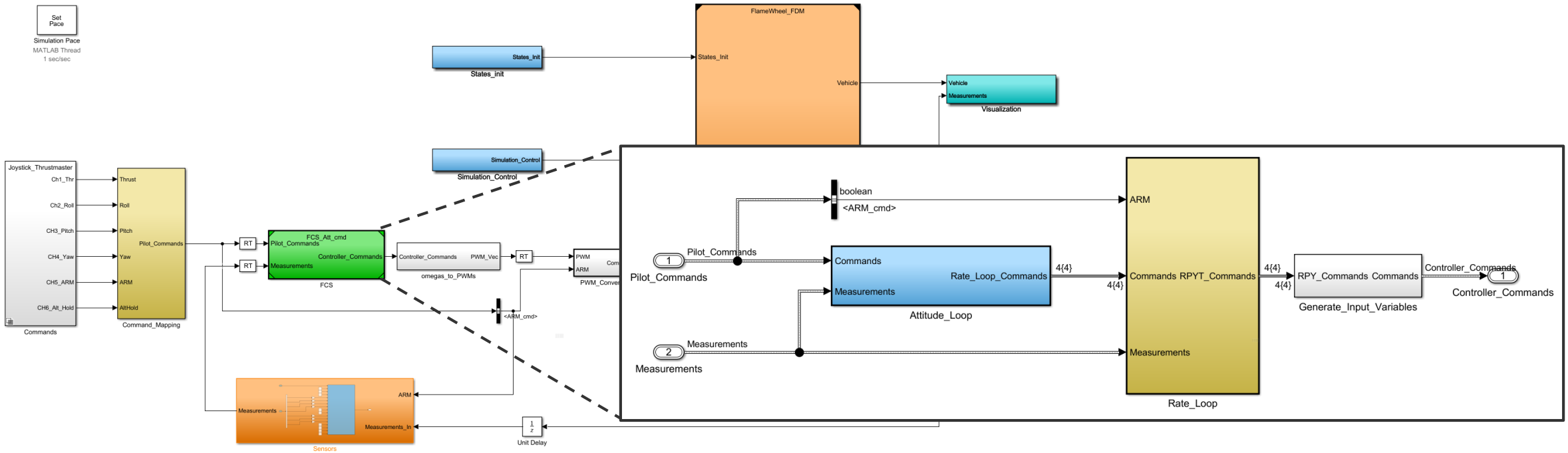




- Umsetzung der Reglerstruktur in Simulink durch die Studenten
- Parametrierung mit den Ergebnissen des linearen Entwurfsschritts
- Evaluation der Performance bei Berücksichtigung nichtlinearer Effekte (Aktuatorbegrenzungen, etc..)
- Schnelle Design Iterationen notwendig
  - daher noch keine Festlegung des Interfaces
  - Parameter als normale MATLAB Workspace Variablen - `double` Datentypen

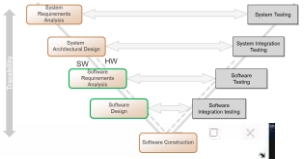
Nichtlineare Gesamtsystemsimulation

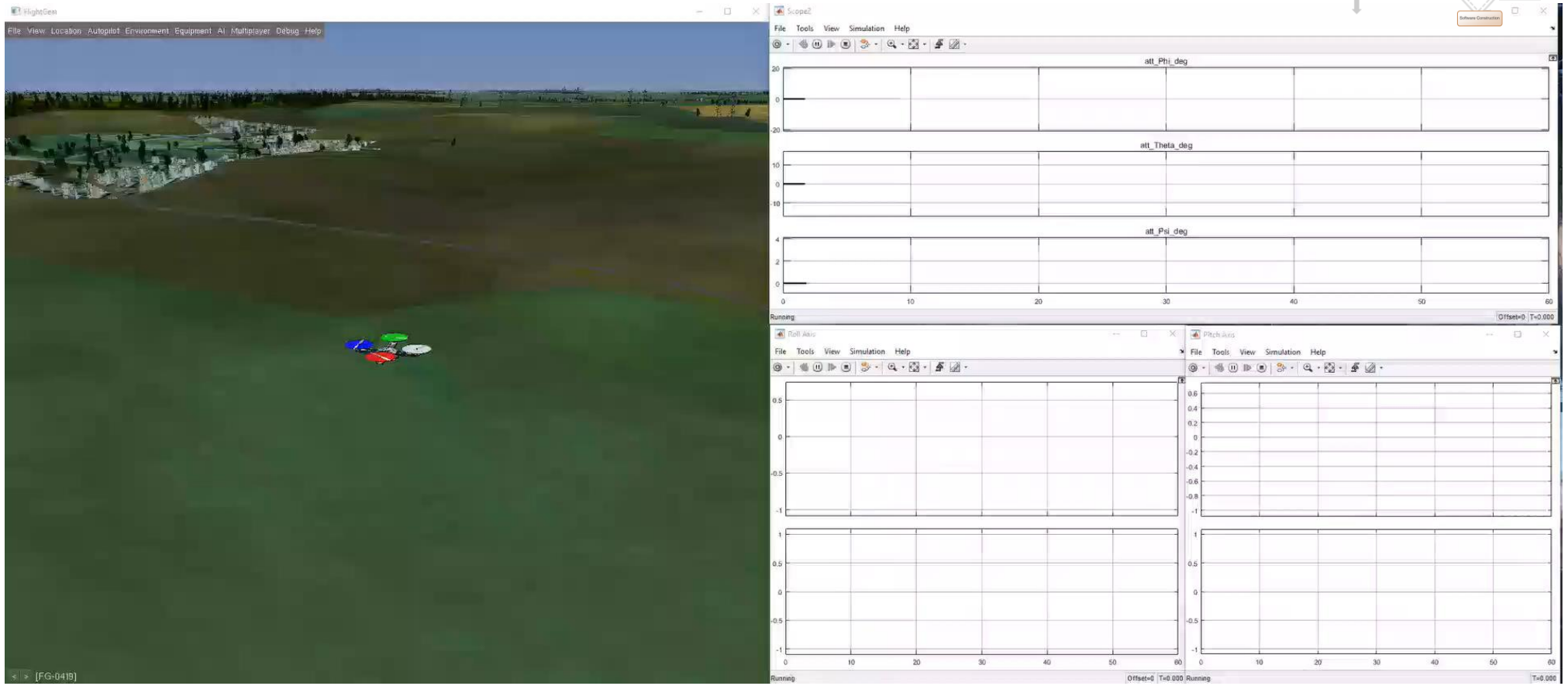




- Erhöhung der Implementierungsnähe des Flugreglers basierend auf dem Design-Modell (`single` Datentypen, etc.)
- Festlegung der Schnittstellen der Softwarekomponente durch Bus-Objekte in Kombination mit Model-Reference Block
- Berücksichtigung unterschiedlicher Abtastraten und Solvertypen zwischen Modell (1 kHz, ode4) und Regler (250 Hz, discrete)
- Verwendung von Configuration References zur Festlegung der Modellkonfiguration
- Umsetzung Reglerparameter als `Simulink.Parameter` Objekte zur Beeinflussung der Codegenerierung

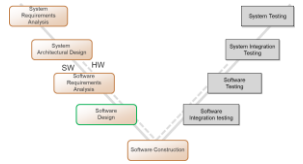
Nichtlineare Gesamtsystemsimulation

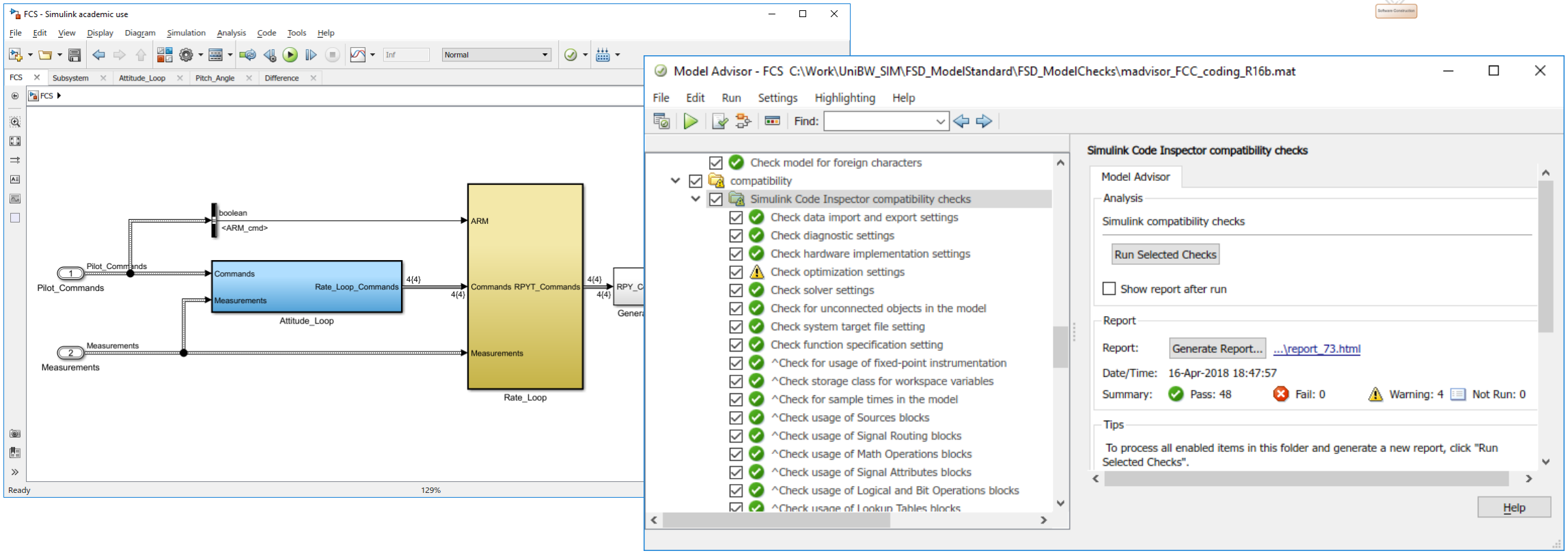




- Implementierungsnahe Gesamtsystemsimulation zur Evaluation von Effekten, die aus der Hardware resultieren. (Sensordatenvorverarbeitung, PWM-Schnittstelle, Datentypen, …)
- Fähigkeit zur Durchführung von Software- und Processor-In-The-Loop Simulationen

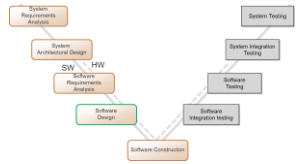
Statische Modellanalyse

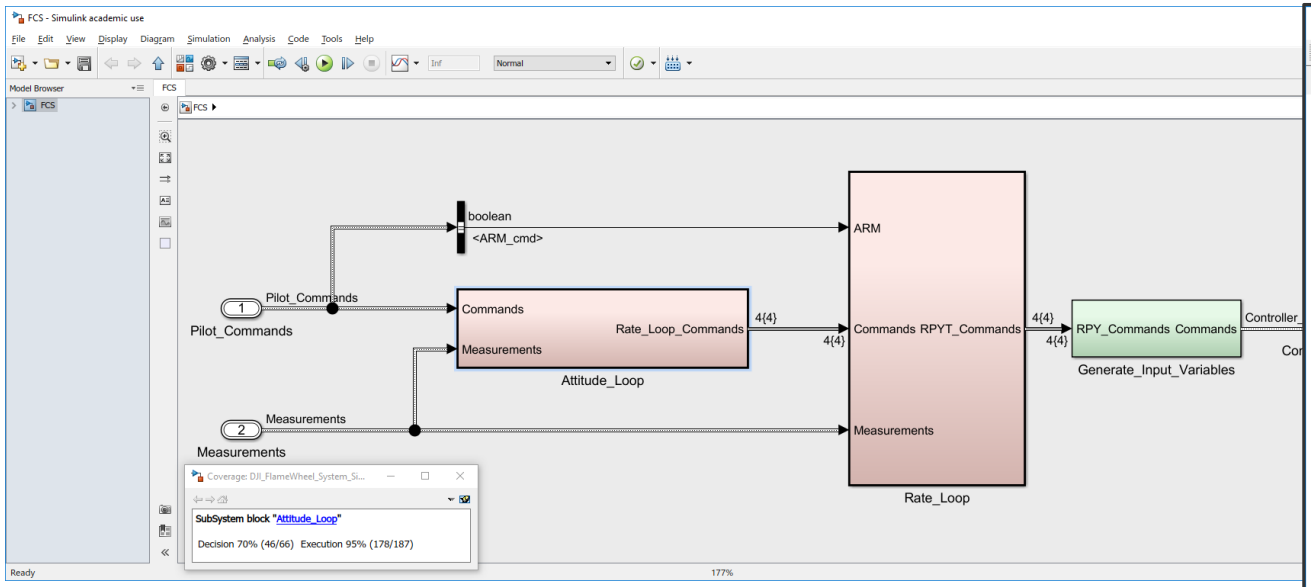


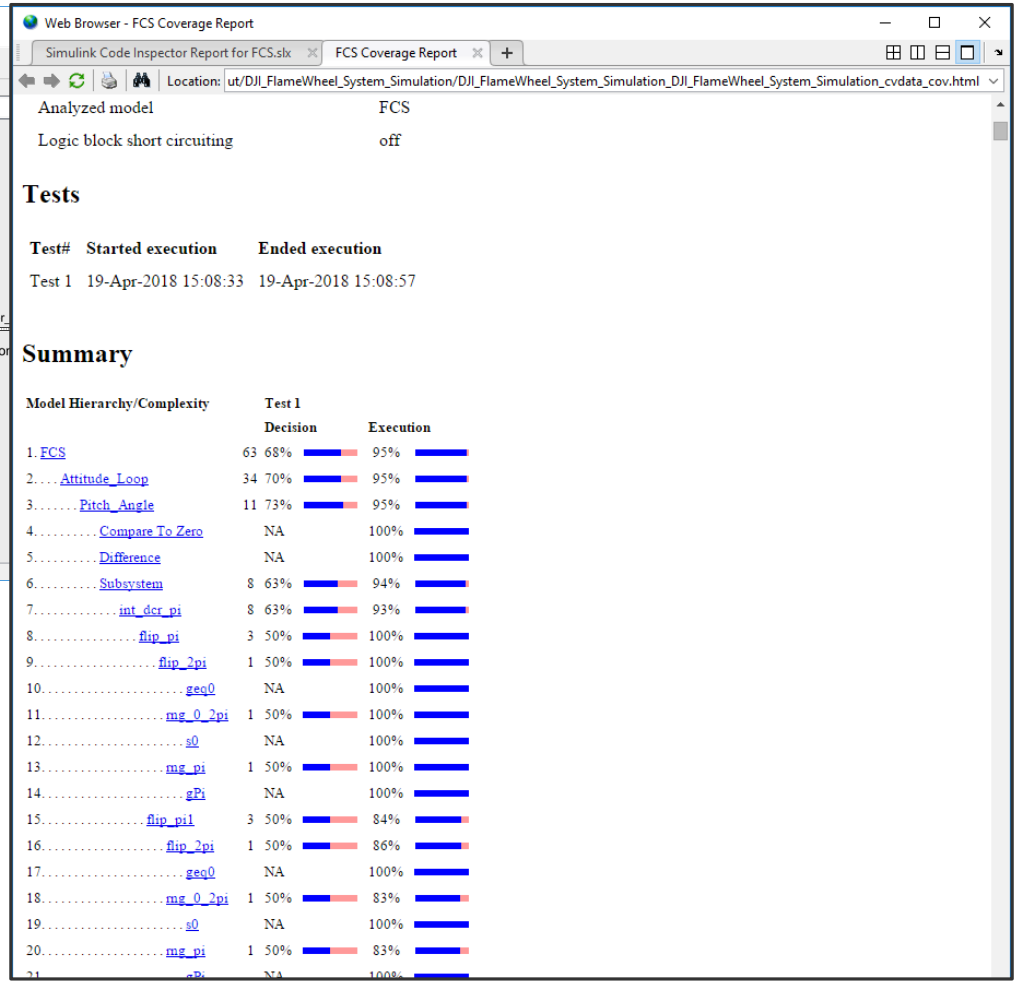


- Überprüfung des Modells auf Standardkonformität vor der Implementierung
- Dient als ein Beispiel dafür, wie durch frühe Verifikation auf Modellebene mögliche Fehler entdeckt werden können

Modellabdeckung

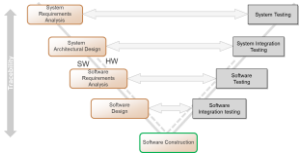


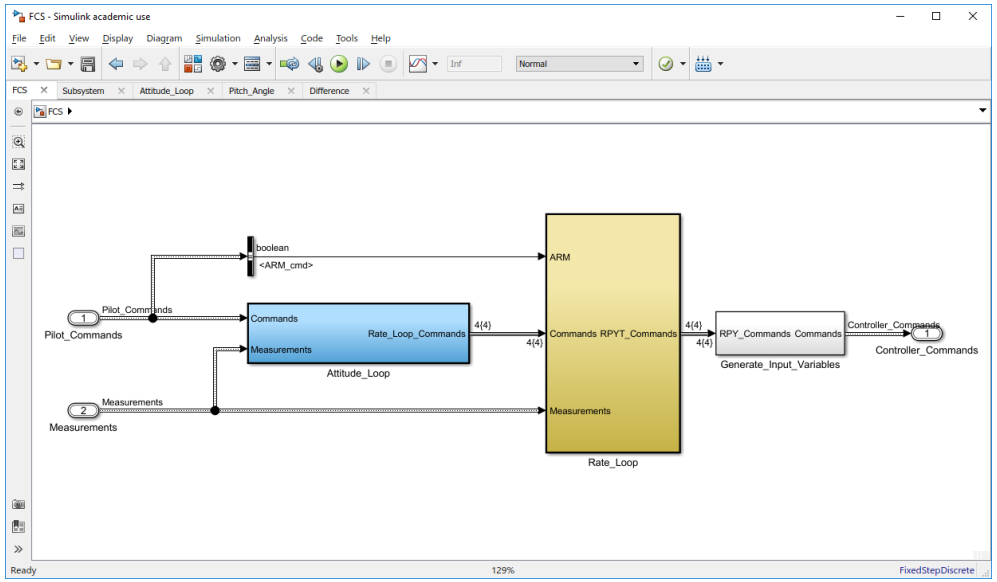




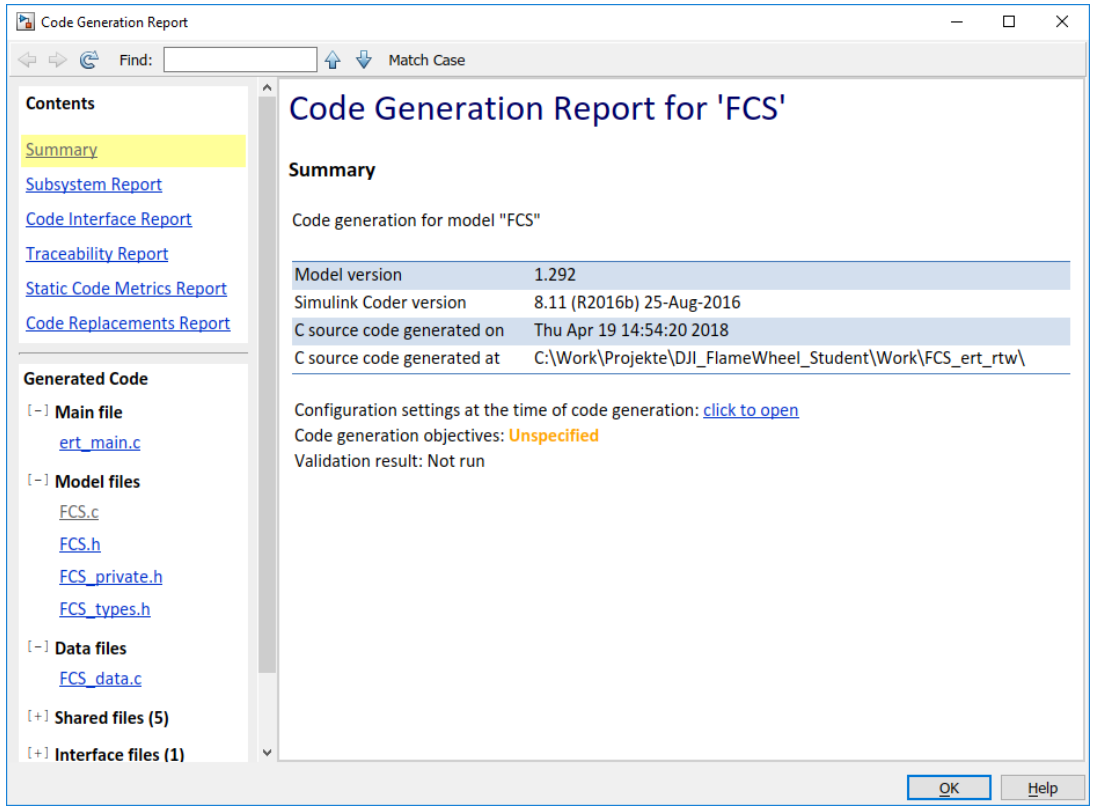
- Maß dafür, wie gut der Algorithmus schon in der Simulation durch Testfälle getestet wird
- Anforderung in vielen Standards (DO 178, …)
- Studenten lernen den Zusammenhang zwischen Requirements, Testfällen und Modellabdeckung

Automatische Codegenerierung



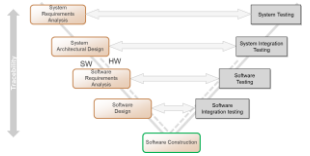


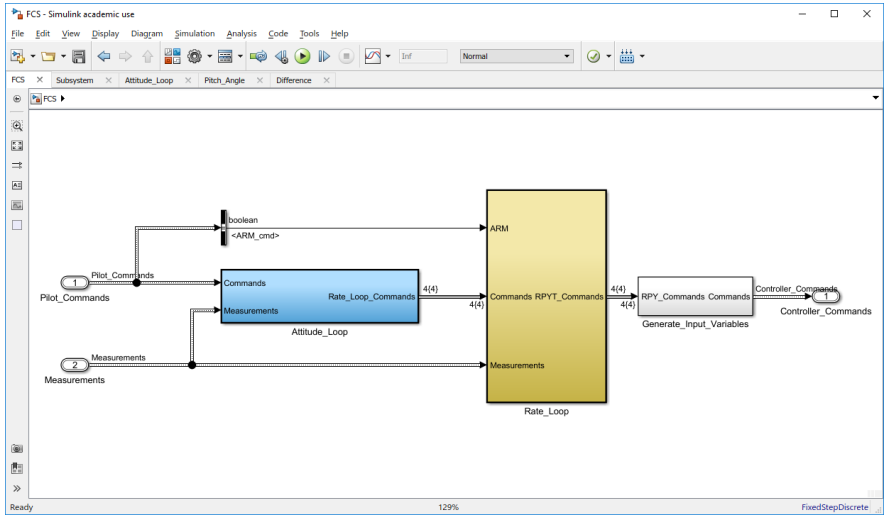
EC

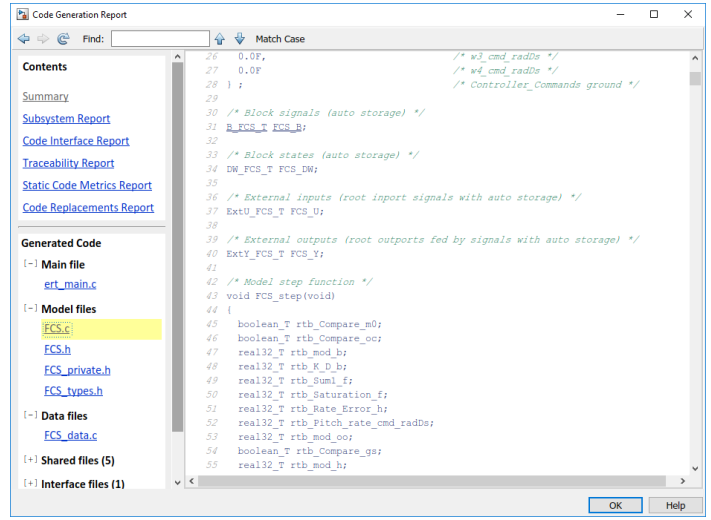


- Automatische Codegenerierung mittels Embedded Coder
- Methoden zur Beeinflussung des generierten Codes durch Simulink Parameter, etc. wird demonstriert.

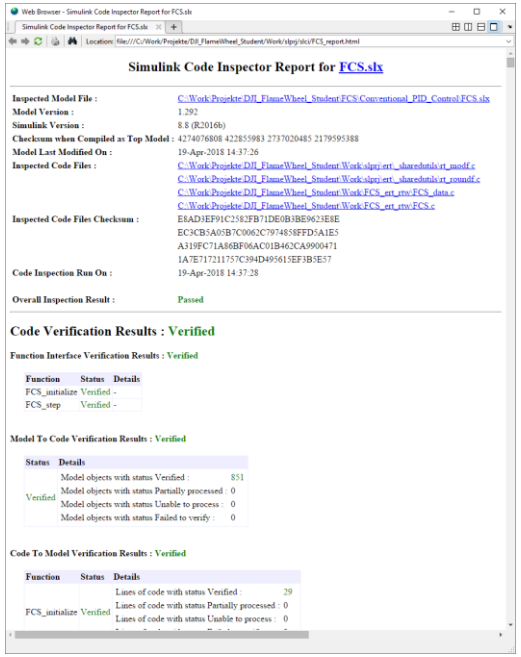
Model-Based Design mit MATLAB/Simulink
Verifikation auf Codeebene







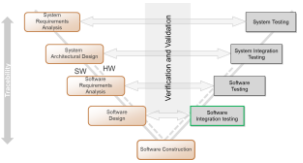
SLCI

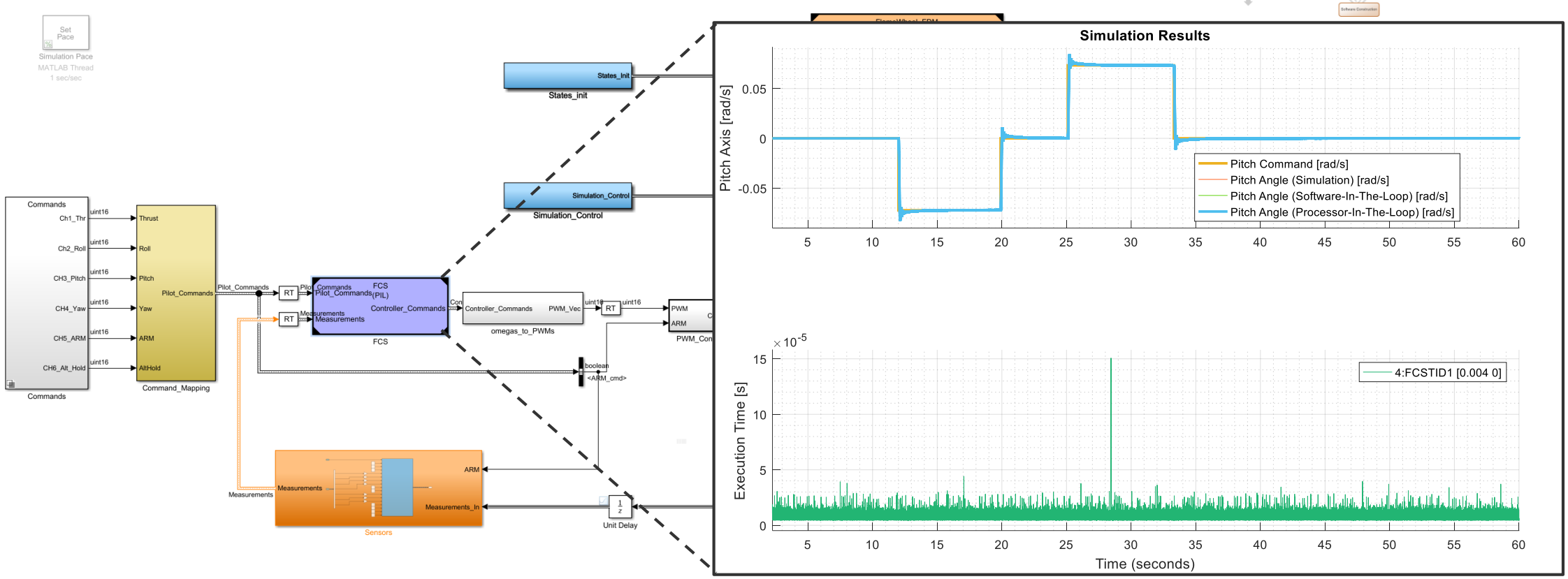


SLCI

- Nachweis struktureller Äquivalenz zwischen Modell und Code
- Beispiel für die Erfüllung einer Anforderung nach DO 178 C

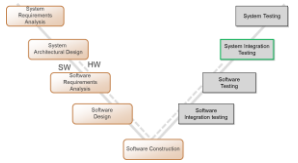
Processor-In-The-Loop Simulation

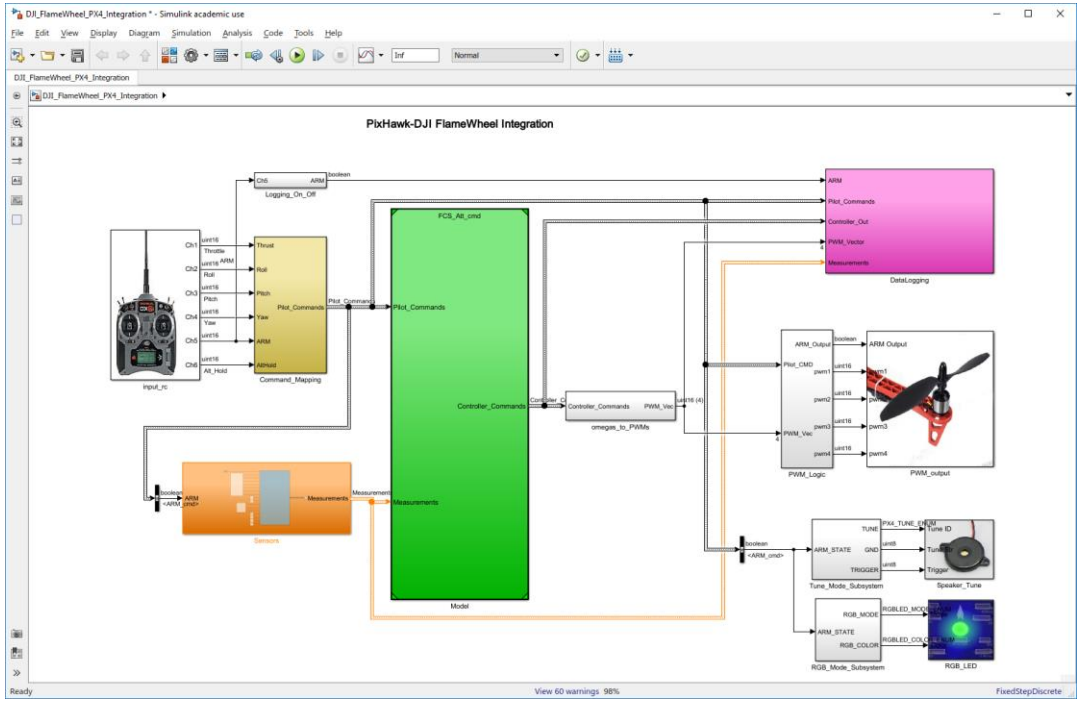


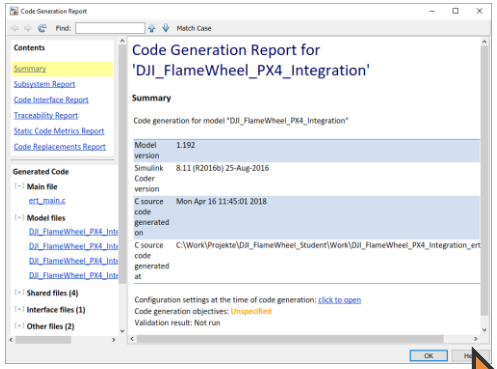


- Nachweis funktionaler Äquivalenz zwischen generiertem Code und Modell
- Wiederverwendung von Testfällen (Frontloading) und Laufzeitmessung

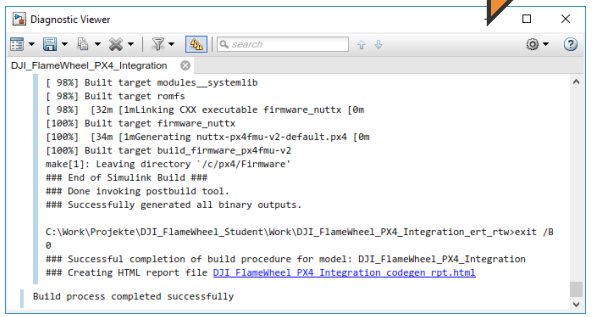
System Integration





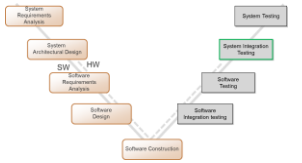


Embedded Coder

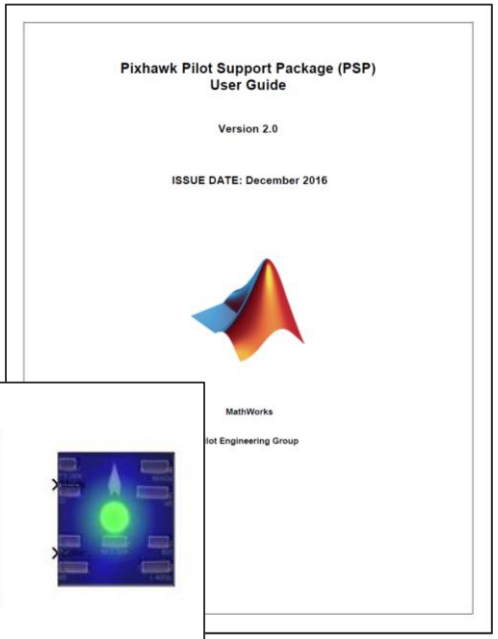


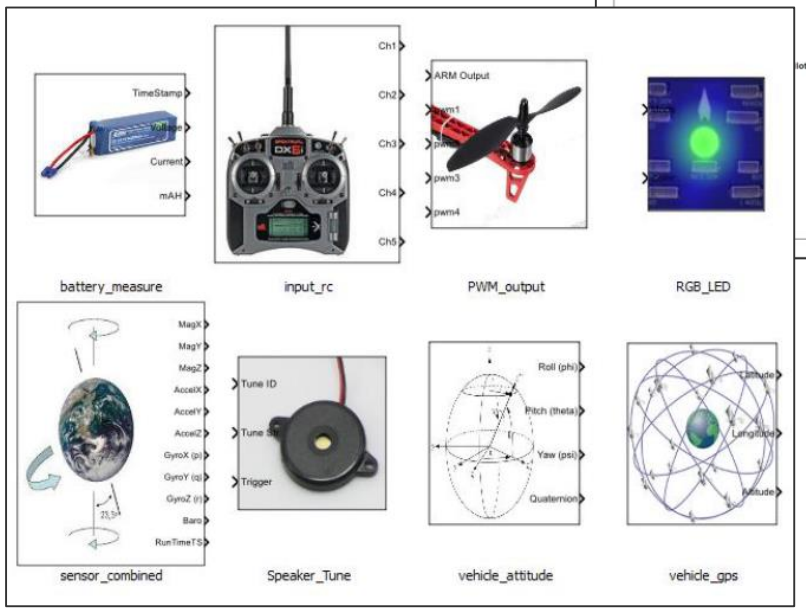
- Integrationsmodell kombiniert Flugregler mit Interfaceblöcken zur Applikation auf der Zielhardware
- Verwendung der gleichen Modellkomponente wie in der nichtlinearen Systemsimulation durch „Model Reference“
- Automatische Codegenerierung mittels Embedded Coder und Pilot Support Package

PixHawk Pilot Support Package (PSP)



- Library Blöcke zur Integration der Algorithmen auf der Hardware
- Toolchain Support („Compile/Link/Download“)
- „External Mode“ – Unterstützung
- Keine zusätzlichen Integrationsschritte





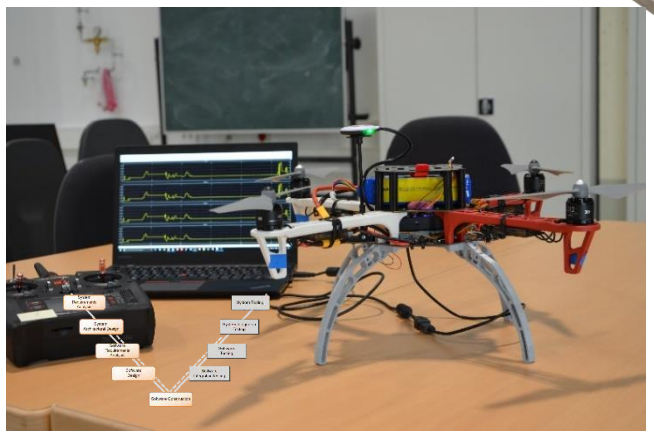
- Fokus auf der Lösung des technischen Problems
- Schnelle Integration und Test verschiedener Entwürfe

Ergebnis

Ergebnis

Zusammenfassung

- Vielseitiger Einsatz von MATLAB und Simulink im Rahmen der Vorlesungen des Studiengangs „Aeronautical Engineering“
- Hoher Praxis- und Anwendungsbezug bei gleichzeitigem fachlichen Tiefgang durch Integration von Simulatoren und fliegender Hardware
- Die Verwendung von MATLAB und Simulink erlaubt es den Studenten, sich auf die Lösung des Problems zu fokussieren
- Hohes Motivationslevel der Studenten durch wirklich fliegende Dinge.
- Einarbeitungsaufwand lohnt sich!



Herzlichen Dank für Ihre Aufmerksamkeit!

stephan.myschik@unibw.de