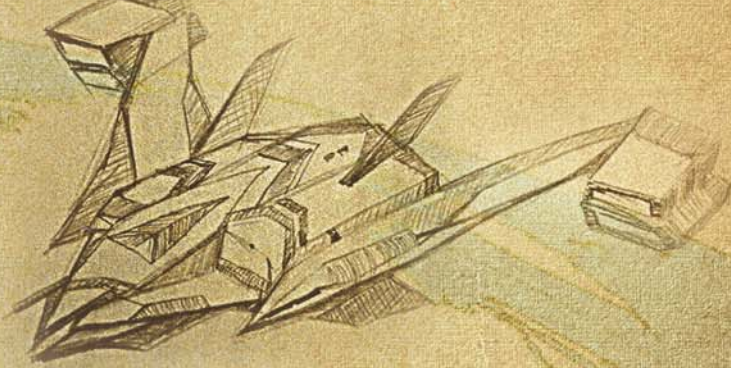
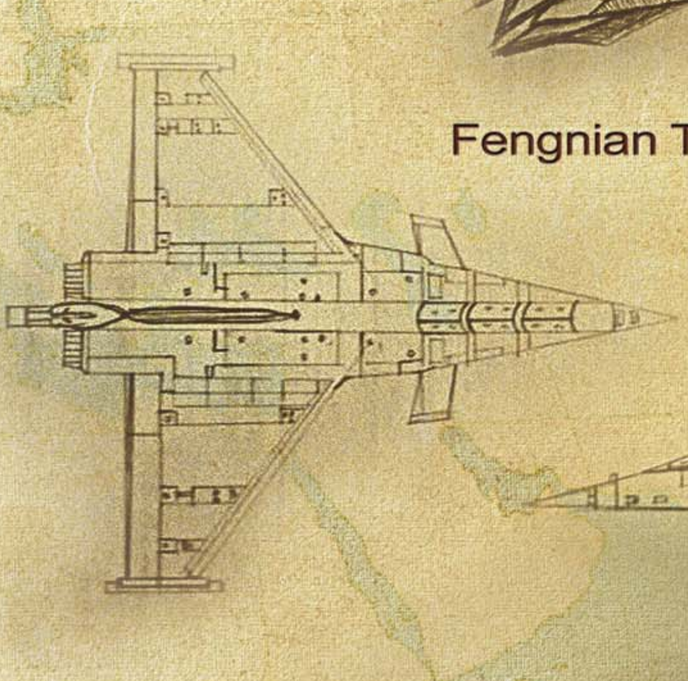
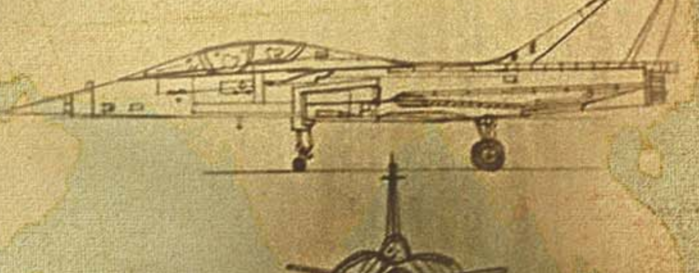
An Integrated Knowledge Based Engineering Mechatronics Modeling Approach to Support the Design of Unstable and Unmanned Aircraft



Fengnian Tian





Propositions

accompanying the dissertation

An Integrated Knowledge Based Engineering Mechatronics Modeling Approach to Support the Design of Unstable and Unmanned Aircraft

by

Fengnian Tian

1. Within a multidisciplinary design optimization framework, it is essential to have a common design representation, suitable for all disciplines involved, to ensure a consistent final result (chapter 3).
2. The common design representation should capture the intrinsic properties of physical systems but must avoid a specific format of representation, such as e.g. dedicated geometry files or schematics of control systems (chapter 3).
3. It is essential to select the right level of fidelity for the analyses in a multidisciplinary design optimization framework, such that the accuracy is suitable for the design process and the required system information in the common design representation is minimal (chapter 3).
4. It is possible to concurrently design the mechatronic systems using advanced design methods but disciplinary analyses must still be performed sequentially (chapter 3).
5. Traditional design methods are largely limited to the initial solution space. KBE systems on the other hand can be used to continuously explore the solution space and act as an evolutionary design system, only limited by the computational speed of the computer (chapter 6).
6. The KBE technique can effectively capture design rules and repetitive activities and support the design process but for future KBE system, it is essential to have human creativity included.
7. If in the creation process of a KBE application, its maintenance is not practically addressed, it will gradually lead to a dead application.
8. Truly novel and innovative research can be hindered by the current mechanism for publishing in peer reviewed journals because there may not be suitable journals and reviewers for the topic.
9. Ideally one should go to high school in China and attend university in Europe.
10. Patience is a word of pain and ambition.

These propositions are regarded as opposable and defendable, and have been approved as such by the supervisor Prof. dr. ir. L.L.M. Veldhuis.

Stellingen

behorende bij het proefschrift

AN INTEGRATED KNOWLEDGE BASED ENGINEERING MECHATRONICS MODELING APPROACH TO SUPPORT THE DESIGN OF UNSTABLE AND UNMANNED AIRCRAFT

door

Fengnian TIAN

1. Het is essentieel om binnen een multidisciplinair ontwerp optimalisatie systeem een gemeenschappelijke beschrijving van het ontwerp te hebben, voor alle betrokken disciplines, om zeker van een consistent eind resultaat te zijn.
2. De gemeenschappelijke beschrijving van het ontwerp moet de intrinsieke karakteristieken van fysische systemen vastleggen. Een specifieke formulering zoals gebruikelijk voor de beschrijving van geometrie of besturing systemen moet echter voorkomen worden.
3. Het is essentieel om de juiste nauwkeurigheid te selecteren voor de analyses binnen een multidisciplinaire ontwerp optimalisatie zodat de nauwkeurigheid voldoende is voor het ontwerp proces en de benodigde hoeveelheid informatie in de gemeenschappelijke beschrijving van het ontwerp minimaal is.
4. Het is mogelijk om de elementen van een mechatronisch systeem gelijktijdig te ontwerpen met geavanceerde ontwerp methodieken. Echter, de analyses moeten nog steeds sequentieel uitgevoerd worden.
5. Traditionele ontwerp methodes zijn grotendeels beperkt tot de initiële oplossingsruimte. KBE systemen kunnen echter continu de oplossingsruimte verkennen en zich als evolutionair ontwerp systeem gedragen. De enige beperking is de beschikbare rekentijd en rekenkracht van de computer
6. De KBE benadering is effectief in het vastleggen van ontwerp regels en activiteiten die veelvuldig herhaald worden, en kan daarmee ontwerp processen ondersteunen. Voor toekomstige KBE systemen is het essentieel dat ook menselijke creativiteit toegevoegd wordt.
7. Als in de ontwikkeling van een KBE applicatie, het onderhoud en doorontwikkeling van deze applicatie niet is overwogen dan leidt dit uiteindelijk tot een niet werkende applicatie.
8. Werkelijk nieuw en innovatief onderzoek kan gehinderd worden door het huidige publicatie proces in wetenschappelijke tijdschriften omdat er mogelijk geen geschikte tijdschriften voor publicatie zijn of vakgenoten die het werk kunnen beoordelen.
9. In het ideale geval gaat men naar de middelbare school in China en vervolgens naar een universiteit in Europa.
10. Geduld bestaat uit pijn en ambitie

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotor Prof. dr. ir. L.L.M. Veldhuis.

An Integrated Knowledge Based Engineering Mechatronics Modeling Approach to Support the Design of Unstable and Unmanned Aircraft

An Integrated Knowledge Based Engineering Mechatronics Modeling Approach to Support the Design of Unstable and Unmanned Aircraft

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K. C. A. M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
donderdag 24 september 2015 om 10:00 uur

door

Fengnian Tian

Master of Vehicle Engineering
Nanjing Agricultural University, China
geboren te Nei Mongol, China

Keywords: Multiphysics simulation model; Electronic control system; Knowledge based engineering; Unmanned aerial vehicle; Multidisciplinary design optimization

Printed by: Ipskamp Drukkers, Enschede, The Netherlands
Cover by: Chunzhe Li
ISBN/EAN: 978-94-6186-528-1
An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

To my parents,

Zengshun Tian and Cuiping Cui,

Without whom none of my success would be possible

SUMMARY

The commercial transport aircraft industry is currently developing new “more electric aircraft” (MEA) designs in which various conventional mechanical, hydraulic and pneumatic power systems are replaced with electrically-based power systems. Their objective is to improve the overall flight performance by reducing the aircraft weight and by a lower overall energy requirement for the systems. The vision for the future is to ultimately replace all systems with electrical systems and even to replace a part of the fuel used as primary source of energy for the propulsion system by an electrical power supply and thereby to achieve either a hybrid electric aircraft (HEA) or even all electric aircraft (AEA) if permitted by future developments in battery technology. In recent years, many small scale electric aircraft were developed to demonstrate the AEA concept. It has been determined that although the MEA, HEA and AEA concepts reduce the overall complexity of the aircraft, it significantly increases the complexity of electrical and electronic systems (E/E systems) and their integration into the aircraft, introducing a new challenge for the aircraft design industry.

Two specific categories of aircraft, currently in operation, face the same challenge. These categories are; (1) unmanned aerial vehicles (UAVs), which by nature have more electrical and electronic systems (E/E systems) on-board and require an automatic flight control system due to the absence of a pilot and (2) aircraft which are inherently unstable and therefore require automatic flight control systems for stabilization. These two aircraft categories can be classified as typical mechatronic products.

E/E systems have a significant impact on the overall flight performance, directly determine the flying qualities of aircraft, and are critical for safety. Thus, these systems should be developed synchronously with the other traditional engineering domains such as aerodynamics, structures and propulsion. However, several challenges need to be overcome before this can be achieved effectively. Three specific challenges are identified and addressed in the current research study:

- The development of high fidelity multiphysics simulation models for analysis and development of the E/E systems is a complex, time consuming and multidisciplinary task that requires a large amount of manual work from simulation experts;

- The design of consistent automatic flight control systems for use throughout the entire flight envelope and for all aircraft weight and c.g. combinations is labor intensive and requires the availability of high fidelity multiphysics simulation models in the early design phases;
- The development of control software components is prone to errors due to inconsistencies between the description of the top level physical configuration, the control architecture and the associated software components.

Traditional aircraft design methods which are largely dominated by the mechanical engineering domains are not suitable to synchronously design complex integrated E/E systems. Moreover, the conventional design process, which is sequential to a large extent, cannot support concurrent engineering requirements. Therefore, novel methods and tools to support the development of the E/E systems on-board aircraft are needed.

The overall objective of this research study is to reduce the development time of aircraft with a high level of integrated E/E systems by automating the design process of the flight control systems, by creating more consistent control software through the entire design envelope. Besides a reduction in development time, this will also improve the quality of the final (mechatronic) product. The three challenges described above will be tackled in particular.

The novel methods and tools are based on the knowledge based engineering (KBE) approach. The KBE approach is highly suitable because it cannot only automate non-creative, repetitive design tasks done for example by simulation experts but also support for multidisciplinary design, analysis and optimization (MDAO). Compared to other existing KBE systems, the proposed system integrates the flight control system design with the physical design in three specific areas.

First, in order to ensure a consistent design representation, the concept of a multiphysics information model (MIM) is proposed in order to integrate the design knowledge present from multiple engineering domains. The proposed MIM (a KBE system) defines objects with attributes to represent various aspects of physical entities (e.g., mass, inertia, geometry, material properties). Moreover, it uses functions to capture non-physical information, such as the control architecture, relevant test maneuvers, simulation procedures, etc. The problem of system coupling and interactions between disciplines involved are taken into account by the proposed KBE system in a knowledge acquisition process. Next, depending on the requirements, the proposed KBE system extracts necessary knowledge from the MIM which is needed for the development of a multiphysics simulation model, which is composed of a physical plant, flight control systems including the embedded control software and simulation configurations. By capturing the expertise of simulation experts, the proposed KBE system is able to automatically instantiate the multiphysics simulation models. This multiphysics simulation model can be used to evaluate the flight control systems in operation practice throughout the flight envelope, for example when performing maneuvers. Altogether, the MIM enables rapid development of high fidelity multiphysics simulation models for analysis and development of the flight control systems.

Second, in order to evaluate the inherent flying qualities of unstable aircraft in a simulation environment, an automatic flight control system is required. For this purpose, model based inversion control is applied. This method has the advantage that tuning is not required. The techniques, processes and knowledge required to develop a model based control system based on the (nonlinear) multiphysics simulation model are captured by the KBE system. Model based inversion control has its disadvantages when implemented on real-life aircraft. For the final design solution developed by the framework, which will enter the detailed design phase and which will ultimately be produced, other control methods and architectures can be developed, more appropriate for a real-life situation. Such a control system will only have to be tuned and developed once in contrast to the thousands of designs evaluated in an MDO framework. This application of model based inversion control is considered new.

Third, in order to avoid errors in the embedded control software as a result of manual programming activities, the dependencies of parameters in the software on physical parameters of an evolving design and the high complexity (thousands of lines of code), control software components of flight control systems should ideally be developed in an automated fashion. The proposed KBE system has the ability to generate consistent control software components. The system extracts the variable definitions and values from physical configurations and control architecture from the information model to specify the variables in the software components. In addition, the system divides software components into basic elements and writes them into strings, which can, in principle, be any computer language. When the top level configuration and control architecture changes, the proposed KBE system can operate the basic elements in specific order and automatically create new software components by capturing the expertise from software engineers. Summarizing, because both the geometry model and multiphysics simulation model including flight control system are obtained from one source, the MIM links the physical modeling and control system design with the development of software components with respect to data and topology structure.

A multirotor UAV configuration is used as test case to demonstrate the novel methods and tools described above. This is an inherently unstable configuration with a wide range of applications. A computational framework is developed which enables the conceptual/preliminary design and optimization of this typical mechatronic product. The proposed KBE system automatically creates thirty thousand designs of multirotor UAVs with different topologies and then evaluates each solution by automatically simulating five test maneuvers and by checking twenty-two constraints. Results show that the proposed KBE system can automatically generate multiphysics simulation models to support the multidisciplinary analyzes not restricted to the mechanical domain but also applicable for evaluation of flight control systems and other domains. Even though different design solutions can have a highly different topology, automatic flight control systems based on the model inversion control method are created automatically for each design solution, enabling the evaluation of the inherent flying qualities of the unstable aircraft configuration. Furthermore, within the framework, design processes are automatically completed from the initial definition of top-level aircraft requirements, to the design and optimization, and finally down to selecting feasible solutions. The approach demonstrated leads to: a reduction in manual work, improved quality of the final solution, and consistent control system and software components.

Key to the MIM concept is that it focuses on capturing the intrinsic properties of physical systems by the KBE approach and a specific format of representation is avoided. Although the current research study focuses on the software of the flight control systems in particular, the concept of the MIM can in principle be applied to design the complete E/E system, including hardware components, as well as other multiphysics systems.

SAMENVATTING

Momenteel worden er in de civiele luchtvaartindustrie vliegtuigen ontwikkeld waarin verschillende mechanische, hydraulische en pneumatische systemen vervangen worden door elektrisch aangedreven systemen. Dit heeft als doel om de vliegtuigprestaties te verbeteren door het totale energieverbruik van de systemen te verminderen en het totale vliegtuiggewicht te reduceren. De visie voor de toekomst is om uiteindelijk alle systemen elektrisch te maken en zelfs het voorstuwingssysteem (deels) elektrisch aan te drijven. Dit zou kunnen leiden tot hybride elektrisch of zelf volledig elektrische vliegtuig ontwerpen, afhankelijk van de toekomstige ontwikkelingen in batterijtechnologie. In de afgelopen jaren zijn er al verschillende kleine volledig elektrische vliegtuigen ontwikkeld om de mogelijkheid tot volledig elektrisch vliegen te demonstreren. Deze concepten hebben weliswaar de potentie om de totale complexiteit van vliegtuigen te verminderen, de complexiteit van alle elektrische system daarentegen zal sterk toenemen. De ontwikkeling en de integratie van de elektrische systemen in het vliegtuigontwerp is daarom met toenemende mate een nieuwe uitdaging voor de luchtvaartindustrie.

Twee specifieke vliegtuig categorieën die momenteel operationeel zijn ondervinden dezelfde uitdagingen. Deze categorieën zijn; (1) onbemande vliegtuigen, welke van nature meer elektrische systemen aan boord hebben en ook gebruik maken van een automatische piloot, en (2) vliegtuigen die inherent instabiel zijn en daarom gestabiliseerd moeten worden m.b.v. een automatische piloot. Deze twee categorieën kunnen geclassificeerd worden als mechatronische producten.

Elektrische systemen hebben een invloed op de vliegprestaties, vliegeigenschappen en zijn kritisch vanuit een veiligheidsperspectief. Deze systemen zouden daarom integraal ontwikkeld moeten worden met de traditionele disciplines zoals het aerodynamische en het constructieve ontwerp van het vliegtuig en het ontwerp en de integratie van het voortstuwingssysteem. Er zijn echter verschillende problemen die opgelost moeten worden voordat dit bereikt kan worden. In dit onderzoek worden drie specifieke uitdagingen geadresseerd.

- De ontwikkeling van vliegtuigsimulatiemodellen met een hoge mate van nauwkeurigheid voor de analyse en ontwikkeling van elektrische systemen

in het complete vliegtuig is een complexe, tijdrovende multidisciplinaire taak die veel handmatig werk vergt van simulatie experts;

- Het ontwerp van consistente automatische besturingssystemen die gebruikt kunnen worden in de gehele 'flight envelope' en goed werken bij alle mogelijk gewicht en zwaartepunt combinaties is een omvangrijke taak waarvoor nauwkeurige vliegtuigsimulatiemodellen benodigd zijn;
- De ontwikkeling van besturingssoftware is gevoelig voor het maken van fouten als gevolg van inconsistenties tussen de beschrijvingen van het fysieke ontwerp, de architectuur van het besturingssysteem en de bijbehorende software componenten.

Traditionele vliegtuigontwerp methoden zijn toegespitst op het fysieke ontwerp en zijn niet geschikt om tegelijkertijd complexe elektrische systemen te ontwerpen en te integreren in het fysieke ontwerp. Het conventionele vliegtuigontwerp proces is bovendien sequentieel en is daarom niet geschikt voor een parallel ontwerp proces waarbij verschillende subsystemen gelijktijdig ontwikkeld worden. Nieuwe ontwerp methodes, geïmplementeerd in software applicaties zijn daarom nodig om de ontwikkeling van elektrische vliegtuigsystemen te ondersteunen.

Het hoofddoel van dit onderzoek is om de tijd nodig voor de ontwikkeling van vliegtuigen met een hoge mate van geïntegreerde elektrische systemen te reduceren door het ontwerp proces van de elektrische systemen te automatiseren en door besturingssoftware te creëren met een hogere mate van consistentie. Naast een reductie in ontwikkelingstijd zal dit er ook toe leiden dat de kwaliteit van het (mechatronische) eindproduct verhoogd wordt. De drie bovengenoemde problemen zullen specifiek worden aangepakt. De nieuwe methodes en resulterende software applicaties zijn gebaseerd op de knowledge based engineering (KBE) techniek. Deze techniek is heel geschikt omdat deze alle niet-creatieve en repetitieve ontwerp en analyse taken, zoals uitgevoerd door de simulatie experts, kan automatiseren. Daarnaast kan deze techniek multidisciplinair ontwerpen, analyseren en optimaliseren mogelijk maken. Het nieuwe KBE systeem integreert het ontwerp van de elektrische systemen met het fysieke ontwerp op drie specifieke vlakken.

Allereerst, om een consistente representatie van het ontwerp mogelijk te maken is het 'multifysica informatie model' ontwikkeld met als doel om de ontwerpkennis van verschillende disciplines te integreren en vast te leggen. In het multifysica informatie model (een KBE systeem) worden objecten met attributen gedefinieerd om verschillende aspecten van fysieke elementen van het ontwerp te beschrijven (bijvoorbeeld massa, traagheid, geometrie, materiaaleigenschappen). Functies worden gebruikt om niet-fysieke informatie zoals de architectuur van het meet- en regelsysteem, relevante test manoeuvres, simulatie procedures, etc. vast te leggen. Interacties en koppelingen tussen de verschillende betrokken disciplines worden geïdentificeerd door middel van een kennisverwervingsproces en vastgelegd door het KBE systeem. De informatie en kennis aanwezig in het multifysica informatie model, welke benodigd is voor de ontwikkeling van een multifysica simulatie model, kan door het KBE systeem gedestilleerd worden. Een dergelijk simulatiemodel bestaat uit meerdere fysische modellen, modellen voor de elektrische systemen en de bijbehorende software en uit routines die het complete geïntegreerde simulatiemodel kunnen aansturen. Dit multifysica simulatiemodel kan gebruikt worden om de werking van besturings systemen te evalueren in realistische operationele condities. Samengevat maakt het multifysica informatie model het mogelijk om snel multifysica simulatie modellen te creëren welke gebruikt kunnen worden voor de analyse, ontwikkeling en integratie van elektrische systemen.

Ten tweede, om de inherente vliegeigenschappen van een instabiel vliegtuigontwerp te evalueren in een simulatie omgeving is een meet- en regelsysteem nodig. Hiervoor kan de model based inversion control techniek gebruikt worden. Deze methode heeft als grote voordeel dat de parameters niet nauwkeurig afgestemd hoeven te worden maar direct worden afgeleid van het simulatiemodel. Dit is heel effectief wanneer veel verschillende vliegtuigontwerpen geëvalueerd moeten worden. De processen die nodig zijn om een model based inversion control systeem te ontwikkelen zijn vastgelegd en geautomatiseerd in het KBE systeem. De model based inversion control techniek heeft een aantal nadelen wanneer deze niet geïmplementeerd wordt op een simulatiemodel maar op een echt vliegtuig. Voor het uiteindelijke vliegtuigontwerp kunnen eventueel andere meet- en regeltechnieken toegepast worden die geschikter zijn voor implementatie op het fysieke product. Deze hoeft men dan slechts eenmalig te ontwerpen in tegenstelling tot de potentieel duizenden ontwerpen die geëvalueerd worden in een multidisciplinair ontwerp en optimalisatie proces. Dit wordt beschouwd als een nieuwe toepassing van de model based inversion control techniek.

Ten derde, om fouten in de besturingssoftware te voorkomen zou deze idealiter automatisch gegenereerd worden. Deze fouten kunnen het gevolg zijn van handmatige fouten, de afhankelijkheid van software parameters op de ontwerp parameters van een evoluerend ontwerp en de hoge mate van complexiteit (duizenden regels software code). Het nieuwe KBE systeem is in staat om automatisch consistente besturingssoftware te produceren die ook makkelijk te begrijpen is en een duidelijke relatie met ontwerpparameters heeft.

Het systeem definieert variabelen in de besturingssoftware gebaseerd op de beschrijving van de architectuur van het besturingssysteem in het informatie model, de definities van variabelen in het informatie model en de waardes van fysieke ontwerpparameters. Het KBE systeem kan in principe automatisch software code schrijven in elke willekeurige programmeertaal. Als de configuratie van het ontwerp en/of de architectuur van het besturingssysteem verandert kan het systeem automatisch opnieuw consistente software genereren. Samengevat slaat het een brug tussen het ontwerp van het meet- en regelsysteem en het fysieke model in de ontwikkeling van besturingssoftware componenten.

Een kleine onbemande helikopter met meerdere rotors is de testcase voor de demonstratie van de nieuwe ontwerp methoden. Deze helikopter is inherent instabiel en heeft vele mogelijke toepassingen voor het gebruik. De ontwerp methoden zijn geïmplementeerd in een software applicatie die het mogelijk maakt om automatisch het voorontwerp van dit voertuig door te rekenen en te optimaliseren. Voor deze testcase creëert het KBE systeem automatisch circa dertigduizend ontwerpen met verschillende topologie van de onbemande helikopter. Voor elk ontwerp worden automatisch vijf verschillende manoeuvres gesimuleerd en tweeëntwintig randvoorwaarden voor het ontwerp worden gecontroleerd. De resultaten laten zien dat het KBE systeem automatisch multifysica simulatie modellen kan genereren in een multidisciplinaire ontwerp omgeving. Ook al hebben verschillende ontwerpen een significant andere topologie, voor elk helikopter ontwerp wordt automatisch een meet- en regelsysteem ontwikkeld gebaseerd op de model based inversion control methode. Dit maakt het mogelijk om de inherente vliegeigenschappen van een instabiele vliegtuigconfiguratie te analyseren. Bovendien worden alle processen, van de lijst met eisen tot en met ontwerpen, analyseren, optimaliseren en het selecteren van een eindoplossing volledig geautomatiseerd. De nieuwe aanpak leidt tot een vermindering van handmatig werk, een hogere kwaliteit van het eindproduct en een meet- en regelsysteem met consistente besturingssoftware.

De kern van het multifysica informatie model concept is dat het de intrinsieke karakteristieken van fysische systemen vastlegd door middel van de KBE benadering zonder een specifiek format voor te schrijven. Het onderzoek gepresenteerd in dit proefschrift is met name gericht op de automatische ontwikkeling van besturingssoftware en simulatiemodellen. Echter, het multifysica informatie model kan in principe ook worden toegepast voor het ontwerp van complete E/E systemen, inclusief hardware en andere multi fysische systemen.

Contents

ACRONYMS

ABS	anti-lock braking system	100
AEA	all electric aircraft	2
AHP	analytic hierarchy process	52
AI	artificial intelligence	20
AM	architecture model	11
ASCEND	advanced system for computations in engineering design	45
CAD	computer-aided design	8
CAE	computer aided engineering	46
CAMP-G	computer aided modeling program with graphical input	10
CD	creative design	38
CE	concurrent engineering	5
CEASIOM	computerized environment for aircraft synthesis and integrated optimization methods	46
CFD	computational fluid dynamics	45
CSV	comma-separated values	36
CVG	constant velocity gearbox	2
DEE	design and engineering engine	24
DRY	don't repeat yourself	103
E/E systems	electrical and electronic systems	3
FEM	finite element method	45
GA	genetic algorithm	22
GDL	genworks general-purpose, declarative, language	27
HIP	hardware-in-the-loop	9
HLCt	high level CAD templates	45
HLPs	high level primitives	45

KBE	knowledge based engineering	12
MBSE	model-based system engineering	44
MDO	multidisciplinary design optimization	11
MDQ	mechatronic design quotient	10
MEA	more electric aircraft	1
MIM	multiphysics information model	12
MMG	multi-model generator	24
MOP	multi-objective optimization problem	21
OOP	object oriented programming	20
PCS	process control systems	46
PIP	product introduction process	20
PSO	particle swarm optimization	38
PVA	proportional plus velocity plus acceleration	84
RD	reference design	37
UAV	unmanned aerial vehicle	7
VEGA	vector evaluated genetic algorithms	29

NOMENCLATURE

Greek symbols

Ω	Feasible region	-
ε	Arbitrarily small positive quantity	-

Roman symbols

Δk_{np}	Coefficient of motor speed due to roll input	-
Δk_{nq}	Coefficient of motor speed due to pitch input	-
Δk_{nr}	Coefficient of motor speed due to yaw input	-
Δk_{nw}	Coefficient of motor speed due to altitude input	-
Δn_p	Rotor speed change as a result of a roll command input	rad/s
Δn_q	Rotor speed change as a result of a pitch command input	rad/s
Δn_r	Rotor speed change as a result of a yaw command input	rad/s
Δn_w	Rotor speed change as a result of an altitude command input	rad/s
$\mathbf{A}$	System matrix	-
$\mathbf{B}$	Input matrix	-
$\vec{f}(\vec{x})$	Objective functions	-
$a1$	New value of the wheel speed derivative threshold	-
$a1'$	Old value of the wheel speed derivative threshold	-
C_i	Cost of component i	\$
D	Damping coefficient of shock absorber	$N*m/s$
D_v	Relative velocity of shock absorber	m/s
f	Reaction force of suspension	N
$fitness(x)$	Fitness function	-
G	New value of gross weight	kg
g	Gravitational acceleration	m/s^2
G'	Old value of gross weight	kg

$g_i(\vec{x})$	Inequality constraints	-
$h_i(\vec{x})$	Equality constraints	-
k	Stiffness of air spring	N/m
K_i	Control system gain	-
N	Number of rotors	-
p	Body axis roll rate	rad/s
q	Body axis pitch rate	rad/s
R	New value of front axle load ratio	%
r	Body axis yaw rate	rad/s
R'	Old value of front axle load ratio	%
$rank(x)$	Ranking function	-
T	Thrust	N
u_x	Speed command in longitudinal direction	-
w	Vertical velocity in body axis	m/s
W_i	Weight of component i	N
x	Deformation of air spring	m
x_i	Decision variables	-

1
INTRODUCTION

1.1. CHALLENGES FOR UNSTABLE AND UNMANNED AIRCRAFT DESIGN

1.1.1. FROM MORE ELECTRIC TO ALL ELECTRIC AIRCRAFT

ON conventional aircraft, all on-board functions are driven by hybrid mechanical, hydraulic, pneumatic, electrical and sometimes “fueldraulic” non-propulsive power systems for military and commercial aircraft [1]. Fueldraulic systems are hydraulic systems pressurized by fuel. Although the performance in this complex hybrid non-propulsive power system has been improved over time, it is still the cause aircraft maintenance “down-times” and failures [1]. For this reason, the US Air Force embarked on a research initiative called the more electric aircraft (MEA) in the 1990s [2]. It has been pointed out that the MEA, in which the centralized aircraft hydraulic power systems are replaced with electrically-based power systems, will greatly improve reliability, maintainability and supportability as well as the potential for significant performance improvements in terms of weight, volume and system complexity [2, 3].

A comparison between conventional aircraft subsystems and the MEA subsystems is illustrated in Figure 1.1 [4]. A more specific example is given by Blanding [5], which is shown in Figure 1.2. Technologies, such as fly-by-wire and fly-by-light, are typical MEA applications [6]. Blanding also emphasizes that one benefit of the MEA approach is a reduction in power conversion, where it is not necessary to convert engine shaft power to electric, hydraulic and pneumatic power systems [5]. Moreover, it has been discussed that the electric power systems cannot only be used to replace the original hydraulic actuator systems,

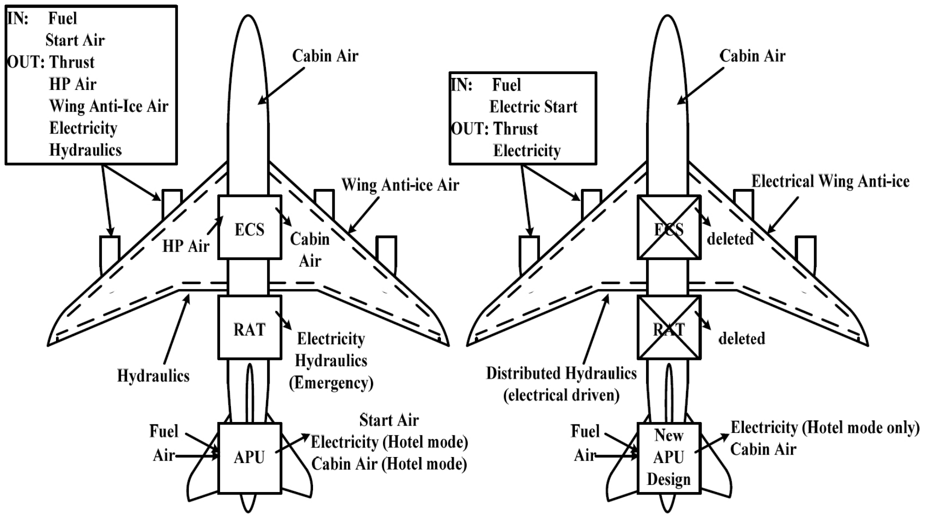


Figure 1.1: Comparison between conventional aircraft systems and more electric aircraft (MEA) systems [4]

such as flight control surface actuators, engine fuel pumps, brakes, landing gear nose wheel steering systems and de-icing systems but also be integrated with the gas turbine as generators [7]. Finally, the concept of the MEA has been applied on the Airbus A380 to replace the constant velocity gearbox (CVG) with the electrical power system [8]. This has also been considered by the Boeing B787 [9].

It should be noticed that the ultimate goal is to achieve an all electric aircraft (AEA), which has no hydraulic or pneumatic systems, resulting in many

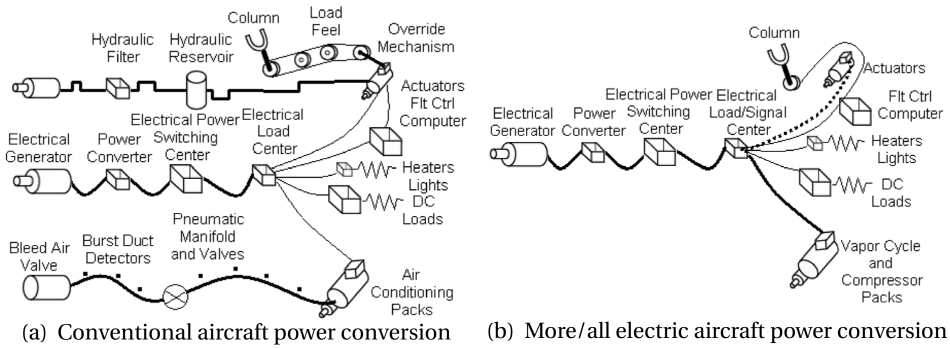


Figure 1.2: Comparison of conventional aircraft power conversion with more-electric/all-electric aircraft power conversion [5]

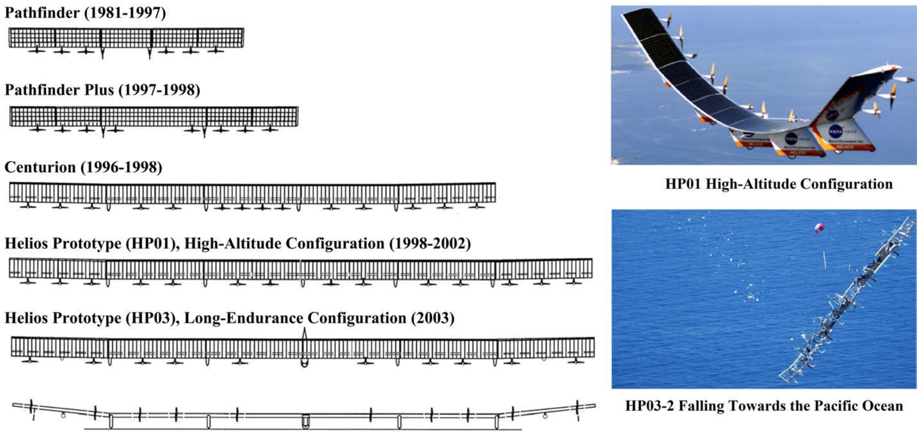

Pathfinder (1981-1997)
Pathfinder Plus (1997-1998)
Centurion (1996-1998)
Helios Prototype (HP01), High-Altitude Configuration (1998-2002)
Helios Prototype (HP03), Long-Endurance Configuration (2003)
HP01 High-Altitude Configuration
HP03-2 Falling Towards the Pacific Ocean

Figure 1.3: Solar aircraft evolution through the ERAST program [16]

advantages [10]. In addition, the AEA is driven by electrical aero propulsion, such as electrical motors [11]. Hoffman et al. [12] estimate that the AEA concept can reduce aircraft weight by 10% and fuel consumption by 9%. For this to become reality, significant advances in battery technology are required.

Many small-size electric aircraft have been developed in recent years to demonstrate the feasibility of the AEA concept. Typically, electric aircraft are driven by electric motors with the electricity coming from batteries, fuel cells, solar cells, etc. The Airbus E-Fan uses on-board lithium batteries to power two electric engines and can carry two persons [13]. Moreover, the EADS Cri-Cri is an ultralight aircraft powered by four electric engines. It is a demonstrator for future technologies [14]. Furthermore, the ENFICA-FC aircraft is proposed by the European Commission to demonstrate an all-electric aircraft with fuel-cells as supply for the main and auxiliary power system [15]. Another example is the NASA pathfinder, a solar- and fuel-cell-system-powered unmanned aircraft developed by AeroVironment, Inc [16]. However, this is just the beginning. New propulsion technologies, such as distributed propulsion [17] or electro thermal turbo propulsion [18], may be enabling the electric aircraft of the future.

Although the MEA and AEA concepts reduce the complexity of the aircraft by replacement of hydraulic and pneumatic power systems with electric power components, it reversely increases the complexity of electrical and electronic systems (E/E systems), introducing a new challenge for the design.

For example, the NASA solar aircraft has evolved from pathfinder (1994) to the HP03 (the long-endurance configuration, 2003), which is shown in Figure 1.3. The project finished with the HP03 aircraft. Several reasons for failure were identified. The research team determined that persistent high dihedral causes instability of the aircraft. The origin of this problem was tracked back to the design phase. A more specific reason for failure was due to inadequate design and analysis tools, cited as followed [16]:

> *The **complexity** and **interactions** between **aeroelastic** and **stability modes** made it difficult to apply **time domain stability and control analysis** to the vehicle.*

The interactions between aero elasticity and flight dynamics were essentially caused by a highly flexible airframe which was required to achieve a light weight design with a large surface area for solar panels and a large aspect ratio for flight performance. For such a design, high fidelity multiphysics simulation models are required in the early design phases and an integrated concurrent design approach is essential.

Summarizing, the trend from the conventional aircraft to MEA and ultimately to AEA increases the complexity of the E/E systems on the aircraft, introducing a new challenge for the traditional aircraft design methods.

1.1.2. A MULTIDISCIPLINARY DESIGN TASK

As is shown in Figure 1.4, the design of an aircraft is a difficult task, involving many scientific/engineering disciplines, such as aerodynamics, structures, propulsion, E/E systems, etc. The multidisciplinary analyses are performed sequentially in the traditional conceptual design process. Typically, the optimal aircraft design is found by several design iterations [21]. Because the performance of a multidisciplinary system is driven not only by the performance of the individual disciplines but also by their interactions [22], the

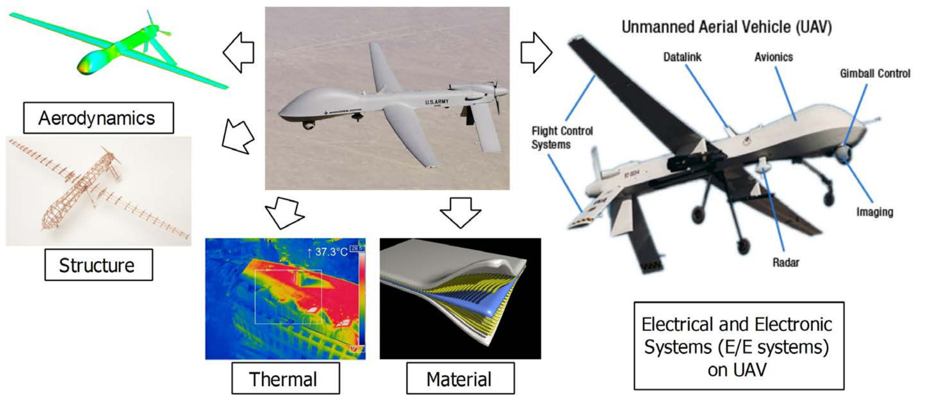


Figure 1.4: Multidisciplinary design of unstable and unmanned aircraft (adapted from [19, 20])

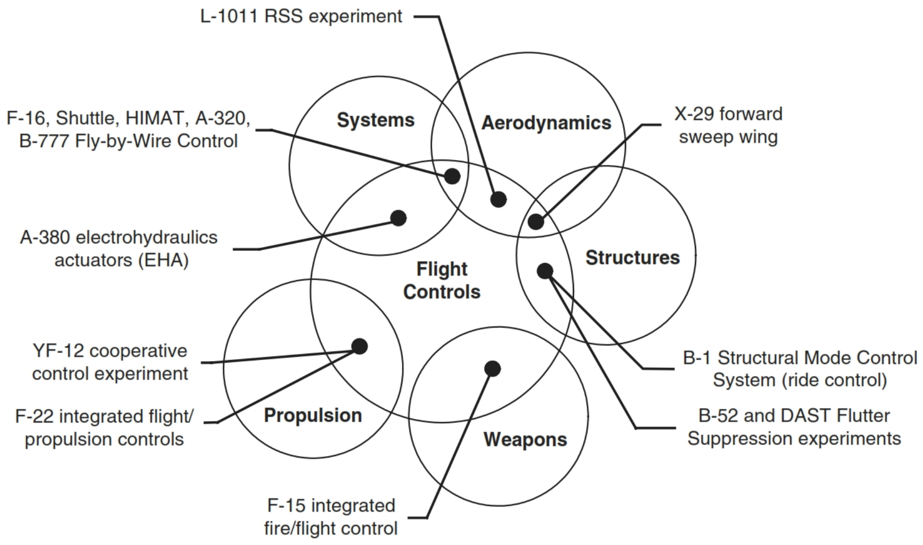

L-1011 RSS experiment
F-16, Shuttle, HIMAT, A-320, B-777 Fly-by-Wire Control
Systems
Aerodynamics
X-29 forward sweep wing
A-380 electrohydraulics actuators (EHA)
Flight Controls
Structures
YF-12 cooperative control experiment
F-22 integrated flight/ propulsion controls
Propulsion
Weapons
B-1 Structural Mode Control System (ride control)
B-52 and DAST Flutter Suppression experiments
F-15 integrated fire/flight control

Figure 1.5: Integration of E/E systems with other disciplines [26]

sequential process may lead to suboptimal designs due to its inability to capture the system couplings of the various disciplines [21].

As a result, the E/E systems have a significant effect on the aircraft performance and cost [23]. This should be analyzed in the conceptual/preliminary design phase [24, 25]. As can be seen in Figure 1.5, the E/E systems are integrated with other disciplines, such as aerodynamic, structures, propulsion, etc. Therefore, the multidisciplinary problem requires methods and corresponding tools to achieve an efficient concurrent engineering (CE) design environment both for the E/E systems and the aircraft.

1.1.3. REQUIREMENTS OF CONCURRENT ENGINEERING

Previously, the CE approach has been used as a guidance for reducing the time-to-market for new product development [27]. Compared with the sequential process, which is described as a "relay race" by Takeuchi and Nonaka, the concurrent process is referred to as a "rugby team" which emphasizes on cross-functional integration [28]. Typically, two concepts are essential for the CE approach. First of all, all elements of a product's life-cycle should be taken into careful consideration in the early design phases [29]. Second, the design activities should all be occurring at the same time [29].

In this research study, two factors are considered to achieve a successful CE approach for the E/E systems and other disciplines. First of all, the design of the E/E systems should be taken into account with other disciplines in the early design phase. In other words, flight performance, stability and control aspects of the aircraft should be considered in the conceptual design stage together with the analysis results from other domains. Secondly, the development of the E/E systems should also keep pace with the design activities from other domains. Needless to say, the modeling of the physical system is a key activity during the design process. It has been pointed out that geometry is the most effective enabler for the integration of disciplines and it is also the most commonly used thread through the different disciplines required [30]. However, the E/E systems are usually represented by logical diagrams and architectures, software components, transfer functions or state space matrices, which are hard to obtain directly from the geometry model. Therefore, the second requirement of CE for the E/E systems and other disciplines is addressed by the following two aspects in this research study:

- Generation of simulation models not limited to geometry but also for the E/E systems and other domains;
- The model generation should keep enough consistency to support multidisciplinary analyses.

1.1.4. COMPLEXITY OF E/E SYSTEMS ON ROAD VEHICLES AND AIRCRAFT

No matter whether fixed wing aircraft, helicopters or road vehicles are considered, the E/E systems are extremely complex. Typically, there are more than 2500 software controlled functions on a modern passenger car, representing 10 million lines of software code [31]. Consequently, the automotive industry is under increasing pressure to prevent unexpected failures in the E/E systems, both in the electronic components and associated software functions. For example, TOYOTA recalled the 9400 Lexus GX460 in 2010 [32]. It is reported that if the vehicle was driven through a sharp turn at high speed, the vehicle could skid in a sideways direction. For this reason, the vehicle stability control systems had to be reprogrammed to solve this problem [32]. Unfortunately, the established development processes, which aim at the efficient creation of high quality mechanical systems, cannot deal with the problem of the high complexity of the E/E systems [31]. Finally, the increasing failures related to the E/E systems make up more than 10% of the road vehicle production in recent years [33].

Aircraft are even more complex and have more E/E systems compared to road vehicles. As can been seen in Figure 1.6, the conventional transport aircraft has many control surfaces and other hydraulic and pneumatic systems. As for the AEA, all of them can be replaced by electric power systems, which are illustrated by Table 1.1 [34] for a generic MEA fighter.

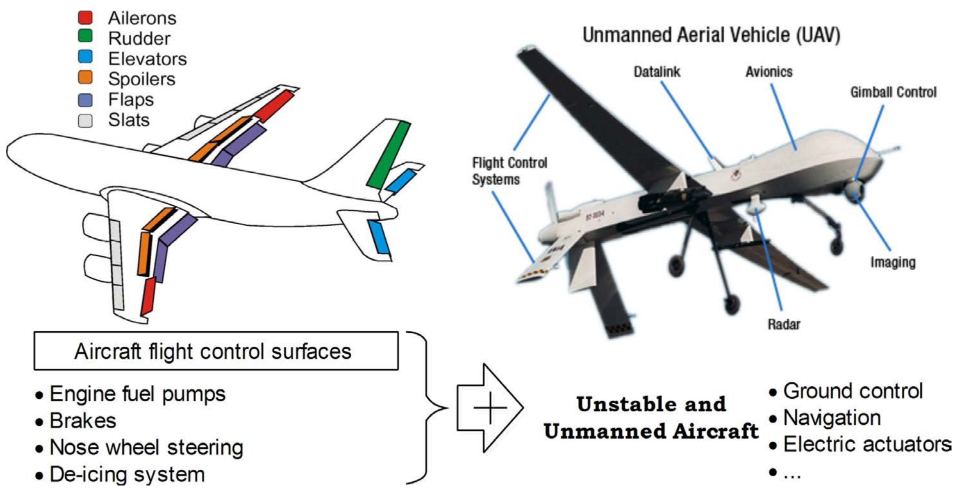


Figure 1.6: Complexity of electrical and electronic systems (E/E systems) on unstable and unmanned aircraft (adapted from [7, 20])

Subsystems	No. of motor drives	Largest motor (kw)	Total power (kw)
Flight control	28	50	80
Environmental control	10	10	40
Engine starter/generation system	6	125	125/channel
Landing system	20	5	30
Fuel pump	10	9	35
Pneumatic system	2	15	30
Miscellaneous	10	1	20
Total	86	-	360

Table 1.1: Motor driven requirements for a generic fighter [34]

For inherently unstable aircraft and unmanned aircraft configurations, the number of on-board E/E systems is even larger than on conventional transport aircraft. On one hand, the traditional energy supply, in the form of fuel, will most likely be replaced partly by batteries or fuel cells. These aircraft are emerging currently worldwide to fulfil a large variety of tasks. On the other hand, besides the electric actuators for the aircraft flight control surfaces, the unmanned aerial vehicle (UAV) needs ground control, navigation and extra control system for auto-stabilization due to the fact that there is no pilot inside.

A comprehensive certification procedure is in place to ensure everything works as planned during the first flight [35]. A call-back, such as in the automotive industry, is not an option because of safety considerations.

Nevertheless, aircraft design faces the same challenge as the automotive industry and this problem is likely to introduce delays in the future design processes.

1.1.5. CHALLENGES FOR THE DEVELOPMENT OF E/E SYSTEMS

The design of the E/E systems is a difficult task due to its high complexity. This situation becomes even worse when the physical systems are rapidly changing in the conceptual design stage. As mentioned in the previous example (Figure 1.3), the NASA solar aircraft HP03 has a wingspan, two times larger than the first generation pathfinder, which caused a problem with respect to interaction s between aeroelasticity and flight dynamics. In fact, because the aircraft flight control surfaces can be modeled as parameterized geometry models, it is straightforward to change the size in computer-aided design (CAD) tools. The number of engines can also be easily duplicated along a specific direction by CAD program. However, it takes a large amount of time and effort to rebuild the mathematic model representing the aircraft behavior and to tune the parameters of the flight controls to adapt such modifications due to non-linear characteristics of the aircraft. It can be expected that the E/E systems have to be repeatedly tuned and tested for the whole design envelope. Three aspects are discussed in more detail below.

First of all, modeling of E/E systems is a difficult task, which requires a multiphysics simulation model to represent the complex systems across multiple engineering domains. On one hand, because more and more systems are electrically controlled, the simulation experts have to create the models both for the original hydraulic and pneumatic systems and for the new electric power systems and they have to make detailed comparisons by means of extensive analysis to ensure proper functionality and similar response. Moreover, the simulation experts also have to select the most suitable components for their purposes from a huge number of different technologies, component libraries and other domains. The interactions among different components of the overall system should also be considered. On the other hand, if the simulation accuracy is required, the model fidelity should be high, which means every element in the physical world should find a corresponding mathematic representation with sufficient detail. Therefore, the multiphysics model can be more complex than the actual physical systems. In short, it takes much time and effort to generate the (high-fidelity) multiphysics simulation model due to the extreme complexity of the E/E systems.

Secondly, it has been pointed out that, in order to achieve consistent handing qualities throughout the operational flight envelope, the design of a control system for unstable and unmanned aircraft is a difficult task and results in substantial cost and time [36]. The problem is that because the variables of the control systems and the overall architecture are linked directly to the physical plant, when the top level configuration changes, the variables have to be tuned again and the architecture may have to be changed. It can be expected that this process will be repeated many times during the whole period of aircraft development, which is time consuming and prone to errors.

Third, control software is required to test the E/E systems in a more realistic environment, such as hardware-in-the-loop (HIP) simulation. The inconsistencies between the top level configuration, control architecture and the software components have caused many software errors. On one hand, there are million lines of software code which are difficult to be understood. Because the data and control logic are separately stored and saved, it is hard to link the values and parameters in the control software with the physical system in the real world. On the other hand, it also costs extra effort to develop and maintain the control software, which involves a large amount of manual work. Weule et al. [37] summarize the most common errors present in control software, like multiple use of the same variable, wrongly setting and resetting of variables, typing errors, etc. Spath and Landwehr point out that almost 70% of the errors during the software development of control technology are software errors [38]. For example, when the topology structure of the physic system is extended, the number of variables, definitions and corresponding comments have to be multiplied consequently. In this case, errors may occur due to improper typing or mistakes.

In short, because of complexity, the development of the E/E systems is a difficult task. Three specific challenges are identified:

- Setting up multiphysics simulation models for the E/E systems is a complex and multidisciplinary task;
- It costs a large amount of time and effort to design consistent control systems for the entire flight envelope;
- The development of control software components is prone to errors due to inconsistencies between the top level physical configuration, control architecture and software components.

1.2. METHODOLOGIES TO SUPPORT THE DEVELOPMENT OF E/E SYSTEMS

1.2.1. THE MECHATRONIC DESIGN APPROACH

As indicated earlier, the complexity of the E/E systems leads to failures in the automotive industry and the delay of aircraft. This phenomenon requires the manufacturers to transfer part of their attention from mechanical design aspects to the integration of the E/E systems into their products. Therefore, the highly integrated mechanical and E/E systems need methods and tools that can deal with both of them in the early design phases. This is called mechatronic design. Originally, mechatronic design or mechatronics is defined by Yasakawa Electric Company as the combination of mechanics and electronics [39]. This concept has been extended to include more technical areas, such as control engineering and computer engineering [40, 41]. In literature, many researchers propose methods for mechatronic design.

Mathematic equations are the most direct way to describe the physical system, both from the perspective of mechanics and of electronics. For example, Maira et al. [42] represent the dynamics behavior of a pick-and-place assembly robot using flexible multibody dynamics. The flexible multibody system is modeled using mathematical equations. Control system design is based on these equations as well. Moreover, other researchers use an evaluation model called mechatronic design quotient (MDQ) to facilitate decision-making in the design process. It is claimed that the controller design issues and parameters are treated simultaneously with other physical issues and parameters [43].

Several software packages are developed for the mechatronic design based on bond graph technique. The bond graph uses elements and junctions to represent the physical system. The definition of the elements and junctions are different in mechanical domain and electrical domain. The advantage is that the mechatronic system can be represented in a compact way. Moreover, the topology structure could be easily modified to build different mechatronic systems by manipulating the elements of bond graph. A software implementation named computer aided modeling program with graphical input (CAMP-G) has been developed with the bond graph technique to automatically generate computer models [44]. 20-sim also supports the domain independent bond graph notation for modeling dynamics systems [45].

However, when the physical system is represented by mathematic equations or bond graphs, too much geometric information has been lost. As a result, this approach cannot support the use of analysis tools requiring high fidelity geometric models. Nevertheless, in the conceptual/preliminary design phase, high fidelity analysis is also required to ensure key requirements are met, such as the aerodynamic performance for aircraft.

Cabrera et al. [46] also notice the complexity of designing a mechatronic system and propose to define a high level model of the system to represent the top-level conceptual hierarchy by utilizing the functional modeling. It is argued that the use of functions as integration elements can represent a system at different levels of detail, focusing on the interests of the user while maintaining coherence of the model [47]. Subsequently, Cabrera et al. [48] propose an architecture model (AM) to support cooperative design for mechatronic products. The AM is built with abstract, high level representations, such as functions and behaviors. The concept of the AM is tested to represent several applications, such as an air condition and a formula student car [49]. The AM approach is useful for modeling the system architecture of complex systems, but it lacks of modeling of physics and other behavioral concerns and cannot deal with irregular situations at that moment [48].

Other researchers focus on the further development of CAD tools for mechatronic design. The basic principle is to extract information from the geometrical model and to prepare it for dedicated simulation model. Therefore, models can be produced with very high fidelity for most analysis tools. Some of them extract information from CAD representation of the physical components to Modelica models dedicated to multibody simulation [50]. Other researchers use component objects which are a combination of CAD models and simulation models [51]. A further research study directly generates the simulation model out of geometry models by mapping between multi-skill engineering domains [52]. However, in these research studies, it should be noticed that the control system is not included in the simulation model which is actually the dynamics model only. The reason is that the control system is usually represented by transfer functions or state space matrices, which are difficult to derive from the geometry model directly. Consequently, it is difficult to achieve a concurrent design environment for both the control system and other engineering disciplines by merely extracting the information from CAD tools.

1.2.2. THE MDO APPROACH

For fulfilling requirements of multiple domains with high complexity, multidisciplinary design optimization (MDO) has been proposed and this technique has rapidly evolved in the last decades of the 21st century. MDO is a field of engineering that focuses on the use of numerical optimization for the design of systems that involve a number of disciplines or subsystems [22]. It is suggested when the performance of a multidisciplinary system is driven not only by the performance of the individual disciplines but also by their interactions [22].

Many researchers proposed methods and tools to support the MDO process. These methods have two disadvantages. An inconsistency error may occur due to lack of data and information communication among the modeling languages. Moreover, a large amount of manual work is required to process/retrieve information from one domain and to reuse it in other domain.

These are further discussed in Chapter 3. Thus, a comprehensive set of methods and tools is required to fully support the MDO approach in all design phases.

1.2.3. THE KBE APPROACH

Knowledge based engineering (KBE) is a technology based on dedicated software tools called KBE systems, which are able to capture and systematically reuse product and process engineering knowledge, with the goal of reducing time and costs of product development [53]. KBE systems cannot only automate the repetitive and non-creative design tasks but also support MDO in all the phases of the design process [53]. Previously, the KBE approach has proven useful for developing complex systems, like aircraft [53], which mainly focuses on the manipulation of high level geometry model for aircraft conceptual design.

This research proposes novel methods and tools to support the control system development of the E/E systems for unstable and unmanned aircraft by using the KBE approach.

To ensure a consistent design representation for serving multidisciplinary analysis, this research study proposes an intelligent modeling system to automatically generate multiphysics simulation models to support multidisciplinary design optimization processes by using a KBE approach (Chapter 3). A key element of this system is a multiphysics information model (MIM), which integrates the design and simulation knowledge from multiple engineering domains. The MIM defines classes with attributes to represent various aspects of physical entities. Moreover, it uses functions to capture the non-physical information, such as control architecture, simulation test maneuvers and simulation procedures. Depending on the domain requirements, the intelligent modeling system extracts the required knowledge from the MIM and uses this first to instantiate submodels and second to construct the multiphysics simulation model by combining all submodels.

Moreover, in order to evaluate the flight performance of the complete aircraft, including E/E systems, it is necessary to design the flight control systems which regulate the behavior of physical plant. However, it is not a straightforward to develop the control systems because the physical plant can have non-linear characteristics throughout its range of operation and there may be a very large amount of functionalities in the control systems. Moreover, it takes a lot of time and effort to select the proper control architecture, select a control method, configure the simulation model, and tune the parameters of control systems at specific operating conditions. In addition, these tasks need to be conducted for each design (potentially in the order of thousands) to be evaluated by the MDO framework. Thus, a consistent control system development must be ensured throughout the whole design envelope.

The model based inversion control method is particularly suited for use within MDO because it doesn't require tuning and the control law can be directly derived from the simulation model. The only input needed is a linear model of the physical plant, which can be obtained by means of numerical perturbation of the model after the physical model (including control allocator) is trimmed in a specific flight condition. Furthermore, a dedicated control system can be directly created for different operating conditions (e.g., high speed forward flight or hover). The intelligent modeling system is again used to automate the design of control system by capturing the process information of model based inversion approach. This application of model based inversion control is considered novel (Chapter 4).

Furthermore, the E/E systems require control software components to regulate the behavior of physical plant in the real world. Software errors typically constitute more than half of the total errors encountered during the development of control systems. This is a result of the large amount of manual and repetitive activities of programmers in combination with the complexity of the overall code. Therefore, many research studies and commercial tools focus on automated generation of software components or automatic code generation. This works well, however the resulting code is often complex, lengthy, hard to understand and has no clear relation anymore with the original physical system in terms of parameters, etc. How to modify the existing software following a physical change in a product is a significant challenge. Because the development of complex engineering applications such as aircraft is a true MDO problem, it requires a seamless integration of the control software with the other engineering disciplines. This challenge is not tackled by existing methods and tools. Therefore, the intelligent modeling system is further developed in Chapter 5 to generate consistent control software components for the E/E systems. On one hand, it extracts variable definitions and values from the MIM to specify the variables in the software components. On the other hand, it also divides the software components into basic elements and writes them into strings. When the top level configuration and control architecture changes, the intelligent modeling system can operate the basic elements in specific order to create new software component by capturing the knowledge from simulation experts, control system design experts, or handbooks. Therefore, the MIM directly links the physical modeling and control system design with the development of software components with respect to aspects of data and topology structure.

The objective of this research is to reduce the development time of complex engineering application by automating the design process of the E/E systems

Methodology	Geometric modeling	Dynamics modeling	Control system design	Software code generation
Direct use of mathematic equations (physics representation)	Low fidelity model	Equations	Equations	-
Bond graph	Low fidelity model	Block diagram	Block diagram	Automated generation of control software
CAD tools (geometric representation)	High fidelity parameterized model	Multibody dynamics	-	-
KBE systems	High level primitive model	Generic multiphysics simulation model	Automated design of control system	Automated generation of control software

Table 1.2: Different methods for mechatronic design

and by creating more consistent control system/software throughout the entire design envelope.

The proposed KBE system is compared in Table 1.2 to other researchers' work in the field of mechatronic design, with a special emphasis on modeling E/E systems. Compared to related work, the proposed KBE system fills the gap between the mechanical design and the E/E systems, which is characterized by :

- Ensuring a consistent design representation both for the mechanical domain and the E/E systems;
- Synchronous design of mechanical systems, control systems and software components to enable the benefits of the MDO approach;
- Automating the mechatronic design both for modeling the mechanical components and developing the control systems.

1.2.4. RESEARCH SIGNIFICANCE

In this research study, the proposed methods and tools brings three distinct benefits.

First of all, the time and effort to generate simulation models for the evaluation of E/E systems in realistic operating conditions is reduced. Moreover, the design of the flight control system which enables the evaluation of the inherent flight dynamics within an MDO framework is automated by using the model based inversion control method. Furthermore, the proposed KBE system also automatically produces consistent control software components that adapt to changing requirements from the top level configurations.

Secondly, it provides consistent E/E systems throughout the design envelope. In this research study, the proposed KBE system defines a MIM to capture the design knowledge in multiple domains. Submodels can be directly instantiated from the MIM and constructed as the multiphysics simulation model for specific analysis purpose. Because all the submodels are obtained from one source (the MIM), a consistent design environment is established. Moreover, the proposed KBE system develops the control system with model based inversion control approach which is a promising method to generate consistent control systems throughout the whole design envelope. Furthermore, the variable definitions and software structure are linked to the physical model and control architecture, which in turns ensures the consistency for the generation of control software components.

Third, unexpected errors as a result of repetitive design activities can be prevented. In order to delete the errors for modeling the complex system, the proposed KBE system can automatically complete an optimization cycle from modeling to simulation to evaluation. The process automation for the model generation and the development of control systems is expected to prevent many manual errors. Moreover, because all the variable definitions in the control software components are inherited from the physical model, it can also avoid software errors due to repetitive definitions, incorrect typing or plain mistakes. Furthermore, the structure of control software components is customized, making it very concise and easy to read.

1.2.5. TEST CASE

The proposed KBE system is tested by designing and optimizing an unstable and unmanned all-electric aircraft - multirotor UAV. A multirotor UAV with four rotors (quadrotor UAV) is shown as an example in Figure 1.7. Although the multirotor UAV is not as complex as commercial aircraft or road vehicles, it is a typical mechatronic product which is composed of several modular components, such as the motor, propeller, landing gear, battery, etc. Moreover, the development of the multirotor UAV is a highly multidisciplinary task, which requires knowledge from mechanical design, electronic control systems, electrical systems (or components), aerodynamics, flight dynamics, etc. Furthermore, the topology of the multirotor UAV can also be subject to severe changes in the early design phases, for example, the number of rotors can be varied from 4, 6, to even more rotors. Finally, the multirotor UAV is an inherently

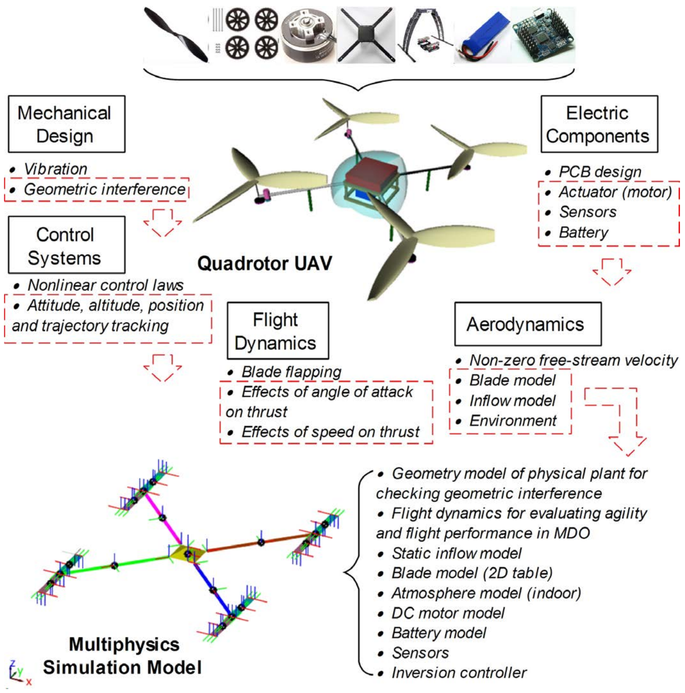


Figure 1.7: The fidelity of multiphysics simulation model of multirotor UAV with four rotors

unstable vehicle with N rotors to achieve six degrees of freedom. Since the translational and rotational motion are coupled, control of the multirotor UAV is even more difficult than for normal aircraft when aerodynamic effects are considered.

Therefore, it takes a lot of time to manually develop a single multiphysics simulation model, representing a single design of the multirotor UAV, taking into account all discipline couplings and interactions between components. Moreover, it may be necessary to develop multiple multiphysics simulation models for a single design in order to evaluate different design requirements. In order to represent the complete design envelope of the multirotor UAV (with highly different top level aircraft configurations), a vast number of different multiphysics simulation models is required. Thus, this is a perfect example to demonstrate how the proposed KBE system can accelerate the development process and solve the challenges for the development of the E/E systems in Section 1.1.5.

The objective of the test case is to automate the design of the E/E systems for the multirotor UAV synchronously with the other engineering domains. In this research study, the multiphysics simulation model of multirotor UAV is generated with enough level of fidelity to check whether the E/E systems of the current design concept give reasonable results and whether the associated control systems are successfully designed along with other domains throughout the whole design envelope. As can be seen in Figure 1.7, the physical plant of multirotor UAV is considered as rigid body dynamics. Some factors, such as blade flapping for flight dynamics, are neglected in current research study.

However, the concept of the MIM (in Chapter 3) supports to construct a comprehensive simulation model with very high fidelity when enough design knowledge are captured from handbooks, simulation experts, engineers, etc. In previous studies, an automotive project called "FORWARD" [54] was conducted in the Netherlands by several trailer manufacturers, one axle manufacturer and two universities. The aim of this project was to measure and identify real-life loading conditions on trailers during everyday operation and to use these for calculation of the fatigue on critical components, enabling weight reduction by re-design. A whole range of different trailer configurations was considered in the project, such as deep loaders, conventional trailers, trailers for transport of liquid cargo, double decker trailers, etc. A generic truck-trailer multiphysics simulation model was successfully developed which could accurately predict the loads on particular components of all these vehicles when subjected to a range of maneuvers. A validation of this generic model is presented in Appendix A. Because the simulation model can be created and analyzed in an automated fashion, it is possible to integrate this tool within an optimization framework [54].

As is shown in Figure 1.8, the research objectives can be divided into various elements. Firstly, the concept of the MIM is proposed to capture different physical aspects of the complex systems in Chapter 3. Secondly, the MIM also captures the procedural information of model based inversion control approach to automatically design the flight control systems in Chapter 4. Thirdly, the intelligent modeling system also automatically generates the control software components based on the results in Chapter 4, which is discussed in Chapter 5. Finally, a complete test case, the design and optimization of multirotor UAV, is illustrated in Chapter 6. It is assumed that top level requirements and design objectives are given by the design experts and simulation experts in the interface of the proposed KBE system. Then, various multiphysics simulation models of the multirotor UAVs are evaluated by simulating complex maneuvers, for which control systems are required, to check the flight performance and flying qualities. An optimization algorithm is also applied to find local or global optimum solutions for the multirotor UAVs. A design framework is established to organize all the design activities presented in Figure 1.8 from the initial definition of top level requirements, modeling of physical entities, design of

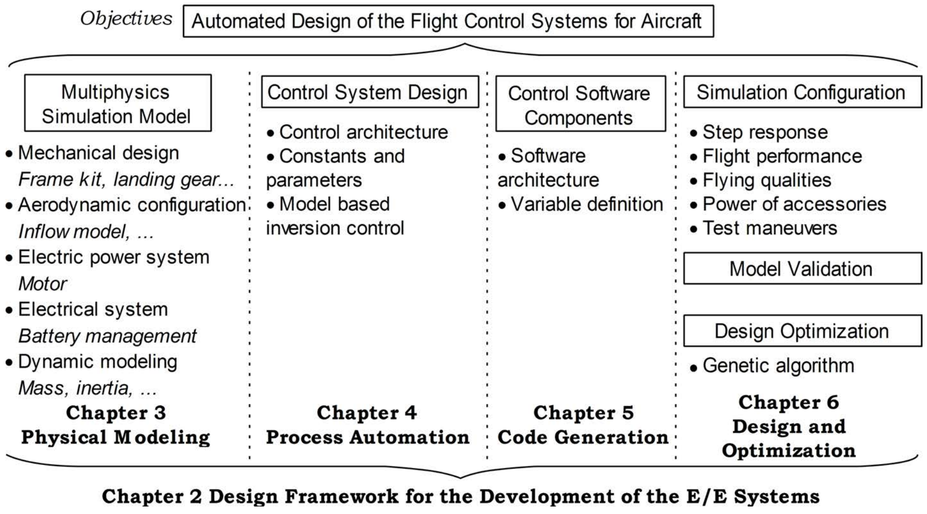

Objectives
Automated Design of the Flight Control Systems for Aircraft
Multiphysics Simulation Model
• Mechanical design
Frame kit, landing gear...
• Aerodynamic configuration
Inflow model, ...
• Electric power system
Motor
• Electrical system
Battery management
• Dynamic modeling
Mass, inertia, ...
Chapter 3 Physical Modeling
Control System Design
• Control architecture
• Constants and parameters
• Model based inversion control
Chapter 4 Process Automation
Control Software Components
• Software architecture
• Variable definition
Chapter 5 Code Generation
Simulation Configuration
• Step response
• Flight performance
• Flying qualities
• Power of accessories
• Test maneuvers
Model Validation
Design Optimization
• Genetic algorithm
Chapter 6 Design and Optimization
Chapter 2 Design Framework for the Development of the E/E Systems

Figure 1.8: Outline of this dissertation

control system, generation of control software components, and finally to the simulation and optimization, which is discussed in Chapter 2.

1.3. OUTLINE

This dissertation is structured as follows. Basic concepts about optimization and the KBE technique are treated in Chapter 2. Then, the design framework of the proposed KBE system is discussed in detail (Chapter 2). Next, details of the method are presented in Chapter 3. Then, the suitability of the new approach is evaluated by automatically generating multiphysics simulation models of a large number of multirotor UAVs design with different topologies in Chapter 4. Later, the proposed KBE system is further developed to automatically generate control software components for the E/E systems in Chapter 5. Finally, the intelligent modeling system is also tested to support a complete design optimization for the multirotor UAVs with significant configuration differences in Chapter 6. A discussion of the results and the associated conclusions are given in Chapter 6 and 7, respectively.

2

KBE to Support Integrated Development of E/E Systems

2.1. A Brief Review of the KBE Approach

2.1.1. Introduction

The MEA and AEA concepts have been proposed to improve reliability, maintainability and supportability while reducing weight, volume and system complexity for the aircraft. Moreover, the design of complex engineering applications, such as aircraft and road vehicles, involves multiple engineering disciplines and subsystems. Because the E/E systems are closely coupled with the other disciplines and have a significant effect on the overall performance and safety of the vehicle. Therefore methods and tools are required to concurrently design the E/E systems together with the other engineering domains. For this reason, new methods and tools to support E/E system development using the KBE technique are proposed in this research study. The KBE approach is a promising method to solve complex, time consuming problems involving multiple engineering domains problems. The multirotor UAV, which is an inherently unstable and unmanned all electric aircraft, is selected as the test case. In order to design the multirotor UAV in an automated way with the possibility to allow large topological/configuration changes, a new design framework is required. In this framework, all the design actives and relevant software modules must be linked together in a coherent way.

A new design framework for E/E system design by using the KBE approach is proposed. A top level overview of this framework is presented in Figure 2.1. Typically, the framework, designated intelligent modeling system, consists of four activities; (1) initialization of design variables (knowledge acquisition), (2)

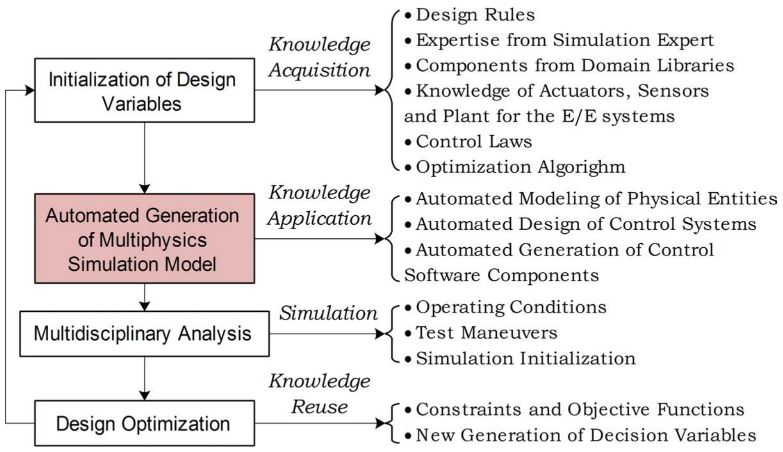

Initialization of Design Variables
Automated Generation of Multiphysics Simulation Model
Multidisciplinary Analysis
Design Optimization
Knowledge Acquisition
Knowledge Application
Simulation
Knowledge Reuse
• Design Rules
• Expertise from Simulation Expert
• Components from Domain Libraries
• Knowledge of Actuators, Sensors and Plant for the E/E systems
• Control Laws
• Optimization Algorighm
• Automated Modeling of Physical Entities
• Automated Design of Control Systems
• Automated Generation of Control Software Components
• Operating Conditions
• Test Maneuvers
• Simulation Initialization
• Constraints and Objective Functions
• New Generation of Decision Variables

Figure 2.1: The design framework for the control system development of electrical and electronic systems (E/E systems)

automated modeling of the E/E systems (knowledge application), (3) multidisciplinary analysis (simulation) and (4) design optimization (knowledge reuse).

Although the focus of this research study is the automated generation of multiphysics simulation models to support the control system development of the E/E systems, it is necessary to explain other phases as well because they provide the necessary inputs and make use of the outputs (results) in the second phase of the process.

2.1.2. Definition of KBE

There are many different KBE definitions in use. Sainter et al. [55] give the definition from the perspective of the generative aspect of KBE systems. They refer to the KBE as "a system can be regarded as a type of knowledge based system that performs tasks related to engineering. KBE systems do not express designs with specific data instances, as ordinary CAD systems do, but with sets of rules that enable the design to apply to large classes of similar parts" [55]. Moreover, Blount et al. [56] view "the strengths of KBE aren't only derived from the capture of design intent but as a true integrator throughout the product introduction process (PIP) supporting the ideals of concurrent engineering". Chapman and Pinfold use the KBE approach to rapidly design and analysis an automotive structure [57]. They state that the KBE approach is "an evolutionary step in CAE and is an engineering method that represents a merging of object oriented programming (OOP) , artificial intelligence (AI) techniques and CAD technologies, giving benefit to customized or variant design automation solution" [57]. Moreover, Verhagen et al. [58] focus on the benefits of KBE systems by "automating repetitive design tasks and achieve significant design time savings, enabling designers to explore a larger part of the design space during the various design phases".

In this research study, the focus is on solving the complexity of E/E systems for the design and optimization of an unstable and unmanned aircraft. Therefore, the definition of KBE in the context of conceptual aircraft design, obtained from La Rocca [59] is most appropriate:

> *Knowledge based engineering (KBE) is a* ***technology*** *based on dedicated* ***software tools*** *called* ***KBE systems****, that are able to capture and reuse product and process engineering knowledge. The main objective of KBE is the reduction of time and costs of product development by means of the following:*
>
> - ***Automation*** *of* ***repetitive, non creative****, design tasks;*
> - ***Support*** *of* ***multidisciplinary design optimization*** *in all the phases of the design process.*

This definition highlights two aspects of the KBE approach. Firstly, the KBE approach is suitable for solving repetitive and non-create design tasks. This is to say, the KBE developer has to keep the boundaries of the design space in mind. Secondly, KBE systems have to provide high fidelity models, which are in most cases geometric models to support for MDO. However, other models, less centred around geometry, may also be needed, for example, the flight dynamics models to evaluate flying qualities and high fidelity flight performance analysis.

2.1.3. BASIC CONCEPTS OF OPTIMIZATION TECHNIQUES AND ALGORITHMS

In order to support for MDO, the KBE system has to be integrated with optimization algorithms. Before introducing the KBE application, it is necessary to have a good understanding of several critical concepts related to optimization techniques and algorithms.

THE MULTI-OBJECTIVE OPTIMIZATION PROBLEM

The multi-objective optimization problem (MOP), which is also called the multi-criteria optimization, multi-performance or vector optimization problem, is defined by Osyczka [6] as the problem of finding:

> *a vector of decision variables which satisfies constraints and optimized a vector function whose elements represent the objective functions. These functions form a mathematical description of*

performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.

Then, the general MOP can be mathematically defined as follows [60]:

Definition 1 (General MOP): *Finding the vector $\vec{x} = [x_1, x_2, ..., x_n]^T$ which satisfies the m inequality constraints:*

$$g_i(\vec{x}) \leqslant 0 \quad i = 1, 2, ..., m \tag{2.1}$$

the p equality constraints

$$h_i(\vec{x}) = 0 \quad i = 1, 2, ..., p \tag{2.2}$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]^T \tag{2.3}$$

where k is the number of objective functions. $\vec{x} = [x_1, x_2, ..., x_n]^T$ is the vector of ***decision variables****, which are the numerical quantities for which values are to be chosen in an optimization problem. The constraints given by equation 2.1 and 2.2 defines the* ***feasible region*** *Ω and any points $\vec{x}$ in Ω defines a* ***feasible solution****.*

GENETIC ALGORITHM BASICS

In this research study, the test case (multirotor UAV) is optimized for two contrary objectives. The design which is determined by many global integer and float parameters significantly changes in the conceptual phase. Thus, this is a mixed-integer multi-objective optimization problem.

The genetic algorithm (GA) is recommended by reference [61] and widely used for solving multidisciplinary problems [62, 63]. Therefore, the GA is selected to generate the values for the decision variables and to evaluate the simulation results with constraints and objective functions. However, it should be pointed out that other optimization algorithms, like hill-climbing or simulated annealing, can also be integrated within the proposed KBE system. Because the focus of this research study is not giving a comparison of optimization algorithms or demonstrating them with applications, interested readers may refer to the references [61, 64] for more details about numerical optimization techniques.

The GA is a stochastic search technique inspired by the mechanism of natural evolution. As is shown in Figure 2.2, the GA starts with a population,

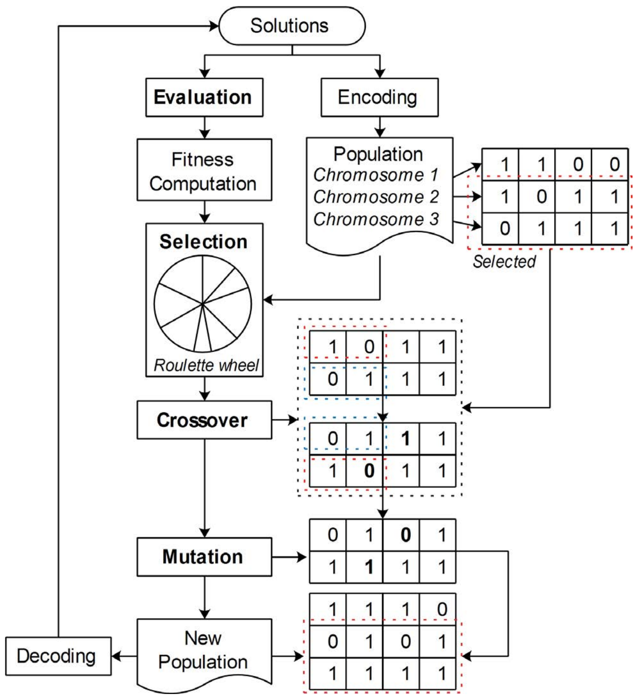


Figure 2.2: The general structure of genetic algorithm (GA)

which is an initial set of random solutions. Every solution in the population is called a chromosome, which evolves through successive iterations, called generations [65].

Every chromosome in one generation needs both an objective and fitness function. The objective function defines the optimality condition (like 2.3) while the fitness function measures how "well" a particular solution satisfies that condition and assigns a corresponding real-value to that solution [66]. For infeasible chromosomes, the most common way is adding a penalty value with their fitness [67, 68], which in turn reduces the opportunity to be selected in further generations.

After the evaluation of the parent population, a new population (also new chromosomes) is produced through sequential procedures, like selection, crossover and mutation [65]. Finally, the new population, which is composed of the chromosomes selected from the parent population (the better fitness, the bigger chance to be selected) and the chromosomes randomly produced, is used to start a new loop.

The GA is different from conventional optimization and search procedures in the following aspects [69]:

- *Genetic algorithms work with a coding of solution set, not the solutions themselves;*
- *Genetic algorithms search from a population of solutions, not a single solution;*
- *Genetic algorithms use payoff information (fitness function), not derivatives or other auxiliary knowledge;*
- *Genetic algorithms use probabilistic transition rules, not deterministic rules.*

2.1.4. DEE FOR AIRCRAFT CONCEPTUAL DESIGN

The concept of the design and engineering engine (DEE), which is a KBE system, has been proposed for aircraft conceptual design by Krakers et al. [70]. The DEE is a modular, loosely integrated design system, able to support the MDO by automating the repetitive and non-creative activities that hamper the design and analysis process [59]. As can be seen in Figure 2.3, the DEE is composed of a set of properly interconnected toolboxes [70].

The DEE starts from the specification of top level requirements. Subsequently, the initiator produces feasible solutions within the format of a set of initial values, which are used for feeding the multi-model generator (MMG) to generate models for multidisciplinary analysis tools. The heart of the DEE is

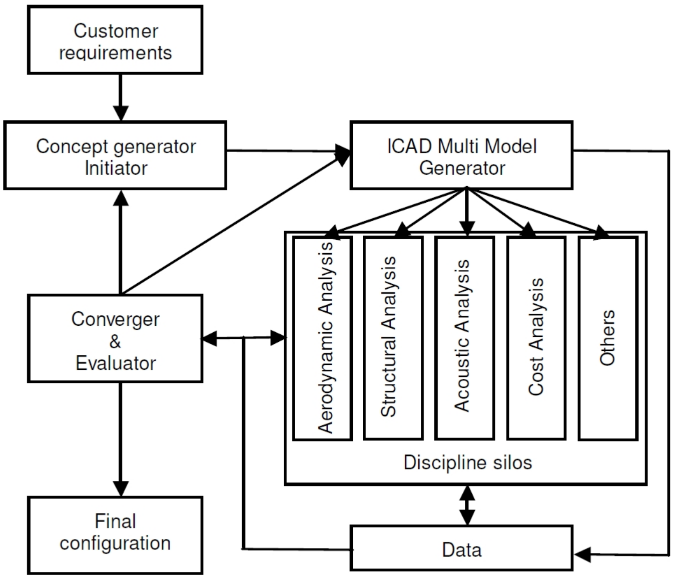


Figure 2.3: Paradigm of a design and engineering engine (DEE) [70]

the MMG, which is a KBE application. The MMG is designed as a report writer. By feeding the MMG with different input values from the initiator, the MMG is able to model different geometry models and configurations' variants. Besides modeling high fidelity geometry, the MMG can also directly create specific data and input files for various analysis tools. Afterwards, the models are analyzed by various multidisciplinary tools and the results are sent to the converger & evaluator of the DEE through the communication framework. The converger & evaluator checks whether the analyzed results are convergent and whether all the requirements are satisfied. Next, the converger & evaluator generates new values to feed the MMG and starts a new analysis loop. If it is impossible to fulfill all the requirements needed to arrive at a design solution, the initiator is called again to produce a different aircraft configuration [59].

2.2. An Intelligent Modeling System for E/E System Design

The concept of the DEE has been used for solving various engineering problems, such as the design of a blended wing body [71], fertilization of a structural wing [72], parametric modeling of movables for aircraft [73], design trade-offs for fiber composite fuselages [74], etc. The DEE has been proven useful for aircraft and automotive conceptual design [33, 53]. These research studies mainly focus on geometric manipulations for structural design, aerodynamic analysis and cost estimation. So far, the E/E systems have not been considered in the DEE.

In the current research study the DEE is extended with the capabilities to; (1) generate multiphysics simulation models for analysis of mechatronic systems, and (2) design and analyse control systems based the multiphysics simulation models, including the development and testing of the associated software algorithms. Therefore, this research study proposes an intelligent modeling system to automatically generate multiphysics simulation models to support the MDO process by using the KBE approach. A more detailed structure is expended in Figure 2.4.

The intelligent modeling system inherits several module names from the original DEE but with different emphases, which are listed as followed:

- The most difference between the original DEE and the intelligent modeling system is the model generation. Instead of geometric modeling, the intelligent modeling system focuses on the physical modeling of the E/E systems in mechanical, dynamics and multiple domains, automated design of control system and automated generation of control software components;

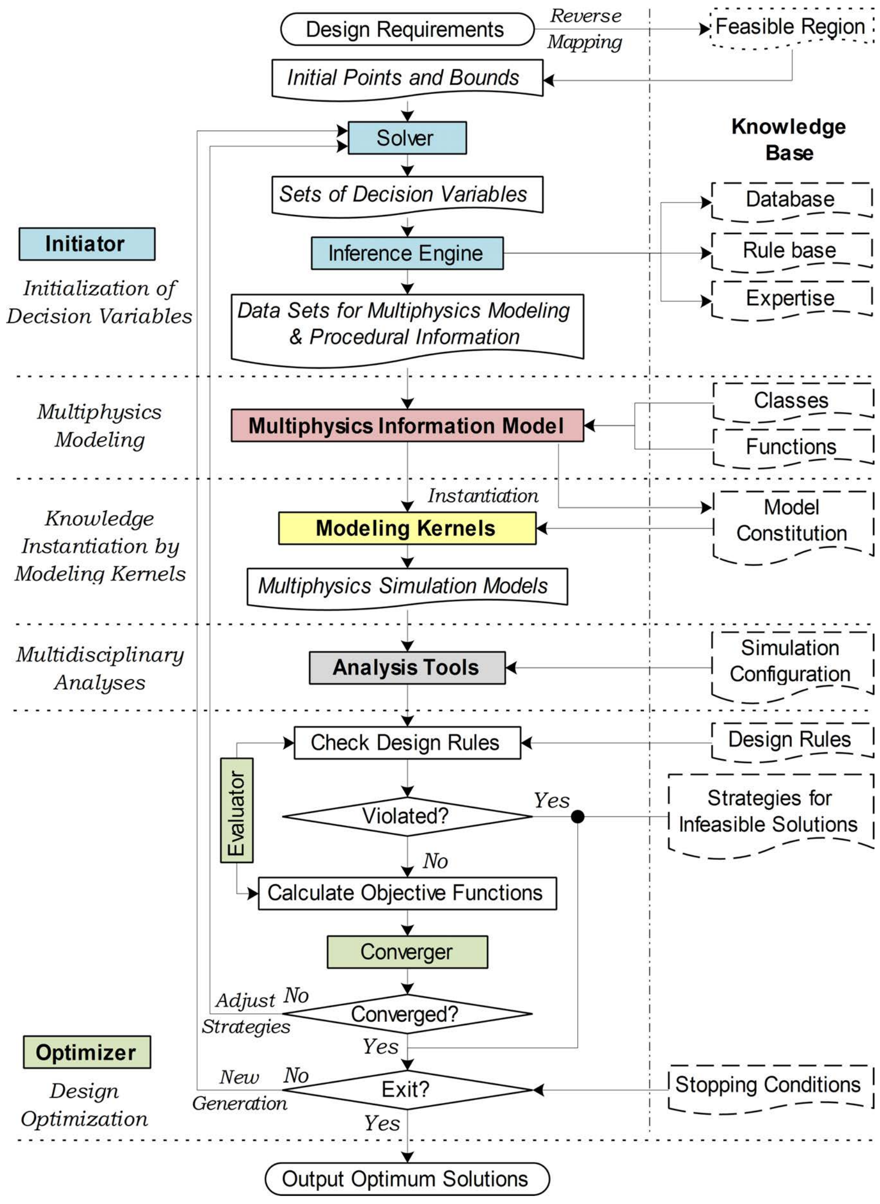


Figure 2.4: Flow diagram of an intelligent modeling system

- In order to keep the consistency for different disciplines, the concept of a MIM is proposed. The MIM only contains the intrinsic properties of physical systems by the KBE approach and a specific format of representation is avoided. This feature is quite different from the MMG where the information of the physical systems is coupled with the instantiation method;

- By capturing the expertise from simulation experts, it is able to construct the simulation model in an automated fashion, which in turn accelerates the MDO process in all domains and analysis tools. Thus, the intelligent modeling system makes it possible to establish a concurrent design environment to support the MDO process.

The intelligent modeling system starts from specification of initial points and corresponding bounds by reversely mapping the design requirements with the feasible objective region $f(\Omega)$. Then, the solver of the initiator generates initial sets of decision variables based on the initial points and bounds. Next, some decision variables are used to select the components from multidisciplinary libraries by the inference engine of the initiator while others can be used to instantiate the MIM of the complex systems. The MIM is instantiated by modeling kernels. Which specific modeling kernel is applied by the intelligent modeling system is determined by the target data (or file) format and the analysis tool. The resulting multiphysics simulation models are simulated in analysis tools. Subsequently, the evaluator of the optimizer checks the analysis results with design rules. Feasible solutions are selected to calculate the objective functions and checked for the convergence sequentially. If the feasible solutions aren't converging, the process will go back to the initiator to adjust the sets of decision variables with strategies. However, if some solutions are violated in previous rule checking, the infeasible ones will skip these steps and be deleted by the optimization algorithms. Next, new sets of decision variables are created along with the beginning of a new loop. The optimization process is terminated until the stopping condition is satisfied. Finally, the intelligent modeling system will output the optimum solutions.

The intelligent modeling system is programmed with the genworks general-purpose, declarative, language (GDL) , which is based on the ANSI standard version of Common LISP. The GDL is particularly effective at representing complex systems, including three-dimensional geometry models and design process [75].

2.2.1. INITIALIZATION OF DECISION VARIABLES

The first module of the intelligent modeling system is initiator. The initiator accepts the design requirements from the engineers and produces the inputs for the MIM. In principle, there are two modules in the initiator: solver and inference engine, which are illustrated in Figure 2.5.

Firstly, in order to determine the initial points and bounds for the decision variables, the simulation experts specify the top level requirements which are

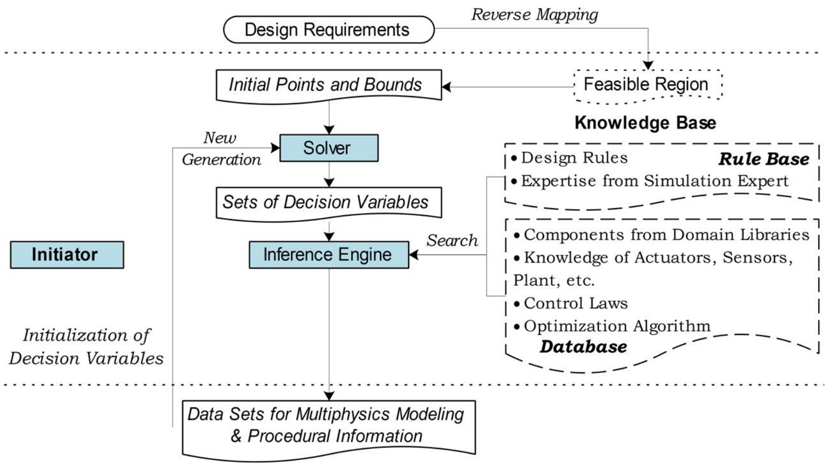

Design Requirements
Reverse Mapping
Initial Points and Bounds
Feasible Region
Knowledge Base
New Generation
Solver
• Design Rules
Rule Base
• Expertise from Simulation Expert
Sets of Decision Variables
Search
Initiator
Inference Engine
• Components from Domain Libraries
• Knowledge of Actuators, Sensors, Plant, etc.
• Control Laws
• Optimization Algorithm
Database
Initialization of Decision Variables
Data Sets for Multiphysics Modeling & Procedural Information

Figure 2.5: Flow diagram of the initiator

mapped with the feasible objective region $f(\Omega)$. Secondly, the solver creates the sets of decision variables with the GA. Thirdly, the inference engine selects proper components based on the sets of decision variables for subsequent modules.

In this research study, a novel methodology is proposed to specify the initial points and bounds by automating the design process. It is better to discuss it later because a complete design process is required. Thus, despite the fact that the first step is the specification of initial points and bounds, it will be discussed lastly. The solver and inference engine are explained in this section.

SOLVER

The solver accepts the initial points and bounds and produces the sets of decision variables. Take the GA for example, the solver creates the generations of the population for the decision variables through encoding, selection, crossover and mutation.

Firstly, as can be seen in Figure 2.6, the solver generates the initial sets of random binary values (initial population) with the inputs of number of chromosomes, length of each chromosome (the precision) and number of decision variables. The binary values are encoded into decimal values, which are sent to the inference engine to instantiate the design. There are many encoding methods for specific problems, such as real number coding for constrained optimization problems [76] and integer coding for combinatorial optimization problem [77].

Secondly, some chromosomes are selected from the initial population due to

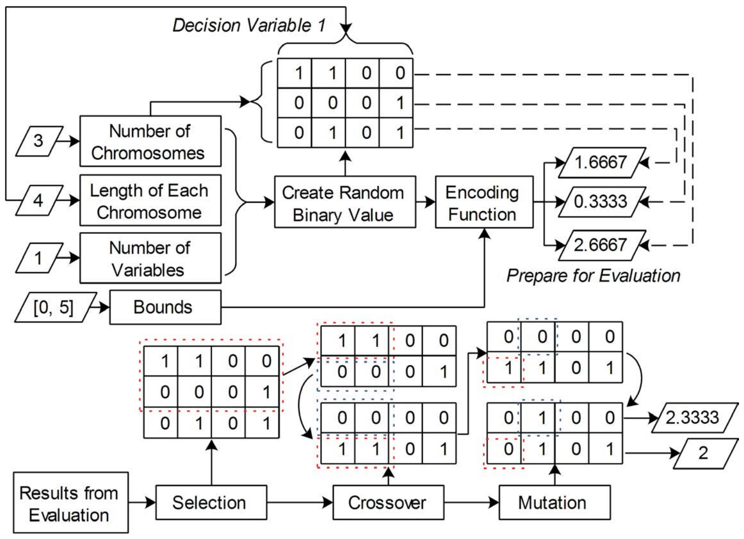


Figure 2.6: Generation of chromosomes by the GA

higher fitness values than the others [1]. Rogers and Prugel-Bennett [78] propose a method for selection schemes by changing population fitness variance. Schaffer [79] recommends a vector evaluated genetic algorithms (VEGA) as an example of criterion selection technique. Ishibuchi and Murata [80] suggest a method of weighted sum for aggregation selection technique.

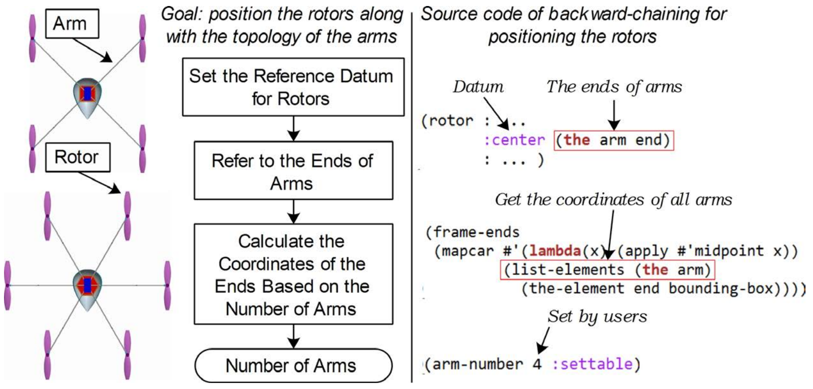
Thirdly, the selected chromosomes are crossed each other at specific point. As illustrated in Figure 2.6, two chromosomes are exchanged at the second position. It is possible that there are multiple crossover points, which is a complicated problem. Gen and Cheng [81] recommend the methods of position based crossover. Syswerda [82] proposes order based crossover operator, which is a slight variation of the position based crossover. More methods related to crossover are compared by Amadori et al. [62].

The final step is mutation, where a few randomly chose bits are changed from 1 to 0 or reversely. Thierens [83] recommends an adaptive mutation rate control schemes for the GA, which is compared with self-adaptive parameter control [84]. Then, new population is generated by the solver and served for the inference engine again.

INFERENCE ENGINE

The principal role of the inference engine is to search for the most appropriate item of knowledge to apply at any given moment [61]. The inference engine

[1]The computing of fitness functions is performed in the optimizer.


Arm
Rotor
Goal: position the rotors along with the topology of the arms
Set the Reference Datum for Rotors
Refer to the Ends of Arms
Calculate the Coordinates of the Ends Based on the Number of Arms
Number of Arms
Source code of backward-chaining for positioning the rotors
Datum
The ends of arms
Get the coordinates of all arms
Set by users

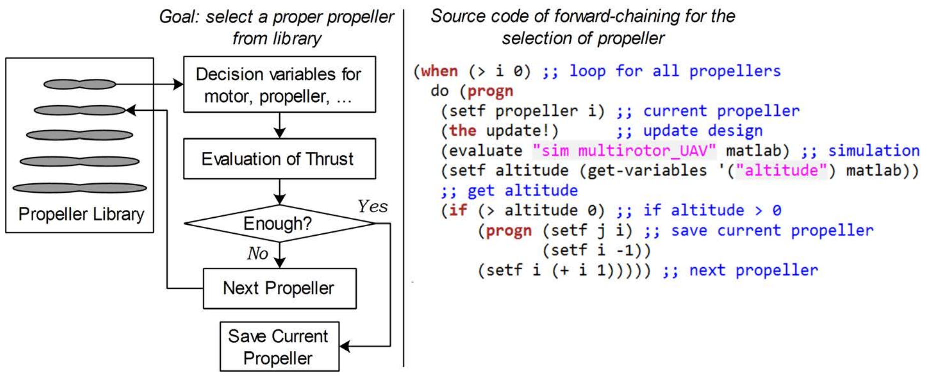
```
(rotor : ..
       :center (the arm end)
       : ... )

(frame-ends
  (mapcar #'(lambda(x) (apply #'midpoint x))
          (list-elements (the arm)
            (the-element end bounding-box))))

(arm-number 4 :settable)
```

Figure 2.7: Example of backward-chaining for positioning the rotors of multirotor UAVs

searches both the rule base and database to provide the inputs for subsequent modules in terms of the design rules, expertise from simulation experts, components, control laws, optimization algorithms, etc. There are two kinds of search strategies: forward-chaining and backward-chaining. Both of them are integrated within the inference engine and used for different cases.

Backward-chaining The backward-chaining starts from the goal and reversely selects the necessary rules for examination [85]. As for the inference engine, the backward-chaining is usually used for collecting extra information to position the physical entities. As is shown in Figure 2.7, the position of the rotors is referred to the arm ends of the multirotor UAV. The inference engine can record the recursive processes by programming and then automatically update the position of the rotors when the simulation expert changes the number of the arms. The backward-chaining is widely used for the cases which need extra information to specify the attributes for the models.

Forward-chaining The forward-chaining takes the available information and generates as many derived facts as it can [61]. It is applied when the selection of components requires the feedback from modeling and evaluation, which is a kind of local iteration. Continued with previous example, although the solver can initialize a random value for selecting the propeller, it is necessary to check whether the thrust is enough for current configuration of the multirotor UAV (see Figure 2.8). If yes, current propeller is saved. Otherwise, the inference engine will select a bigger propeller than the previous one and perform a new evaluation. In this example, it is possible to try all propellers if none of them is suitable for the current configuration.


Goal: select a proper propeller from library
Propeller Library
Decision variables for motor, propeller, ...
Evaluation of Thrust
Enough?
Yes
No
Next Propeller
Save Current Propeller
Source code of forward-chaining for the selection of propeller

```
(when (> i 0) ;; loop for all propellers
  do (progn
   (setf propeller i) ;; current propeller
   (the update!)      ;; update design
   (evaluate "sim multirotor_UAV" matlab) ;; simulation
   (setf altitude (get-variables '("altitude") matlab))
   ;; get altitude
   (if (> altitude 0) ;; if altitude > 0
       (progn (setf j i) ;; save current propeller
              (setf i -1))
       (setf i (+ i 1))))) ;; next propeller
```

Figure 2.8: Example of forward-chaining for specifying the propellers

2.2.2. MULTIPHYSICS MODELING AND MODEL INSTANTIATION

The basic principle of the MIM is modeling the physical entities into classes with properties, which is illustrated in Figure 2.9. Every time when the initiator creates the data sets and feed them into the MIM, the classes are instantiated into various models which aren't limited in geometries but also dynamics, aerodynamics, control systems, software code and multiple domains. Moreover, the MIM also uses functions to capture the non-physical information. The functions can be instantiated as script file to represent the process information. The detailed methods are discussed in Chapter 3, 4 and 5 for multiphysics simulation modeling, automated design of control system and automated generation of software code, respectively.

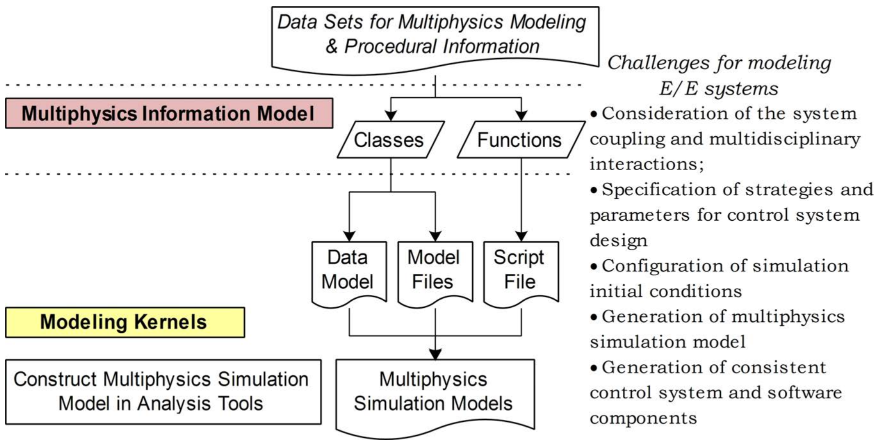

Data Sets for Multiphysics Modeling & Procedural Information
Challenges for modeling E/E systems
Multiphysics Information Model
Classes
Functions
• Consideration of the system coupling and multidisciplinary interactions;
• Specification of strategies and parameters for control system design
• Configuration of simulation initial conditions
• Generation of multiphysics simulation model
• Generation of consistent control system and software components
Data Model
Model Files
Script File
Modeling Kernels
Construct Multiphysics Simulation Model in Analysis Tools
Multiphysics Simulation Models

Figure 2.9: Multiphysics modeling and model instantiation

2.2.3. MULTIDISCIPLINARY ANALYSIS

The resulting multiphysics simulation models (automatically) are evaluated in analysis tools, such as MATLAB or Modelica. The inference engine of the initiator specifies the simulation configurations for the evaluation of the E/E systems, such as a specific flight maneuvers which the aircraft must be able to conduct. In this research study, the multiphysics simulation models are developed in MATLAB. Thus, a connection between the intelligent modeling system and MATLAB is required for communication (the communication framework - Section 2.2.5).

2.2.4. DESIGN OPTIMIZATION BY GA

The optimizer module checks the feasibility of each design solution and convergence. As can be seen in Figure 2.10, the optimizer is composed of two sub modules: the evaluator and the converger. The evaluator first checks whether the design rules are satisfied based on the analysis results of a design solution. Next, the objective function is calculated. For feasible solutions the converger checks whether the results are converging. Infeasible solutions will not go through this evaluation process and will be deleted in further generations of the GA. Finally, a new generation of decision variables is produced by the solver. The overall process is iterated until the conditions to terminate the process are satisfied.

Obviously, the optimization method and settings have an effect on the final

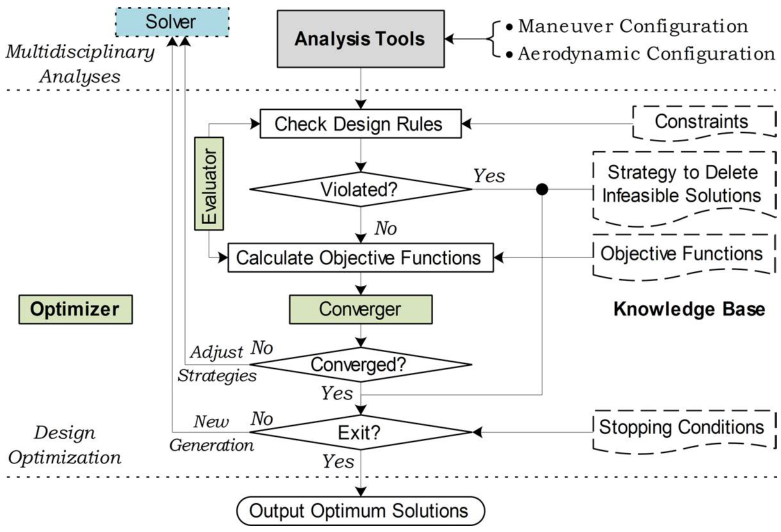


Figure 2.10: The workflow of the design optimization in the intelligent modeling system

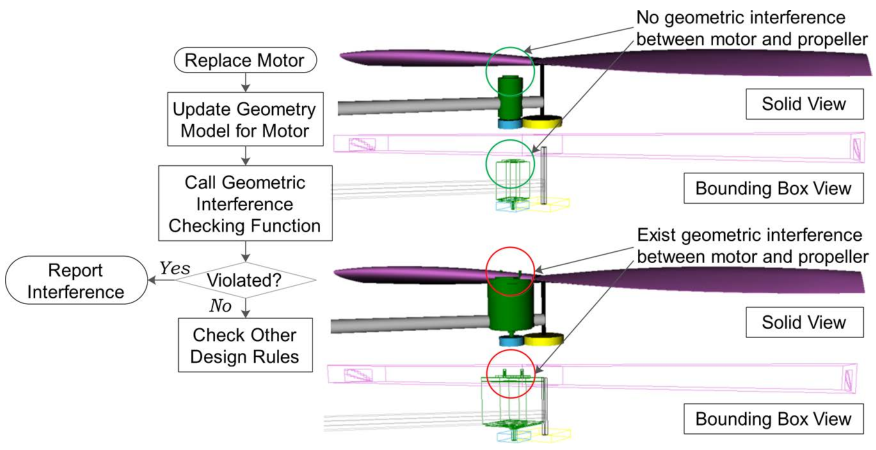

Replace Motor
Update Geometry Model for Motor
Call Geometric Interference Checking Function
Violated?
Yes
No
Report Interference
Check Other Design Rules
No geometric interference between motor and propeller
Solid View
Bounding Box View
Exist geometric interference between motor and propeller
Solid View
Bounding Box View

Figure 2.11: Example of geometric interference checking for arm assembly

results. However, the optimization process is aimed to test whether the proposed KBE system can automate the whole design process (mechanical design, dynamics and E/E systems) for a mechatronic product and whether a consistent control system is ensured for the whole design envelope instead of choosing the best optimization strategy and method. Therefore, reliable methods are chosen which can deal with constraints (infeasible solutions), selecting solutions and stopping conditions.

EVALUATOR

Design rules and objective functions specified by the inference engine of the initiator are defined as constraints in the evaluator.

Design Rules These define the criteria that have to be satisfied in order to achieve a specific task. As is shown in Figure 2.11, a geometric interference checking is considered as example. The arm assembly is composed of an arm, a motor and a rotor. The motor is selected from a database provided by manufacture. During the optimization process, the decision variable for the motor (index number) may significantly changes. Therefore, it is possible that the bottom of the motor may touch with the rotor, resulting an infeasible configuration.

Every time when the motor is replaced, the intelligent modeling system will firstly update the geometry model of the arm assembly and then the evaluator will call the function for geometric inference checking. In order to accelerate computing process, the bounding boxes of the geometries are input into the function instead of complex surfaces. If some configurations are violated, the evaluator will report error messages to the engineers, otherwise, it go through other design rules.

Infeasible Solutions These should be avoided in the final set of (optimum) results. How to deal with feasible and infeasible solutions in a GA is referred to as constraint-handling. Richardson et al. [86] propose to add constraints as weighted penalty functions to each MOP. Other researchers suggest to repair each infeasible solution in order to make it feasible [87, 88]. In this research study, the method of adding penalties is selected. The infeasible solution will be assigned a zero fitness.

Objective Functions and Selection Similar to the evaluation of the design rules, the objective functions are also computed for each design solution in every loop. A fitness function is needed for the GA to show the differences among the chromosomes. There are different methods for computing fitness values, such as scaling methods of Goldberg [69] and the linear-ranking algorithm of Baker [89]. The strategy of ranking selection is chosen to evaluate the fitness of every solution. The advantage of this method is that all the solutions have a chance to be selected. In order to counteract the low convergence of the rank selection, three best solutions of the current generation are kept into new generation.

CONVERGER

The converger is applied for checking the convergence of the feasible solutions. Whether the optimization results are convergent or not can be determined by the following three convergence criteria [90]:

$$\|\vec{x}_i - \vec{x}_{i-1}\| < \varepsilon_1 \tag{2.4}$$

$$\|\nabla f(\vec{x}_i)\| < \varepsilon_2 \tag{2.5}$$

$$|f(\vec{x}_i) - f(\vec{x}_{i-1})| < \varepsilon_3 \tag{2.6}$$

Equation 2.4 shows that the decision variables are almost constant. The objective functions are also not changing any more (Equation 2.5). The distance between the objective functions for the decision variable $\vec{x}_i$ and $\vec{x}_{i-1}$ are smaller than an arbitrarily small positive quantity (Equation 2.6).

For the MOPs, the objective is to find a group of feasible solutions instead of a single one. A Pareto front is typically used to specify the group of feasible solutions. Thus, it is usually try to specify a Pareto front where exists no feasible vector $\vec{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion, then the $\vec{x}$ is the Pareto optimal [60].

The rate of convergence depends on both the optimization algorithms used and the problem under investigation. For GA's, elitist selection is a useful strategy to accelerate the rate of convergence [69]. Louis and Rawlins [91] discuss various methods for predicting convergence time of the GA.

STOPPING CONDITIONS

The optimization process of GA is stopped when a specific condition is satisfied, such as the number of maximum generations, time limit, convergence, etc. In this research study, the optimization process is set to stop at a maximum number of generations.

2.2.5. PROCESS AUTOMATION

The benefits of the KBE approach can only be exploited when the processes described in the previous sections are automatically completed. Therefore, a communication framework is developed to allow communication between the multidisciplinary analysis tools and other elements of the framework. In addition to the exchange of data and information, the communication framework can also directly perform design activities within the distributed software tools. In order to set up the connections, the inference engine is requested to prepare the necessary knowledge for the communication framework. such as target multidisciplinary tools, ports, paths, etc. Several key aspects are discussed below.

A communication framework which connects the intelligent modeling system with MATLAB has been created [33, 92] based on the common lisp interface to MATLAB, originally developed by Carlos Ungil [93]. With this communication framework, the intelligent modeling system can call the GA in MATLAB to initialize sets of decision variables. Next, the values are fed back from MATLAB to the MIM for modeling the physical systems. Next, the multiphysics simulation model is created in MATLAB again and test maneuvers are simulated. Finally, the intelligent modeling system obtains the analysis results from the MATLAB simulations and generates a new generation of decision variables. Summarizing, design automation is achieved between the intelligent modeling system and MATLAB by using the communication framework.

DATA FORMAT

In order to connect with the target analysis tool, the intelligent modeling system should be able to create output files in common data formats. The data format should be recognized and supported by most analysis tools. In this research study, a geometric model is written as a STEP file, which is the most widely accepted format by CAD tools [94]. Compared to other possible data formats for

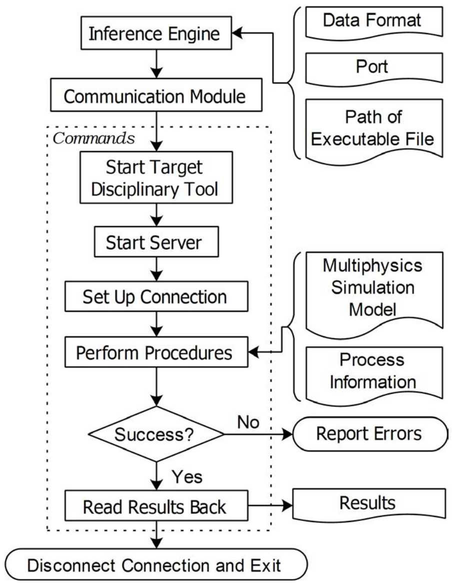


Figure 2.12: The procedures to communicate with analysis tools by the communication framework

geometry, such as IGES and VRML, the STEP file is more comprehensive in terms of hierarchy and assembly references. Moreover, the values of decision variables are written in a comma-separated values (CSV) file, which is a common format used by many programs. CSV is a suitable format to store vectors and matrices. Other formats, like XML, which are also supported by the intelligent modeling system aren't discussed in detail.

Port, Server and Path

As can be seen in Figure 2.12, the inference engine has to specify the port of target analysis tool to avoid conflicts with other programs. The communication framework also needs a server as a bridge to connect the intelligent modeling system with the target software. For compatibility reasons, the server is usually programmed with the same computer language as the target tool. Moreover, the path for the executable file of the target tool is also required by the communication framework.

Workflow

After the knowledge collection by the inference engine, communication framework starts the target analysis tool with the commands of operating system. Then, it sets up a connection through the common server between the intelligent modeling system and the analysis tool. Next, the communication framework imports the multiphysics simulation models produced by the MIM and carries out sequential commands in the target analysis tool, such as a complete simulation. Finally, if all procedures are successfully performed, the communication framework disconnects the connection and reads the results back to the intelligent modeling system. Otherwise, an error message will be sent to the engineers.

After the knowledge collection by the inference engine. the communication framework starts the target analysis tool by means of operating system commands. Then, a connection is set up through the common server between the intelligent modeling system and the analysis tool. Next, the communication framework imports the multiphysics information model instantiated from the MIM and carries out sequential commands in the target analysis tool, such as a complete simulation of a maneuver. Finally, if all procedures are successfully performed, the communication framework sends the results back to the intelligent modeling system and disconnects. Otherwise, an error message is created.

2.2.6. FROM REFERENCE DESIGN TO CREATIVE DESIGN

Conceptual design is a challenging process for complex engineering products, such as aircraft, helicopters and road vehicles, because there is a great deal of uncertainty between the decision variables and the final performance of the vehicle. Traditionally, text books suggest the engineer to refer to one or several existing designs after the specification of requirements, especially in the conceptual stage [95, 96]. The methodology of referring to existing solutions in the initial stage of the design process is designated reference design (RD) in this research (Figure 2.13).

The apparent advantage of the RD approach is that engineers can estimate

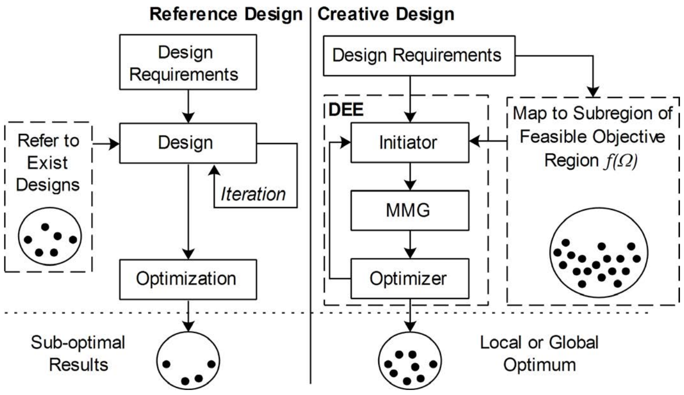


Figure 2.13: Comparison of reference design (RD) and creative design (CD) approaches

the performance of the intended design based on existing solutions. The values of the decision variables can also be approximately determined which in turn minimizes the uncertainty of the design. Finally, the design will be fixed after one or more iterations. Nevertheless, the inherent defect of RD approach is that it limits the creativity of the designer and thereby the design space. Moreover, even if there are iterations inside the RD method, it is not developed for the framework of the MDO used today which integrates various analysis tools. Whenever existing designs aren't suitable for a new problem, it is possible that the final result is a suboptimal solution instead of a local optimum or even the global optimum.

The reason for using the RD approach is that it is difficult to determine what the design should be at the early stage. However, this problem can theoretically be tackled by the proposed KBE system. As indicated above, if the complete design process can be automated by the intelligent modeling system, it is possible to build a subregion of a feasible objective region $f(\Omega)$. When a new product has to be developed, the new set of requirements can be mapped on the subregion and reversely, the values for the decision variables can be found.

The most comprehensive method to specify a complete feasible objective region $f(\Omega)$ is by enumerative scheme. However, the enumerative scheme is inefficient especially when there are many decision variables. Many optimization algorithms (like the GA or particle swarm optimization (PSO)) can search for the feasible objective region $f(\Omega)$ by a group of decision variables. This is considered much more efficient than the enumerative scheme. If there is not time limitation, it is possible to build a subregion of the feasible objective region $f(\Omega)$. The GA is selected again as example. First, the initiator randomly generates sets of decision variables for modeling the physical system, which is shown in Figure 2.14. Second, the design solutions are evaluated by the evaluator and divided into two groups. The fitness values of the feasible solutions are set to one while the infeasible ones are given zeros. Third, due to their higher fitness values than the infeasible ones, the feasible solutions are saved in every generation, which constitutes the subregion of the search space. In the meantime, the objective functions can be computed by feeding the feasible solutions, resulting the subregion of the feasible objective region $f(\Omega)$. Because the initial population is randomly set by the algorithm, innovative new solutions are created. Hence this process is named creative design (CD) .

The objective of the creative design (CD) approach is to give the engineer a comprehensive view of the problem at hand and to specify the starting point for the optimization process, which is the first phase of the intelligent modeling system (the specification of initial points and bounds in Figure 2.4). Unlike the RD method which can only provide a limited solution set, the CD approach is

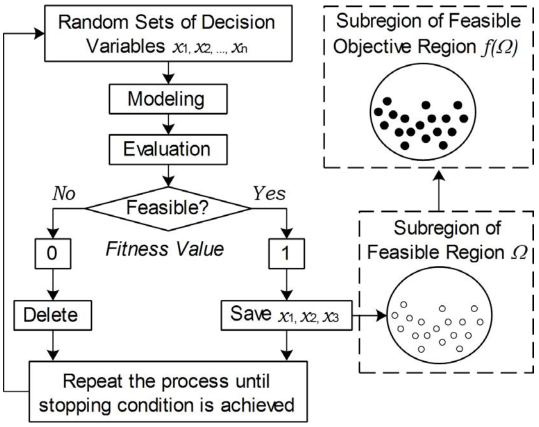

Random Sets of Decision Variables $x_1, x_2, ..., x_n$
Modeling
Evaluation
No
Feasible?
Yes
0
Fitness Value
1
Delete
Save x_1, x_2, x_3
Repeat the process until stopping condition is achieved
Subregion of Feasible Objective Region $f(\Omega)$
Subregion of Feasible Region Ω

Figure 2.14: Construction of feasible objective region $f(\Omega)$ by the knowledge based engineering (KBE) system and the creative design (CD) approach

able to determine all feasible solutions in the early design phases. The feasible solution set is limited by the bounds of the decision variables specified by the designer. The optimization process can start from these novel solutions and potentially find more arbitrary local optima than the RD method.

This approach also has disadvantages. First of all, it takes much time to construct a comprehensive feasible objective region $f(\Omega)$ regardless of which algorithms/strategies are applied. Because the KBE system can automatically iterate the design processes, it is possible to compute at least a subregion of the complete feasible objective region $f(\Omega)$, which there is no limitation on the available computation power and time. The CD approach can be applied on the subregion for designing a new product. Second, if a new design is proposed which is put in the feasible objective region $f(\Omega)$ but outside the subregion that is already computed, it may be also time consuming to extend the subregion to include the new design.

2.3. SUMMARY

New methods and tools are required to concurrently design the E/E systems together with the other engineering domains. The multirotor UAV, which is an inherently unstable and unmanned all electric aircraft, is selected as test case. Because the development of this vehicle involves several disciplines and many subtasks, a novel design framework is proposed to enable MDO for aircraft (and other vehicles) with a high level of E/E systems integration. The design framework is an extension to the so-called design and engineering engine (DEE). The new intelligent modeling system is composed of elements for knowledge acquisition, knowledge application, simulation and knowledge reuse. A definition of KBE is given in this chapter along with a treatment of basic concepts related to MDO.

Next, the structure of the original DEE is discussed. a new intelligent modeling system is proposed especially for the integrated development of E/E systems in the early design phases of an aircraft. The new intelligent modeling system is different from the original DEE mainly in several aspects. All modules of the new intelligent modeling system are discussed in detail except for the concept of the MIM, which is responsible for modeling the physical systems, control systems and control software components. Unlike the original DEE designed for aircraft conceptual design, the new focuses more on the multiphysics modeling aspects of the physical system, design of the control system and generation of software code for the E/E systems.

3 GENERATION OF MULTIPHYSICS SIMULATION MODEL TO SUPPORT MDO

3.1. INTRODUCTION

THE KBE approach and corresponding software modules, the DEE, are discussed in the previous chapter except for the model generation. Because the E/E systems should be concurrently developed with other engineering domains, the concept of a MIM is proposed in the current investigation to enable the integration of design and simulation knowledge from multiple engineering domains and to enable the automatic generation of multiphysics simulation models for MDO. To achieve this, several challenges must be addressed. These are discussed below.

Although most disciplinary analyses require geometric information as input, others such as the E/E systems, require non-geometric information, like the control architecture, to evaluate the performance of the engineering application. Moreover, simulation experts have to specify the initial condition, environmental conditions, test maneuvers and other configurations, such as the numerical solver required for the simulation model. These repetitive design and simulation activities should also be captured by the KBE systems to accelerate the MDO process. In other words, only geometric modeling is not enough to support the MDO process. The KBE systems should include the design and simulation knowledge from multiple domains and provide the models with the right level of fidelity to serve for multidisciplinary analysis purposes.

Typically, design knowledge from multiple engineering domains are coupled. Moreover, in order to model the complex systems, the simulation experts have to select proper components (or technologies) from potentially a vast number of simulation libraries and ensure that submodels are compatible. Thus, the KBE systems should not only solve the problem of system coupling but also take the interactions among the disciplines (or components) into account when modeling the complex systems.

Furthermore, in order to fully support the use of analysis tools, the KBE systems should be able to output not only geometric files but also other data formats, like script files, in an efficient way. For instance, if the configuration of the complex system changes significantly in the early design phases, the control software components of the E/E systems which are usually represented by source code should be updated to stay consistent with the geometry and configuration of the physical plant.

Various research studies [97–99] have focused on the use of mono disciplinary analysis tools as part of an MDO framework. However, little research has been performed on the use of multiphysics simulation models as part of an MDO framework. As discussed in Chapter 1, some researchers propose methods and tools for mechatronic design which is a multidisciplinary approach. Nevertheless, the proposed methods and tools either emphasize too much on electric control systems or on mechanical aspects.

Therefore, an intelligent modeling system is proposed to automatically generate the multiphysics simulation model for the MDO problem including the E/E systems. The challenge is how design rules, constraints, requirements and interactions from the multiple engineering domains are taken into account and integrated within the model generation. This challenge consists of three key elements:

- Consideration of the system coupling and interactions among the disciplines during the model generation;
- Representation of design knowledge from multiple domains in an unified format to ensure consistency among the disciplines;
- Generation of the multiphysics simulation model with right level of fidelity to serve the analyses in multiple engineering domains.

This chapter is structured as follows. First, a review of model generation methods to support MDO is given. These methods are compared to the traditional design process and to the intelligent modeling system proposed in this research study. How the intelligent modeling system tackles the three challenges presented above is discussed in more detail in the subsequent sections. At the end, the multirotor UAV test case is used to demonstrate the capabilities of the intelligent modeling system.

3.2. A Brief Review of Model Generation to Support MDO

In order to support MDO for complex engineering systems, a different design representation must be created for each separate disciplinary analysis. Through a literature review, it has been found that the different design representations typically evolve when new requirements emerge during the design and analysis process.

Modeling the physical systems across multiple domains is required for the MDO process. The common practice for designing robots, airplanes and ground vehicles is a sequential approach, where the mechanical subsystems are designed prior to other subsystems [100]. Nevertheless, as a complex system, the mechanical design cannot be optimized without taking into account its influence upon other subsystems, like the controls [101]. Therefore, various modeling methods for supporting mechatronic design which is a multidisciplinary approach are proposed in the literature.

As can be seen in Figure 3.1, mathematic equations are the most direct way to describe the physical system across multiple domains. For example, He and McPhee [100] use equations to model the dynamics of a half-vehicle. A GA is applied to simultaneously optimize relevant parameters for the dynamics model and control systems. Moreover, da Silva et al. [42] represent the dynamics behavior of a pick-and-place assembly robot using flexible multibody dynamics. The flexible multibody system is modeled using mathematical equations. Control system design is based on these equations as well. Furthermore, other researchers use an evaluation model called MDQ to facilitate decision-making in the design process [43]. It is claimed that the controller design issues and parameters are treated simultaneously with other physical issues and parameters [43]. However, if the complex systems are modeled by mathematical equations, it is not straightforward to see the physical meaning behind them.

Bond graphs were proposed by Paynter [102] at 1959. The bond graph approach starts by taking into account the energy flows between the ports of the (actual and conceptual) components of an engineering system instead of establishing and reformulating mathematical equations [103]. The advantage of the bond graph methodology is that the complex engineering (mechatronic) system can be represented in a compact and explicit way. Margolis and Shim [104] apply the bond graph methodology to generate a four-wheel, non-linear vehicle dynamics simulation model with electrically controlled brakes and

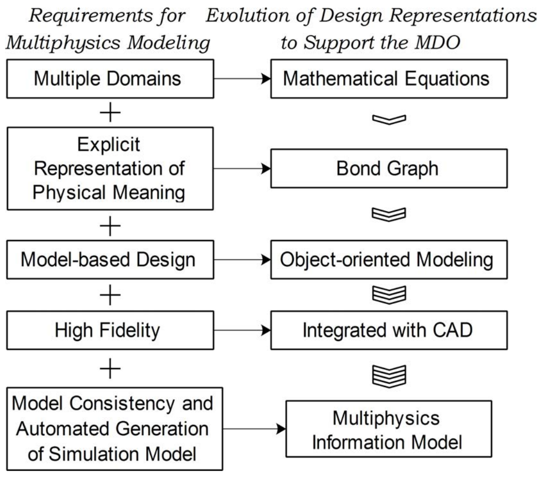

Requirements for Multiphysics Modeling
Evolution of Design Representations to Support the MDO
Multiple Domains
Mathematical Equations
Explicit Representation of Physical Meaning
Bond Graph
Model-based Design
Object-oriented Modeling
High Fidelity
Integrated with CAD
Model Consistency and Automated Generation of Simulation Model
Multiphysics Information Model

Figure 3.1: Requirements for multiphysics modeling and evolution of modeling techniques

steering as well as stability control at each suspension corner. Submodels are first modeled and then assembled into a computable overall model. Moreover, Granda [44] explored the bond graph technique and developed a software tool, named as CAMP-G, to create state space models for serving simulations in MATLAB Simulink. Commercial software packages, like 20-sim, also support the domain independent bond graph notation for modeling dynamics systems [45].

Subsequently, model-based system engineering (MBSE) is widely accepted as a useful approach for designing complex systems [105]. Paredis et al. [106] propose a rapidly deployable manipulator system which combines the flexibility of reconfigurable modular hardware, modular control software and an agent-based design framework. By combining a wide range of general purpose hardware and software modules, a wide range of robotic systems can be assembled. This is just one of many applications using the modular design approach.

The core of the MBSE, modular modeling, is also listed by researchers as a major requirement for the modeling language and the simulation environment [107]. Modularization of complex systems, like mechatronic systems, allows to establish algebraic relations (constraints) among the design variables and splits a large model in a number of submodels for model and software reuse [107]. In order to enable the modular modeling, object-oriented modeling is introduced. The physical entities, like a resistor in electrical engineering or a pump in hydraulic engineering, can be defined as standard components (classes) in each domain library. The engineer is expected to directly operate the components from the multidisciplinary libraries to quickly construct the complex systems and their variants. Piela et al. [108] introduce an object-oriented modeling language for modeling and analysis, which is called the advanced system for computations in engineering design (ASCEND). It allows the user to work with both the high-level abstractions and underlying structure (mathematical equations) if necessary [108]. After the development of ASCEND, a more widely used physical modeling language, Modelica, has been developed in an international effort [109]. The Modelica language depends on non-causal modeling with true ordinary differential and algebraic equations and the use of object-oriented constructs to facilitate reuse of modeling knowledge [109]. A commercial application of Modelica language is Dymola which has a powerful graphic editor for composing models [110]. Ferretti et al. [107] use of Dymola with Modelica language to study a virtual prototyping of mechatronics systems. A complete machining center is modeled by the elements from Modelica standard libraries and then simulated in a software environment which fully supports the Modelica language, like OpenModelica [111]. Moreover, another example of model-based modeling is Simscape [112] which is a commercial software package integrated within MATLAB. It provides fundamental building blocks from multiple domains and allows the user to assemble into more complex models of the physical systems [112].

It should be pointed out that in the conceptual (or preliminary) design phase, high fidelity analysis is required to ensure key requirements are met, such as the aerodynamic performance for unmanned aircraft. However, when the physical system is represented by mathematic equations or bond graphs, too much geometric information has been lost. As a result, these approaches cannot support the use of analysis tools requiring high fidelity geometric models, such as computational fluid dynamics (CFD) or finite element method (FEM). Although the object-oriented modeling language, like Modelica, allows to convert mechanical CAD models to Modelica models in terms of geometrical and inertia parameters, the translation is unidirectional. CAD models cannot be reversely modified [107]. Therefore, several researchers develop methods and tools to integrate the object-oriented modeling technique with CAD systems. La Rocca and Van Tooren [113] propose the concept of high level primitives (HLPs) for aircraft conceptual design. The HLPs are designed for capturing elements of similarity among very different configurations and using them as the parametrized modules for the geometric modeling [113]. Every HLP is programmed as an object. The engineers can quickly construct an aircraft concept by constructing the parametric geometric elements. Unlike code-based modeling, other researchers select CAD tools to generate the components of physical systems. Similar to those of La Rocca and Van Tooren, Amadori et al. [62] introduce the concept of high level CAD templates (HLCt), with the exception that the HLCt are generated and utilized in a CAD environment. Berard and Rizzi [114] establish a large parametrized library of aircraft components, from which the aircraft model can be created through a sequential process of sizing and assembly.

However, only physical modeling is not enough to support high fidelity analyses in the MDO framework. On one hand, a typical complex system has many aspects which can be analyzed by corresponding professional software packages. However, high fidelity modeling of physical systems in a single modeling language is almost impossible and unnecessary, because it is not straightforward to estimate which kinds of analyses are required in reality. Therefore, in order to perform different analyses, the complex physical systems are usually represented by various models (design representations) within specific modeling languages. As is shown in Figure 3.2, an inconsistency error may occur due to lack of data and information communication among the modeling languages. A feasible solution is to extract information from one single model, which is typically a (parametric) geometric model and then to generate the models for other domains. For example, a multidisciplinary software environment called computerized environment for aircraft synthesis and integrated optimization methods (CEASIOM) is developed for overall aircraft design [115, 116]. The geometric model of the aircraft is produced by CAD tools, which provides the geometry for CFD analysis and other information such as mass distribution and the aerodynamic data obtained using CFD, for the evaluation of flying qualities. Moreover, Barth and Fay [117] extract the information from a computer aided engineering (CAE) system to build the simulation model for control code tests. The parameters from the CAE system are mapped to process control systems (PCS) by setting up the libraries for both of them. As a result, the consistency of the simulation models can be ensured by using the above methods and tools. However, the disadvantage of this method is that the transformation is usually unidirectional, potentially leading to unexpected disciplinary errors. For instance, a brushless motor can be modeled by constructing the components from the Modelica.Electrical.Analog.Basic library. As mentioned, because there is only unidirectional translation from the CAD model to the Modelica model, an update of the motor model in Modelica environment will not have an effect on the motor dimensions in the CAD tools. Thus, when the engineer decides to use a bigger motor based on the results from a dynamics analysis, the new motor may not fit the original position, resulting in geometric interference. On the other hand, the models also have to be configured somewhat to enable the simulations in the analysis tools, such as assigning values to variables, selecting test maneuvers, trimming and executing simulation. These processes are repeatedly performed in every MDO cycle. If a

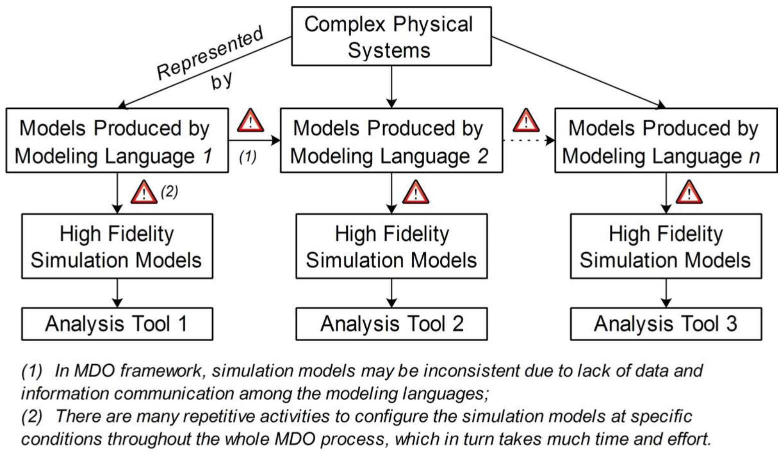

Complex Physical Systems
Represented by
Models Produced by Modeling Language 1
Models Produced by Modeling Language 2
Models Produced by Modeling Language n
(1)
(2)
High Fidelity Simulation Models
High Fidelity Simulation Models
High Fidelity Simulation Models
Analysis Tool 1
Analysis Tool 2
Analysis Tool 3
(1) In MDO framework, simulation models may be inconsistent due to lack of data and information communication among the modeling languages;
(2) There are many repetitive activities to configure the simulation models at specific conditions throughout the whole MDO process, which in turn takes much time and effort.

Figure 3.2: The drawbacks of traditional modeling language to support MDO framework

MDO problem involves several analysis tools, it will take much time and effort to configure the models at specific operating conditions for all of them throughout the whole MDO process (Figure 3.2).

The concept of a “master model" has been proposed by La Rocca and Van Tooren [113] and Berard and Rizzi [114] in their publications. However, the master model only contains the information of physical modeling aspects and lacks information on simulation expertise. The remaining challenges can be summarized as follows:

- The generation of consistent design representations for use throughout the entire MDO process requires the availability of high fidelity multiphysics simulation models in the early design phases;
- The development of high fidelity multiphysics simulation models for analysis and development of the engineering systems is a complex, time consuming and multidisciplinary task that requires a large amount of manual work from simulation experts.

To ensure consistency, this research study proposes the concept of MIM by using the KBE technique. KBE is a technology based on dedicated software tools called KBE systems, which are able to capture and systematically reuse product and process engineering knowledge, with the final goal of reducing time and costs of product development. The KBE systems cannot only automate the repetitive and non-creative design tasks but also support the MDO in all the phases of the design process [53].

As can be seen in Figure 3.3, the MIM contains not only the design knowledge for modeling physical systems across multiple domains but also the

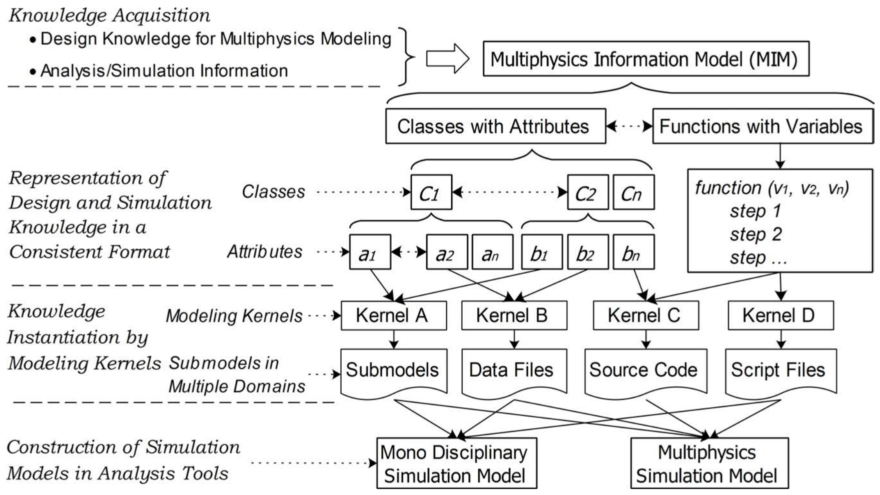
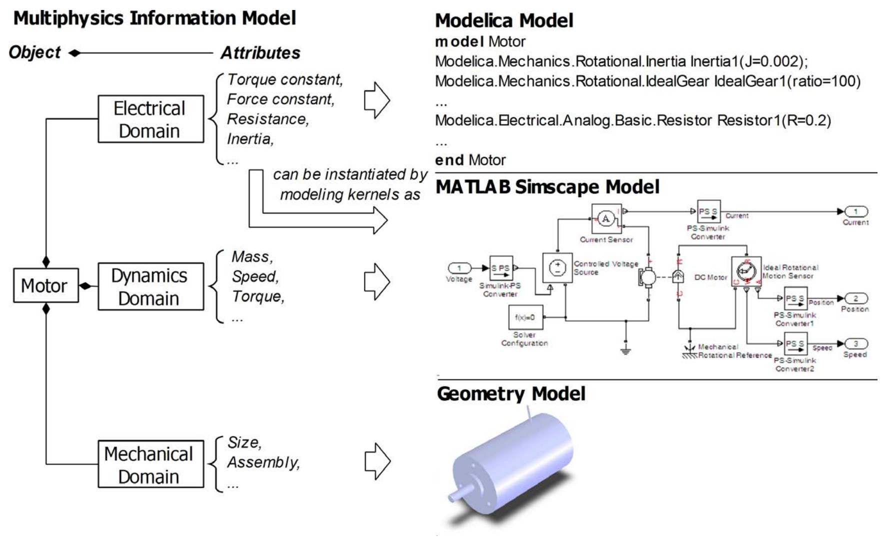

Knowledge Acquisition
• Design Knowledge for Multiphysics Modeling
• Analysis/Simulation Information
Multiphysics Information Model (MIM)
Classes with Attributes
Functions with Variables
Representation of Design and Simulation Knowledge in a Consistent Format
Classes
C1
C2
Cn
Attributes
a1
a2
an
b1
b2
bn
function (v1, v2, vn)
step 1
step 2
step ...
Knowledge Instantiation by Modeling Kernels
Modeling Kernels
Kernel A
Kernel B
Kernel C
Kernel D
Submodels in Multiple Domains
Submodels
Data Files
Source Code
Script Files
Construction of Simulation Models in Analysis Tools
Mono Disciplinary Simulation Model
Multiphysics Simulation Model

Figure 3.3: The concept of multiphysics information model (MIM)

analysis/simulation information. On one hand, characteristics of physical entities are defined as classes with attributes. The classes are structured hierarchically from the top level assembly to components and finally parts. For a single class, the attributes are grouped by the relevant engineering domains, such as dynamics, aerodynamics, structure, etc. Because design parameters can affect multiple domains, attributes in one group can be determined by referring to attributes from other groups. It is also allowed to compute the attributes for one class based on the attributes from other classes. Thus, interactions among the physical entities and domains are taken into account.

For example, a DC motor which is a mechatronic product can be represented in mechanical, dynamics and electrical domains (see Figure 3.4). In this example, the motor can be modeled by Modelica language or MATLAB Simscape blocks. Both of these models can give accurate results in electrical and dynamics domains. However, the Modelica language and MATLAB Simscape approach are too specialized to model the motor as a whole product. There is no geometry information captured by the Modelica or MATLAB Simscape model. Compares to traditional modeling language, the MIM represents the motor as a whole system in the knowledge level rather than focusing on specific simulation requirement. Next, this information model can be instantiated by modeling kernels as the simulation models for serving the MDO analysis tools. Therefore, this method ensures all the simulation models are consistent.

On the other hand, the MIM also defines functions with variables to capture the non-physical information. As for the analysis activities during the MDO process, the non-physical information is more about specifying the operation



```
model Motor
Modelica.Mechanics.Rotational.Inertia Inertia1(J=0.002);
Modelica.Mechanics.Rotational.IdealGear IdealGear1(ratio=100)
...
Modelica.Electrical.Analog.Basic.Resistor Resistor1(R=0.2)
...
end Motor
```

Figure 3.4: Comparison of an electric motor modeled by multiphysics information model, Modelica language, MATLAB Simscape and geometry model

conditions (e.g., environmental conditions), determining the test maneuvers and configuring the simulation model (e.g., weight, c.g., position, etc.). The functions can be instantiated as script files which are used to manipulate the submodels of the physical systems.

For instance, control system design is typically done following a set of procedures. First, a control system architecture and control method is defined. Next, the plant model is trimmed in a specific operating condition and a linear model is derived. Based on the linear model, the control system is tuned and analyzed. Once it complies with the requirements it is tested on the original plant model. All these procedures (both for design and analysis) are captured by functions in the MIM. In the current research study, model based inversion control is chosen as control method. This is a powerful method because consistent control system can be designed without having to tune gains [118, 119].

Subsequently, one or several submodels construct the simulation models together with the script files. Based on the requirements, the simulation models can be mono disciplinary or multiple domains (multiphysics simulation models). Because all simulation models are obtained from one source, the representation of the physical system is consistent among the disciplines.

Moreover, the proposed KBE tool is also an intelligent modeling system. By capturing the expertise from simulation experts, it is able to construct the simulation model in an automated fashion, which in turn accelerates the MDO process in all domains and analysis tools. Thus, the intelligent modeling system makes it possible to establish a concurrent design environment to support the MDO process. Compared to related work, the proposed KBE system is characterized by the following two items:

- Representation of design knowledge from multiple domains in a unified format to ensure consistency among the disciplines.

- Automated generation of the multiphysics simulation models with a right level of fidelity to serve the analyses in MDO.

3.3. MULTIPHYSICS MODELING BY INTELLIGENT MODELING SYSTEM

A possible solution for solving model inconsistency is that we define a unified representation of the physical systems in information level and then instantiate this into the specific simulation model required by the analysis tools. As is shown in Figure 3.3, there are four steps to generate the multiphysics simulation model. First, the intelligent modeling system collects the product and process information from handbooks, engineers and experts. Second, the collected information is written as the MIM (classes and functions) by utilizing the KBE technique. Third, the MIM is instantiated as submodels by modeling kernels. Finally, the submodels are assembled as the multiphysics simulation model in the analysis tools.

The intelligent modeling system is composed of three modules: (1) the inference engine, (2) the MIM and (3) the communication framework. Moreover, the intelligent modeling system calls the modeling kernels during the generation of the simulation models although they are not considered as software modules in the intelligent modeling system. The software modules of the intelligent modeling system are further discussed along with the generation of multiphysics simulation model.

The proposed KBE system is built in the compiler of Emacs with the *genworks general-purpose, declarative, language* (GDL), which is based on the ANSI standard version of Common LISP. GDL is particularly effective at representing complex systems, including three-dimensional geometry models and design process [75].

3.3.1. KNOWLEDGE ACQUISITION FOR MULTIPHYSICS INFORMATION MODEL

Before modeling the physical systems, the intelligent modeling system has to collect the information from several aspects. As is shown in Figure 3.5, the MIM not only needs the design knowledge for physical modeling and control system design but also the information for analysis/simulation configurations. Moreover, in order to construct the overall simulation model, it may also need the components from existing libraries and some analysis results obtained before (e.g., tables, fitted curves, response surface, etc.). Furthermore, in order to automatically produce the simulation models, the intelligent modeling system also has to collect the expertise from simulation experts to organize the collected information in a proper structure. Typically, there are two key challenges faced by the intelligent modeling system:

- Consideration of the interactions among the domains during the model generation;
- Making sure the components from different domain libraries are consistent.

On one hand, some inputs may be computed based on other analysis results (Figure 3.6). For example, an airfoil model is required to provide aerodynamic lift and drag for a helicopter model. If simulation time is a critical factor, the intelligent modeling system has to first instantiate the 2D geometries of specific airfoil from the multiphysics information model, test it in a low fidelity analysis

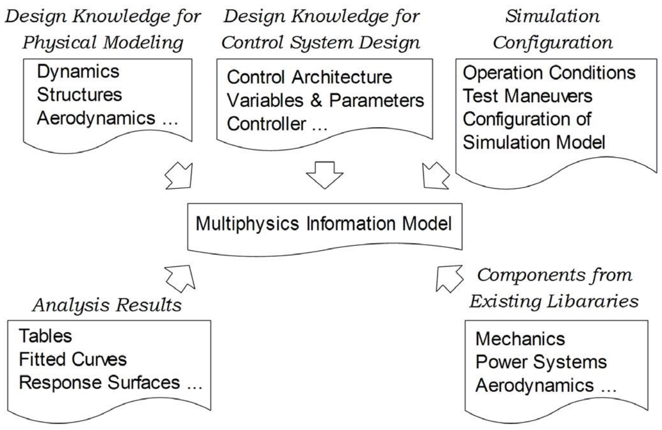


Figure 3.5: The required information for the construction of multiphysics information model (MIM)

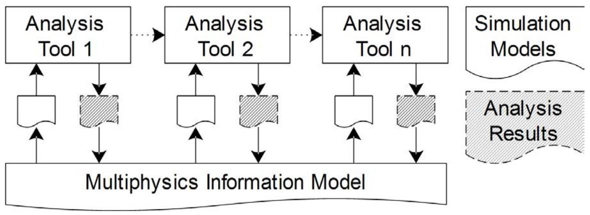


Figure 3.6: Specification of the sequence for analysis tools to construct the multiphysics information model

tool and then get the table of lift vs. drag back to the multiphysics information model. Lately, the 2D geometry model may be expended as a whole blade and then simulated in a CFD environment to provide more accurate results, such as torque vs. Mach number, when the simulation accuracy is more important. In this case, intelligent modeling system has to specify the sequence for the analysis tools from low fidelity to high as well as the model generation from 2D to 3D models.

On the other hand, the intelligent modeling system has to configure the multiphysics simulation model from a list of possible components (or technologies) and make sure that submodels are consistent with each other, especially when components from different disciplinary libraries are used. For example, if structural vibrations at high speed forward flight of a new helicopter design must be calculated, a complex high fidelity inflow model is required, which is highly integrated with the structural model of the rotor [120]. However, if the multiphysics simulation model is required to calculate the achievable climb rate of the helicopter at low speed flight, or for the development of basic flight control laws used for stabilization, a much simpler inflow model and structural model can be used.

An inference engine is included in the intelligent modeling system to solve above challenges. The principal role of the inference engine is to search for the most appropriate item of knowledge to apply at any given moment [61]. Initially, the inference engine of the intelligent modeling system collects the expertise from the simulation experts and converts this into logic expressions or requirements. Next, the inference engine can organize the design and simulation knowledge in a proper structure or select the required knowledge (components or strategies) by evaluating logical expressions or by using a decision-making strategy, such as an analytic hierarchy process (AHP). For example, if structural vibrations at high speed forward flight of a new helicopter design must be calculated, a complex high fidelity (aerodynamic) rotor inflow model is required, which is highly integrated with the structural model of the rotor [120]. However, if the multiphysics simulation model is required to calculate the achievable climb rate of the helicopter at low speed flight, or for the development of basic flight control laws used for stabilization, a much simpler rotor inflow model and structural model can be used. A rule is defined for the inference engine to select the best inflow model based on requirement of the design to be analyzed and the components used for modeling other elements, such as the structural dynamics of the rotor model.

3.3.2. CONSTRUCTION OF MULTIPHYSICS INFORMATION MODEL

A dedicated code-based modeling approach is applied in current research study to implement such concept. As can be seen in Figure 3.7, a typical MIM follows a tree structure. The root of the MIM is a top-level assembly which falls into several subclasses. The subclasses may also include the subclasses in next level or down to the “leaf” classes at the bottom. Typically, each “leaf” class corresponds to a physical entity in real world and the associated attributes are grouped by engineering domains. As mentioned in Section 3.2, the attributes can be determined by top-level configuration or expressions referring to other attributes or classes. Several examples are given in Figure 3.7. For example, the dynamics attribute of a21 in class a1 is computed by an expression of the attributes of a11 and a12 in mechanical domain. Again, the attribute of class a3 a41 together with class a2 a31 determine the attribute of a41 for class a1. Hence, it is possible to integrate design rules as well as the interactions among the domains (or physical entities) within the definitions of the classes. Moreover, functions can be defined as methods within a class definition or associated with specific class as external operations. The variables in the functions can also refer to the attributes of the classes.

3.3.3. KNOWLEDGE INSTANTIATION BY MODELING KERNELS

Subsequently, as can be seen in Figure 3.3, the MIM can be instantiated by modeling kernels into different of data (or file) formats depending on the analysis requirements. Unlike the functions used to save non-physical information, the intelligent modeling system defines extracting functions to get the values from the class attributes. The extracting function loops for the classes at same hierarchy level and extracts class attributes and definitions. In the meanwhile, the intelligent modeling system also searches for the non-physical information if necessary.

Next, the collected information is written into a specific data format by the modeling kernels within and outside of the intelligent modeling system. It should be noticed that which modeling kernel is applied by the intelligent modeling system is determined by the target data (or file) format and the

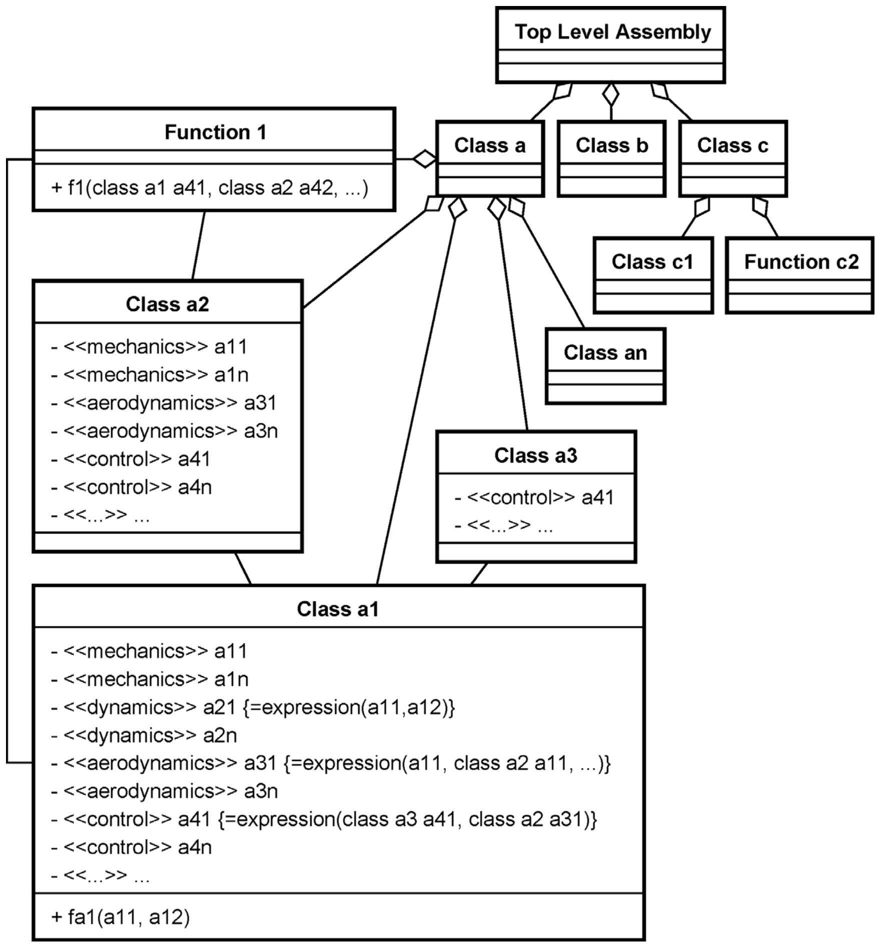


Figure 3.7: The class diagram of multiphysics information model

analysis tool. Typically, the intelligent modeling system uses internal functions which are inherent from the programming language to output the non-physical information and class attributes into strings and data files, respectively. Lately, the data files can be used to initialize the simulation models in the analysis tools. Moreover, the intelligent modeling system can also generate the model file which includes both class attributes and definitions with extra modeling kernels. The modeling kernels can be integrated within the intelligent modeling system, such as geometric modeling kernels, or defined as functions which generate the source code in specific format, like the definition of an XML file used for a dynamics simulation in MATLAB SimMechanics or a script file for the CFD analysis.

3.3.4. CONSTRUCTION OF MULTIPHYSICS SIMULATION MODELS IN ANALYSIS TOOL

Soon after, the instantiated model files can be directly assembled as a complete simulation model which could be mono disciplinary or multiphysics. It is argued that geometry is the most effective enabler for the integration of disciplines and it is also the most commonly used thread through the different disciplines required [30]. For example, in order to calculate the aerodynamic drag of an aircraft concept in the CFD tools, it mainly requires a geometry model of the vehicle and associated simulation configurations. However, to enable the analysis of the overall performance of a complex system, only the geometry model itself is not sufficient. Simulation experts have to develop a physical plant model which can be multiphysics represents the object to be simulated; including one or more control systems to regulate the behavior of the physical plant. In order to implement the concept of MIM with a concrete example, the multiphysics simulation model is defined as the combination of a multiphysics plant model, associated control systems and related simulation configurations in this research study. However, it should be pointed out that the constitution of the multiphysics simulation model can be different.

Hence, the process is presented in Figure 3.8. It is assumed that the intelligent modeling system has collected the structural information of the multiphysics simulation model from the simulation experts. This information is written by the intelligent modeling system as a script file. The script file includes the information of necessary submodels and the steps for the construction. Next, intelligent modeling system sets up connections with the analysis tool through the communication framework. Third, it executes the script file in the analysis tool which includes the commands to sequentially import the submodels instantiated by the modeling kernels. The results are the model file of the physical plant, source code of the control systems and the data file for simulation initialization, etc. The components in the domain libraries can be indexed by the data file as well. Finally, the intelligent modeling system compiles the script file in the analysis tool to construct a complete multiphysics simulation model by combining all the submodels together.

3.4. TEST CASE - MULTIPHYSICS MODELING OF MULTIROTOR UAV

The intelligent modeling system will be demonstrated by the automatic generation of multiphysics simulation models for various multirotor UAV designs with highly different dimensions and configurations (number of rotors for example). In the current chapter, the intelligent modeling system is tested to

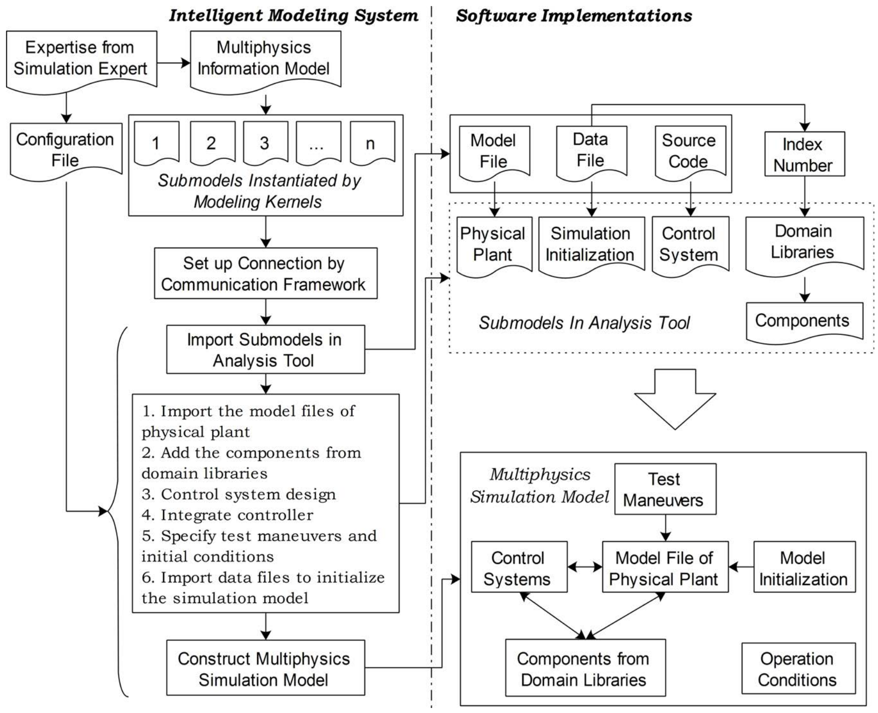

Intelligent Modeling System
Software Implementations
Expertise from Simulation Expert
Multiphysics Information Model
Configuration File
1
2
3
...
n
Submodels Instantiated by Modeling Kernels
Set up Connection by Communication Framework
Import Submodels in Analysis Tool
1. Import the model files of physical plant
2. Add the components from domain libraries
3. Control system design
4. Integrate controller
5. Specify test maneuvers and initial conditions
6. Import data files to initialize the simulation model
Construct Multiphysics Simulation Model
Model File
Data File
Source Code
Index Number
Physical Plant
Simulation Initialization
Control System
Domain Libraries
Components
Submodels In Analysis Tool
Multiphysics Simulation Model
Test Maneuvers
Control Systems
Model File of Physical Plant
Model Initialization
Components from Domain Libraries
Operation Conditions

Figure 3.8: Construction of multiphysics simulation model for mechatronic design

model the physical plant across multiple domains. The integrated design of the control system is discussed in the next chapter. Several steps are taken to generate the multiphysics simulation model:

(1) Specification of analysis sequence for setting up the multiphysics simulation model (test case for Section 3.3.1);

(2) Consideration of the interactions among the disciplines to select appropriate components from domain libraries (test case for Section 3.3.1);

(3) Representation of multirotor UAV by the MIM (test case for the definition of MIM in Section 3.3.2);

(4) Instantiation of the MIM into geometric shapes and dynamics submodels (test case for Section 3.3.3);

(5) Construction of multiphysics simulation model for multirotor UAV (test case for Section 3.3.4);

3.4.1. TOP LEVEL CONFIGURATION

For the current test case, the intelligent modeling system requires knowledge from experts in the fields of mechanics, aerodynamics, propulsion, dynamics and flight controls to specify the top level requirements for generating the MIM (see Figure 3.9). The disciplinary experts can provide their requirements in a web-based interface. Although the inputs are classified in the interface to clarify the responsibility, the knowledge is shared by all disciplines when creating the multiphysics simulation model. For example, the dynamics modeling of the physical system may also use the knowledge of the hierarchy structure, topology structure, dimensions and datum from the geometric modeling. Moreover, the airfoil shape of the blade will have an effect on the dynamics performance of the multirotor UAV. Furthermore, the mass and inertia can be specified by the simulation experts or calculated based on geometric information.

Part of the top level configuration for the arm assembly of the multirotor UAV is shown in Figure 3.9. In this case, the simulation expert specifies the dimensions and topology for the arms from the mechanical perspective, which is defined by the length, radius and number of arms. As for the aerodynamics, the expert also determines the information related to the airfoil shape and level of fidelity required for the rotor inflow model. The method to drive the rotor is determined by the simulation expert. In this example, it can be a “direct” or “indirect” drive system. Finally, the simulation expert also decides how the power is allocated to each rotor. It should be pointed out that the objective of showing the top level configuration is to illustrate that the design knowledge from multiple domains can be given in the same interface for designing the multirotor UAV. It does not imply that it is a straightforward task to clearly clarify

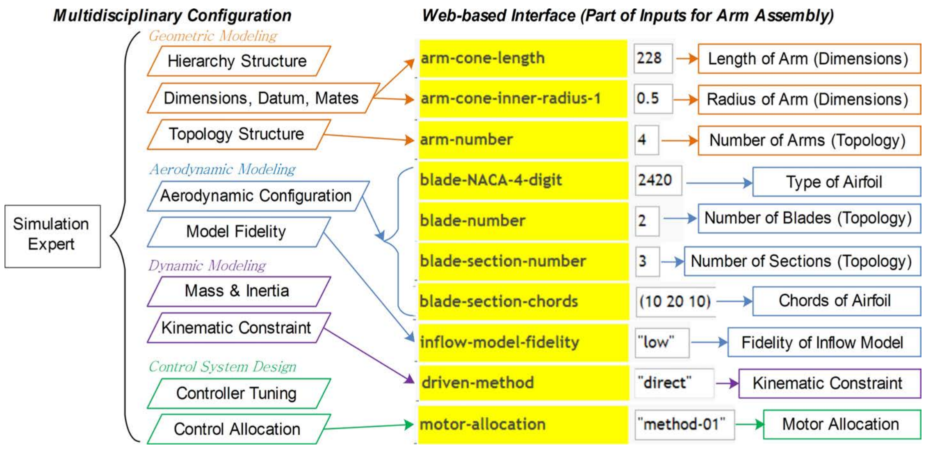


Figure 3.9: The top level configuration of multirotor UAV

the interactions among the disciplines.

3.4.2. SPECIFICATION OF ANALYSIS SEQUENCE

Before the modeling the physical entities, the intelligent modeling system should organize the analyses in a proper structure by capturing the knowledge from handbook and simulation expert. As can be seen in Figure 3.10, there are six subsystems coupled with each other for modeling the multirotor UAV. Initially, the inference engine of the intelligent modeling system sets the geometric modeling at the start of the sequence, prior to other aspects because most analyses require geometric information.

Then, there are five pairs of coupled domains in this example. Firstly, the battery capacity is calculated based on its size which is specified by geometric modeling and energy density. The amount of electrical power provided by the battery is used by the motor, which reversely determines the energy consumption of the battery. Next, the motor speed, combined with the motion of the vehicle is used to compute the forces and moments produced by the propeller. These forces and moments act on the body of the multirotor UAV and affect its motion. Moreover, aerodynamic forces and moments produced by the propeller and the aerodynamic forces acting on the main body are computed based on geometric information, such as the airfoil shape and the surface area of the multirotor UAV, respectively. The motion (accelerations) of the vehicle is calculated by the dynamics model based on the forces exerted on it, such as the aerodynamics. The control system uses various sensors to determine the state of the vehicle and sends commands, (a voltage in this example) to control the speed of motors and thereby the motion of the vehicle. The feedback signal

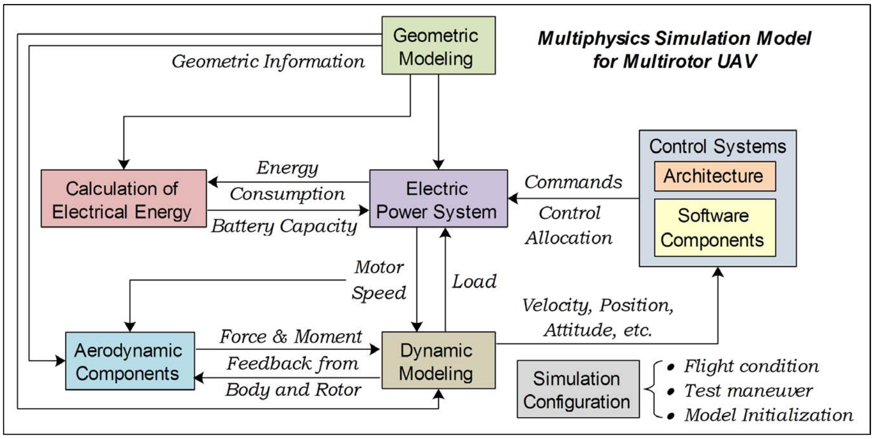


Figure 3.10: The disciplinary sequence for modeling the multirotor UAV

from the dynamics model is a load which may cause the motor to stall.

3.4.3. SELECTION OF THE MODELS FROM PREDEFINED COMPONENT LIBRARIES

It is common that the simulation expert selects components (or strategies) from predefined component libraries to construct the overall simulation model. In order to ensure the selected components are compatible with each other (or the strategies suitable for current case), the simulation expert has to determine which type of model is needed for a specific application based on the requirements. Typical requirements are; What aspects needs to be calculated with the multiphysics simulation model? What is the required fidelity of the result? How fast must the result be computed. Through capturing the knowledge from simulation experts, the inference engine of the intelligent modeling system is responsible for selecting appropriate components (or strategies) from predefined libraries for feeding the model generation.

On one hand, the inference engine has to select appropriate components (or strategies) based on the design and simulation requirements. As shown in Figure 3.11, the payload has an effect on the topology structure of the multirotor UAV. By capturing the knowledge from the design and simulation experts, the inference engine defines an IF-THEN logic to select different possible topology structures for the multirotor UAVs based on the payload. Consequently, it also specifies the corresponding possible control architectures for the multirotor UAV with N rotors.

On the other hand, the inference engine has to make sure the selected

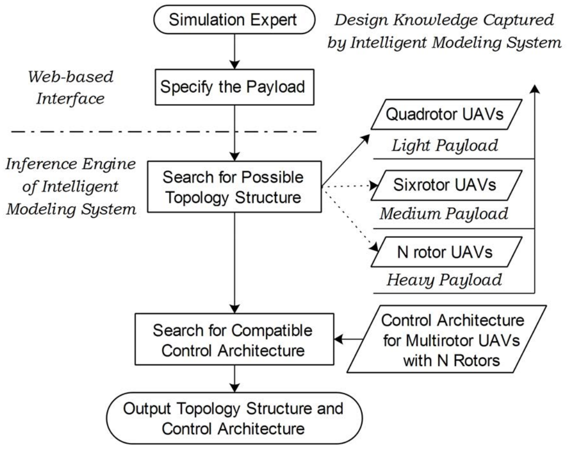

Simulation Expert
Design Knowledge Captured by Intelligent Modeling System
Web-based Interface
Specify the Payload
Quadrotor UAVs
Light Payload
Inference Engine of Intelligent Modeling System
Search for Possible Topology Structure
Sixrotor UAVs
Medium Payload
N rotor UAVs
Heavy Payload
Search for Compatible Control Architecture
Control Architecture for Multirotor UAVs with N Rotors
Output Topology Structure and Control Architecture

Figure 3.11: The mechanism to select different control architectures for the multirotor UAVs

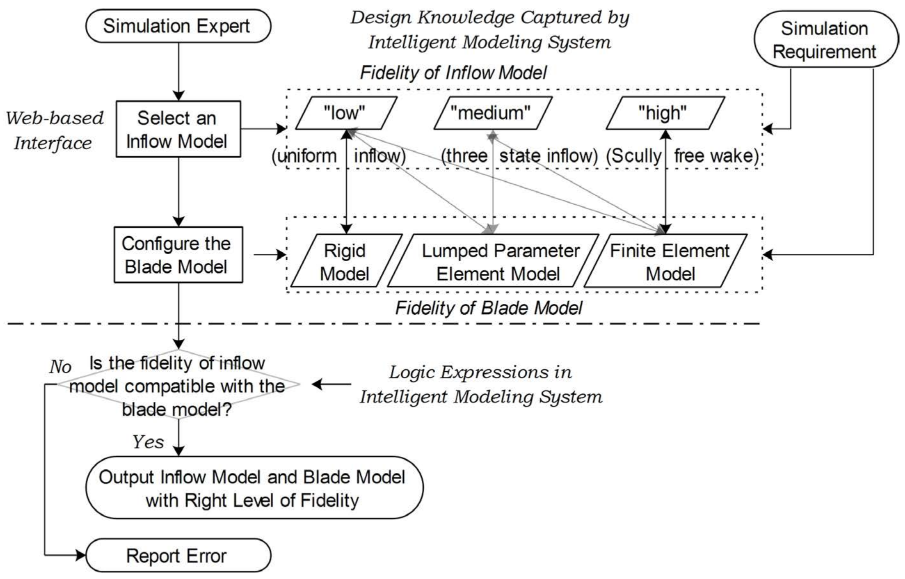

Simulation Expert
Design Knowledge Captured by Intelligent Modeling System
Simulation Requirement
Fidelity of Inflow Model
Web-based Interface
Select an Inflow Model
"low"
"medium"
"high"
(uniform inflow)
(three state inflow)
(Scully free wake)
Configure the Blade Model
Rigid Model
Lumped Parameter Element Model
Finite Element Model
Fidelity of Blade Model
No
Is the fidelity of inflow model compatible with the blade model?
Logic Expressions in Intelligent Modeling System
Yes
Output Inflow Model and Blade Model with Right Level of Fidelity
Report Error

Figure 3.12: Integration of design rules in intelligent modeling system to ensure model consistency

models are compatible with each other. For example, in order to evaluate the flight performance of multirotor UAV, a rotor inflow model is required and this must be integrated with the model representing the dynamics of the rotor blade (equations of motion). Thus, the inflow model must be compatible with the way the blade is modeled. As can be seen in Figure 3.12, the intelligent modeling system uses simulation expertise, and requirements for the simulation to select the model fidelity for the inflow model and the corresponding blade model. A rule is defined in the inference engine to check which inflow models are compatible with the available blade models. This is expressed in the IF-THEN form. For example, the blade model with high fidelity can be used in a simulation with any of the inflow models. However, based on simulation requirements, such as the required fidelity of the final result, it may be decided not to use the low fidelity blade model in conjunction with the high fidelity inflow model. When the simulation experts select the inflow model and configure the blade model in the web-based interface, the inference engine checks the inputs of “inflow-model-fidelity” and “blade-section-number” with the knowledge captured in logical expressions to determine whether they are compatible or not. If yes, the inference engine returns selection results for the inflow model and rotor model with right level of fidelity. Otherwise, it will report an error. Finally, the inflow model and the blade model can be instantiated for the simulation of multirotor UAV.

3.4.4. REPRESENTATION OF THE MULTIROTOR UAV BY MULTIPHYSICS INFORMATION MODEL

The construction of multiphysics simulation model for multirotor UAV follows the procedures in Figure 3.8. A top-level class diagram of the MIM is shown in Figure 3.13. The biggest class in Figure 3.13 represents the top-level configuration class which can be configured by the simulation experts. Then, the attributes in the top-level configuration class are passed down to corresponding subclasses, such as control system, component library and test maneuvers.

The multirotor UAV consists of several identical arm assemblies, composed of three elements; the arm, motor and rotor. An example of the MIM for the arm assembly is shown in Figure 3.14. To define these three elements, different approaches have to be used:

- Model generated by modeling kernels (the arm);
- Model indexed from libraries/database (the motor);
- Extra information from other domains is required to specify the attributes (the rotor).

GEOMETRIC INFORMATION

As shown in Figure 3.10, the geometric modeling element is a starting point and it provides geometric information to the other elements. For the mechanical

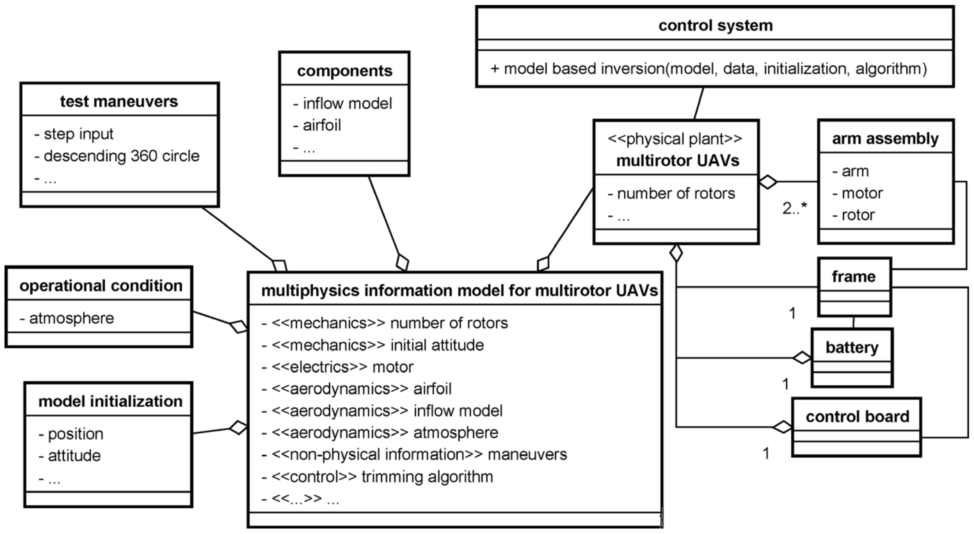


Figure 3.13: Class diagram of multiphysics information model (MIM) for multirotor UAV

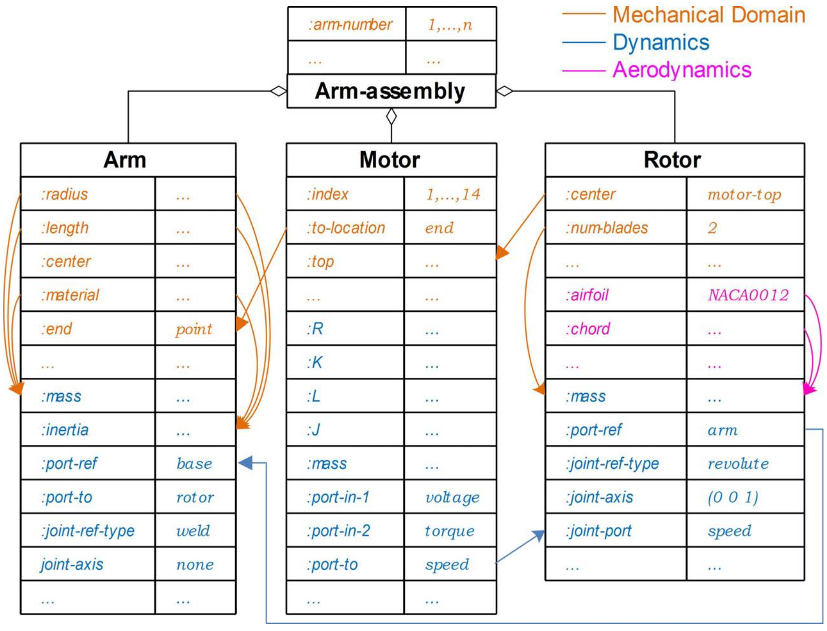
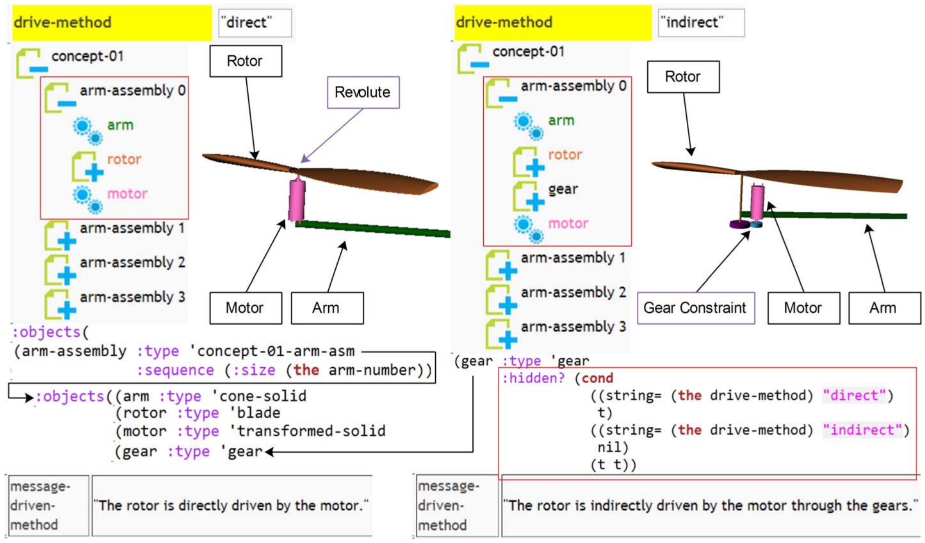


Figure 3.14: Representation of the arm assembly by using the concept of multiphysics information model (MIM) across multiple domains

design (orange font in Figure 3.14), the attributes are typically hierarchical structure, topology structure, dimensions and assembly references.

Hierarchical Structure The hierarchical structure records the relationship from the top-level assembly to the components and parts. As mentioned, the intelligent modeling system is programed with an OOP language. Thus, the hierarchical structure of the physical system is represented by class and its child class in the MIM. This is shown in Figure 3.15. The keyword ":type" for each class defines the name of the child class. The physical entities within the brace of the keyword of ":objects" are positioned at same level. In other words, they are shared by a common parent class. For example, as can be seen in Figure 3.15, the name of the child class for the arm assembly is the "concept-01-arm-asm". Because the arm, rotor, motor and gear are within the brace of "objects" for the "concept-01-arm-asm", they are the child classes of the arm assembly.

It should be pointed out that how the classes are organized is determined by the inference engine. By defining a rule, it is possible to modify the hierarchical structure according to the selected configuration. In this example, the gear class is hidden. If the input "drive-method" is changed from "direct" to "indirect" in



```
:objects(
(arm-assembly :type 'concept-01-arm-asm
              :sequence (:size (the arm-number))
:objects((arm :type 'cone-solid
         (rotor :type 'blade
         (motor :type 'transformed-solid
         (gear :type 'gear
```

```
(gear :type 'gear
:hidden? (cond
          ((string= (the drive-method) "direct")
           t)
          ((string= (the drive-method) "indirect")
           nil)
          (t t))
```

message-driven-method	"The rotor is directly driven by the motor."

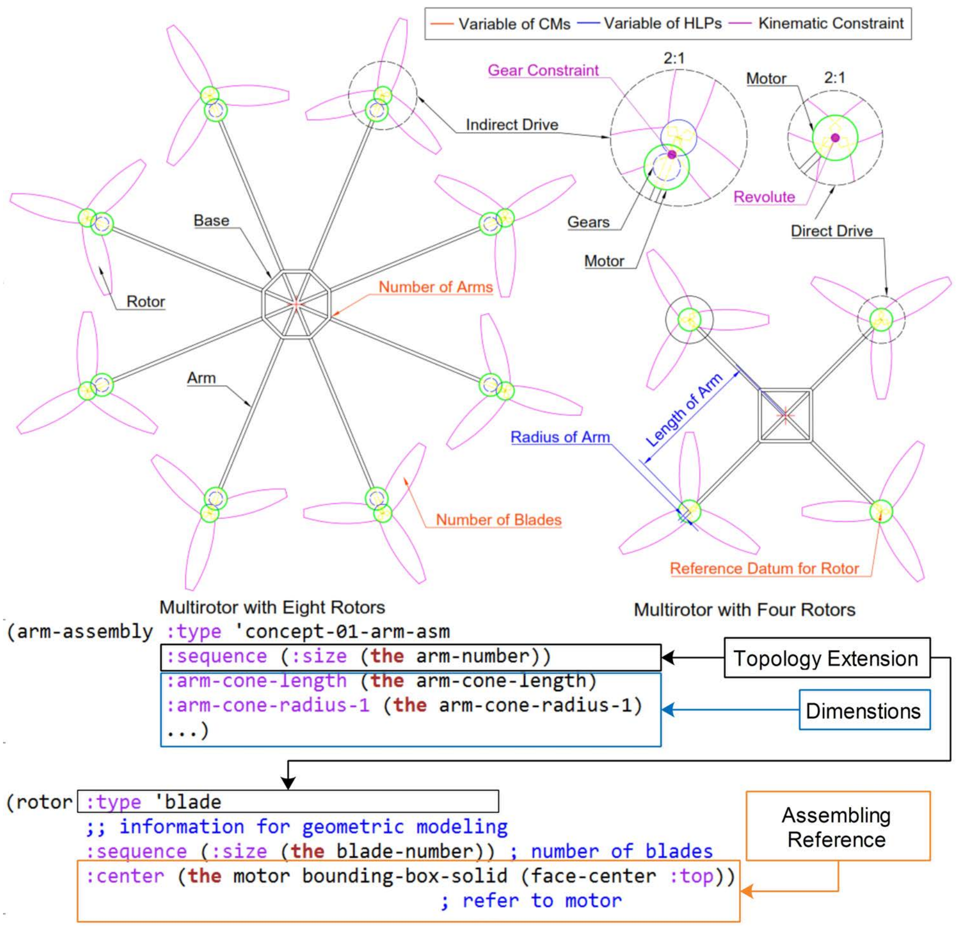
message-driven-method	"The rotor is indirectly driven by the motor through the gears."

Figure 3.15: The hierarchy of the arm assembly for the multirotor UAV (source code in GDL)

the web-based interface, a pair of gears is added to the arm-assembly (see Figure 3.15). In other words, a switch logic can be defined in the rule base to control the class to be hidden or shown in the hierarchical tree. Thus, the engineers can use predefined keywords in the web-based interface to rapidly configure the hierarchy structure. Meanwhile, the intelligent modeling system will return the selection results to the user through the interface, which is illustrated in Figure 3.15. Moreover, it should be noticed that the motor is also repositioned to adapt to the gear system.

Topology Structure It is also allowed to change the topology structure. A part or component can be duplicated by specific design rules, such as translation or rotation, to create a more complex component or assembly. How many duplications is determined by specific attribute. For example, the modeling kernel will instantiate the arm assembly *N* times by giving different values to the attribute of “arm-number” in Figure 3.14. Therefore, a multirotor UAV with *N* rotors is automatically obtained. It should be pointed out that the number of the components, such as the motors and rotors, as well as the reference points are also updated although with the topology extension.

Adapted to the geometric modeling kernel (SMLib [121] in this test case), the key word for topology structure is “:sequence”, which is shown in Figure 3.16. The source code in Figure 3.16 shows that the topology of the multirotor UAV is determined by the attribute of “arm-number”. Because the value of the “arm-



```
(arm-assembly :type 'concept-01-arm-asm
              :sequence (:size (the arm-number))
              :arm-cone-length (the arm-cone-length)
              :arm-cone-radius-1 (the arm-cone-radius-1)
              ...)

(rotor :type 'blade
       ;; information for geometric modeling
       :sequence (:size (the blade-number)) ; number of blades
       :center (the motor bounding-box-solid (face-center :top))
                                        ; refer to motor
```

Figure 3.16: The geometries of multirotor UAVs instantiated from the multiphysics information model (MIM)

number” is set previously in the web-based interface (Figure 3.9) as four, there are four arm assemblies shown in the tree structure (Figure 3.15). If the value of the “arm-number” is changed to eight, a multirotor UAV with eight rotors can also be automatically generated.

By extending the topology of the physical system, the design freedom is not limited to simple modification of dimensions but the manipulation of components in a group. The topology extension obviously has a major effect on the dynamics, aerodynamics and control of the multirotor UAV.

Dimensions and Reference Points The MIM also includes attributes for dimensions and assembly references. Depending on the geometric shape to be modeled, the attribute for the dimensions can be “:length”, “:width” and “:height” for a box, or “:radius” for a sphere. The attribute of “:center” and “:orientation” determines the assembling references where the class is positioned and the orientation matrix, respectively.

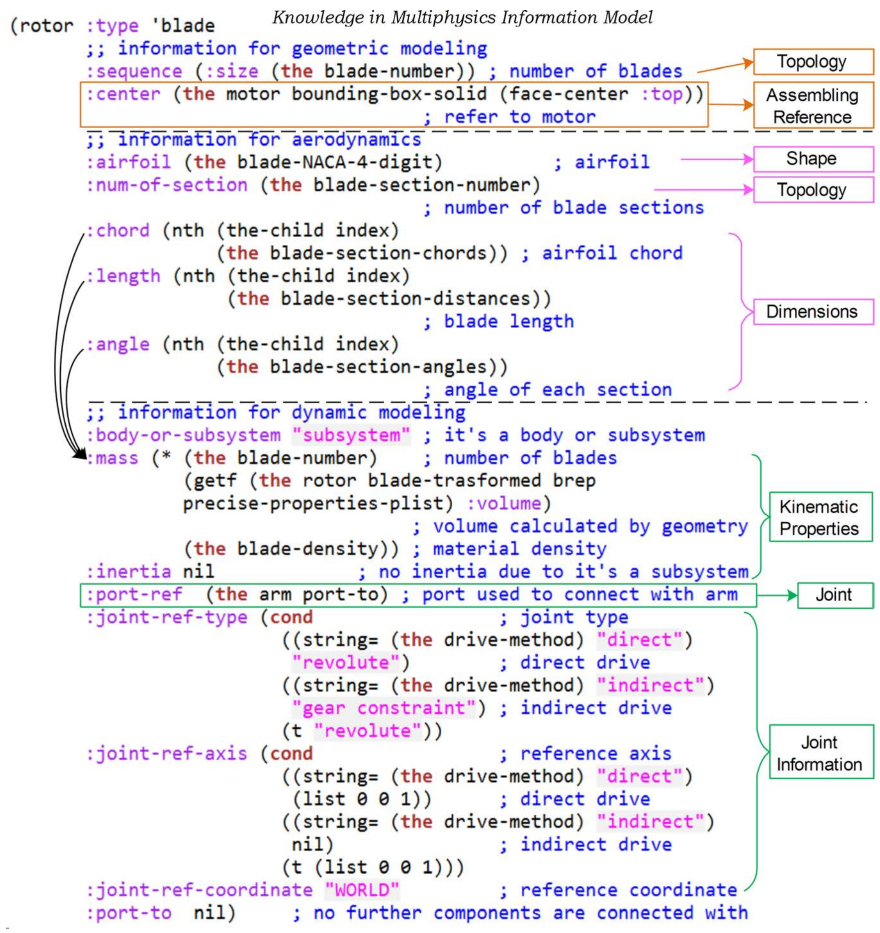
The dimensional attributes control the geometric shapes in mechanical design. In this example, the arm is represented by a cone, which is determined by two attributes: “:radius” and “:length” (see Figure 3.14). Adapted to the source code (see Figure 3.9 and 3.16), the engineer can set the value of “arm-cone-length” and “arm-cone-inner-radius-1” in the web-based interface to change the its shape. This is to say, feeding the MIM with different input values will produce various geometric shapes. Every geometric representation is different although they are derived from the same source.

Besides dimensional attributes, there are also attributes for positioning the geometries. For example, the motor is positioned at the endpoint of the arm. Therefore, the attribute of “to-location” for the motor refers to the attribute of “:end” for the arm (see Figure 3.14). Again, the rotor is positioned at the top of the motor shaft. Hence, its “center” refers to the “top” of the motor. Because all of the reference points are computed by the current configuration, the motor and rotor can automatically move to the new positions when the top-level configuration is changed. In more complex cases, the positioning attributes can also be specified by expressions of serial operations, such as rotation, translation in direction or along a vector. More complex engineering rules can also be integrated in a similar way.

Dynamics Information

The dynamics of the physical plant is a key component of the multiphysics simulation model. For the current application, multi-rigid body dynamics is a suitable approach for simulating the dynamics. Typically, the hierarchical structure and topology structure can be inherited from geometric information required for dynamics modeling. However, the MIM requires additional information, such as inertias and the definition of kinematic constraints, to model the dynamics characteristics. The dynamics properties of each physical element are defined by two elements: bodies and joints.

The bodies represent the physical entities in real world. The mass and inertia of each body is computed based geometric information. Every time the geometry is changed, the mass and inertia are automatically updated. As can be seen in Figure 3.14 (blue font), the attributes “:mass” and “:inertia” of the arm are calculated based on the geometry and selected material. The material can be specified in the top-level configuration. A more specific example is given in Figure 3.17. The attribute of “:mass” calculates the mass of the rotor based on

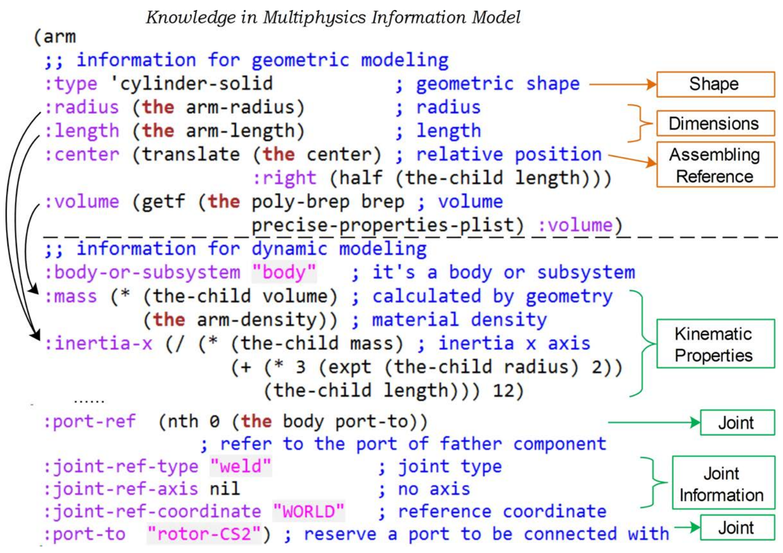


```
(rotor :type 'blade
       ;; information for geometric modeling
       :sequence (:size (the blade-number)) ; number of blades
       :center (the motor bounding-box-solid (face-center :top))
                                        ; refer to motor
       ;; information for aerodynamics
       :airfoil (the blade-NACA-4-digit)          ; airfoil
       :num-of-section (the blade-section-number)
                                        ; number of blade sections
       :chord (nth (the-child index)
                   (the blade-section-chords)) ; airfoil chord
       :length (nth (the-child index)
                    (the blade-section-distances))
                                        ; blade length
       :angle (nth (the-child index)
                   (the blade-section-angles))
                                        ; angle of each section
       ;; information for dynamic modeling
       :body-or-subsystem "subsystem" ; it's a body or subsystem
       :mass (* (the blade-number)     ; number of blades
                (getf (the rotor blade-trasformed brep
                precise-properties-plist) :volume)
                                       ; volume calculated by geometry
                (the blade-density)) ; material density
       :inertia nil                ; no inertia due to it's a subsystem
       :port-ref  (the arm port-to) ; port used to connect with arm
       :joint-ref-type (cond                ; joint type
                        ((string= (the drive-method) "direct")
                         "revolute")        ; direct drive
                        ((string= (the drive-method) "indirect")
                         "gear constraint") ; indirect drive
                        (t "revolute"))
       :joint-ref-axis (cond                ; reference axis
                        ((string= (the drive-method) "direct")
                         (list 0 0 1))      ; direct drive
                        ((string= (the drive-method) "indirect")
                         nil)               ; indirect drive
                        (t (list 0 0 1)))
       :joint-ref-coordinate "WORLD"        ; reference coordinate
       :port-to  nil)     ; no further components are connected with
```

Figure 3.17: The source code of the multiphysics information model (MIM) for the rotor

the blade number, volume and material. In this example it is 4.5 g. The blade number and material can be specified in the top-level configuration, while the volume of the blade is extracted from its geometric model. The inertia is treated by the MIM in a similar fashion.

The joints are the kinematic constraints in the multi-rigid body dynamics simulation. A joint is determined by four elements: the bodies to be connected with, joint type, the reference axis and the reference coordinates. For example, the rotor is driven by the motor to rotate around the local Z axis of the endpoint of the arm. Therefore, the rotor refers to the arm by the attributes of ":port-ref" in the dynamics domain, which is shown in Figure 3.14. Correspondingly, the attribute of ":joint-ref-type" is "revolute" in this example. If a gear system is



```
(arm
 ;; information for geometric modeling
 :type 'cylinder-solid            ; geometric shape
 :radius (the arm-radius)         ; radius
 :length (the arm-length)         ; length
 :center (translate (the center) ; relative position
                   :right (half (the-child length)))
 :volume (getf (the poly-brep brep ; volume
                   precise-properties-plist) :volume)
 ;; information for dynamic modeling
 :body-or-subsystem "body"   ; it's a body or subsystem
 :mass (* (the-child volume) ; calculated by geometry
          (the arm-density)) ; material density
 :inertia-x (/ (* (the-child mass) ; inertia x axis
                  (+ (* 3 (expt (the-child radius) 2))
                     (the-child length))) 12)
     ......
 :port-ref  (nth 0 (the body port-to))
                ; refer to the port of father component
 :joint-ref-type "weld"            ; joint type
 :joint-ref-axis nil               ; no axis
 :joint-ref-coordinate "WORLD"     ; reference coordinate
 :port-to  "rotor-CS2") ; reserve a port to be connected with
```

Figure 3.18: The source code of the multiphysics information model (MIM) for the arm

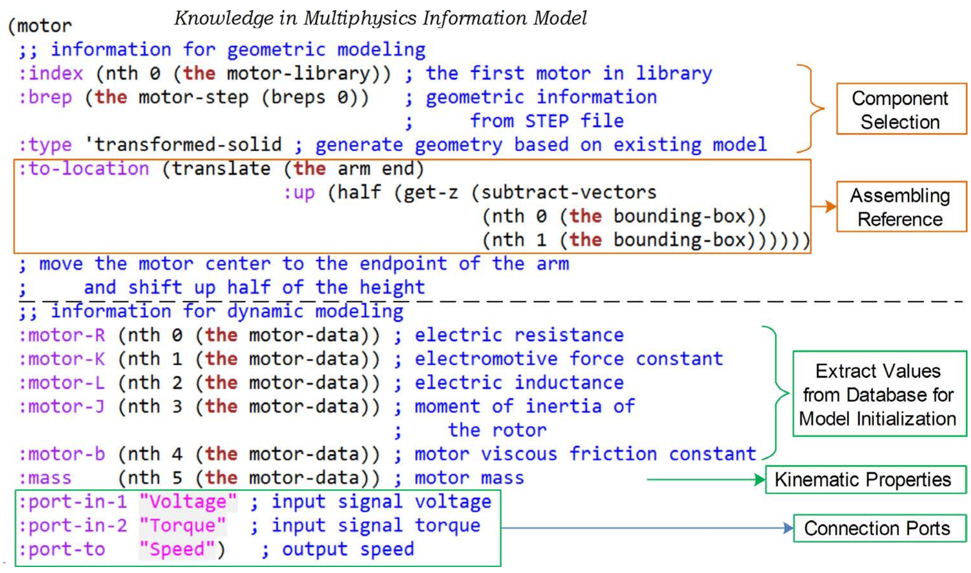
added between the rotor and the motor, the ":joint-ref-type" will change to "gear constraint".

In summary, the source code of the MIM to represent the arm, rotor and motor are shown in Figure 3.17, 3.18 and 3.19, respectively.

MODELING OF E/E COMPONENTS

There are also attributes for modeling the E/E components. As is shown in Figure 3.14, the motor has attributes "R", "K", "L" and "J", which define the electric resistance, electromotive force constant, electric inductance and moment of inertia of the rotor, respectively. These attributes can later be instantiated by the modeling kernels as data file or model file to construct the multiphysics simulation model. Moreover, the electric attributes can also link to attributes in other domains.

Furthermore, the motor itself is an actuator which provides a rotational velocity input to drive the rotor. The motor speed is controlled by the input signals voltage and torque. Therefore, the attributes of ":port-in-1" and ":port-in-2" specify the ports left for the control signals while the ":port-to" determines the output signal to the rotor (see Figure 3.14 and 3.19). Hence, the class definition for the motor not only includes geometric and dynamics information but also the electric properties and the method to connect with the controller (or other components). Other E/E components, like sensors, can be treated in a similar fashion.


Knowledge in Multiphysics Information Model
Component Selection
Assembling Reference
Extract Values from Database for Model Initialization
Kinematic Properties
Connection Ports

```
(motor
 ;; information for geometric modeling
 :index (nth 0 (the motor-library)) ; the first motor in library
 :brep (the motor-step (breps 0))    ; geometric information
                                     ;     from STEP file
 :type 'transformed-solid ; generate geometry based on existing model
 :to-location (translate (the arm end)
                         :up (half (get-z (subtract-vectors
                                            (nth 0 (the bounding-box))
                                            (nth 1 (the bounding-box))))))
 ; move the motor center to the endpoint of the arm
 ;     and shift up half of the height
 ;; information for dynamic modeling
 :motor-R (nth 0 (the motor-data)) ; electric resistance
 :motor-K (nth 1 (the motor-data)) ; electromotive force constant
 :motor-L (nth 2 (the motor-data)) ; electric inductance
 :motor-J (nth 3 (the motor-data)) ; moment of inertia of
                                   ;    the rotor
 :motor-b (nth 4 (the motor-data)) ; motor viscous friction constant
 :mass    (nth 5 (the motor-data)) ; motor mass
 :port-in-1 "Voltage" ; input signal voltage
 :port-in-2 "Torque"  ; input signal torque
 :port-to   "Speed")   ; output speed
```

Figure 3.19: The source code of the multiphysics information model (MIM) for the motor

INTEGRATION OF COMPONENTS FROM DOMAIN LIBRARIES

Compared to the arm and motor, the rotor is a special component in the sense that there is a local design and analysis cycle. Every time when a different airfoil is selected for the rotor (the attribute of ":airfoil" in Figure 3.14), the aerodynamic characteristics are obtained from a database. In order to calculate the forces and moments acting on the complete rotor, a dedicated aerodynamic analysis module must be used. For the current application, the blade element method, combined with a uniform inflow model is appropriate. Which type of aerodynamic model (fidelity) should be used is determined by the simulation expert. This feature reduces the difficulty to model the complex systems in multiple domains.

3.4.5. INSTANTIATION OF THE MULTIPHYSICS INFORMATION MODEL

EXTRACTION OF THE CLASS ATTRIBUTES

The first step is to extract the attributes from object definitions. Take dynamics modeling as an example, a function is programmed to complete this task within the intelligent modeling system. The dynamics modeling needs information in terms of the hierarchical structure, topology structure and kinematics. As shown in Figure 3.20, there are four arm assemblies at the top-level. Each of them is composed of three objects: arm, rotor and motor. First, the function loops for each arm assembly to collect the kinematic properties. It starts from

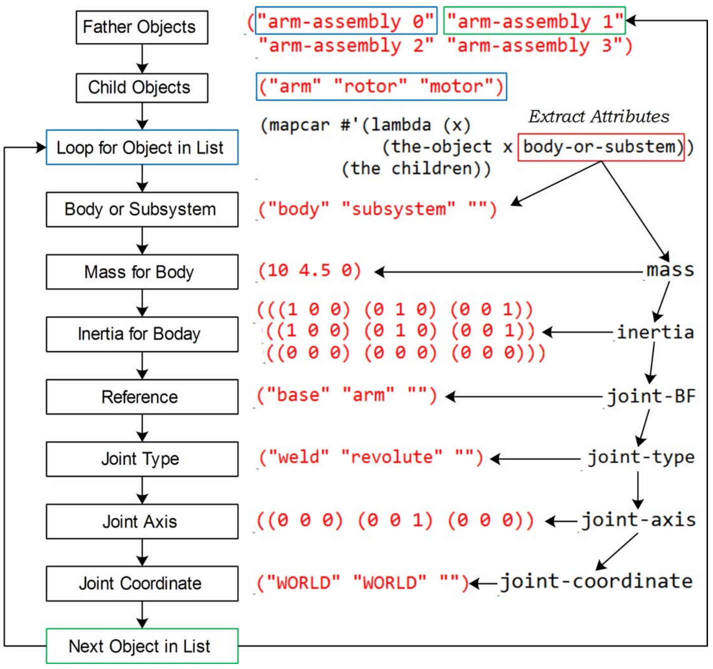
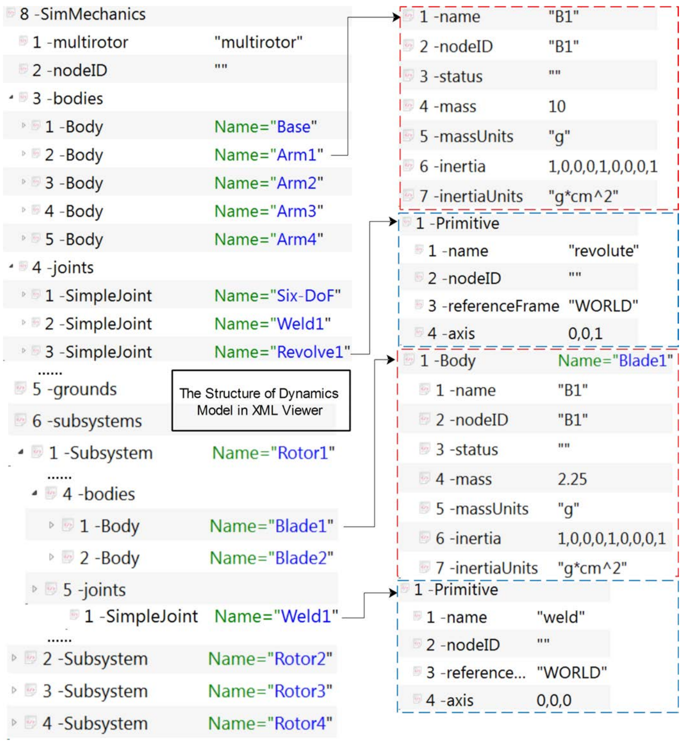


Figure 3.20: Extraction of the object attributes to write the XML file by the intelligent modeling system

the first object “arm-assembly 0” in the sequence. Then, it collects the values of “:body-or-subsystem” to determine the object type. In this example, the arm and the rotor are the body and the subsystem, respectively. The motor is modeled by state functions in the library. Thus, there is no type specified for the motor. Second, the mass and inertia are calculated based on the geometries for the objects. Third, the knowledge related to the joints is also sequentially captured from the objects. When the current object “arm-assembly 0” is completed, the function writes the collected knowledge in strings based on the format specified by the MATLAB and then loops for next object in the object list. Finally, a complete XML file is generated by the intelligent modeling system. An example XML file for the dynamics modeling aspects is presented in Figure 3.21. This file can be used as input for a dedicated MATLAB script to produce the multibody dynamics model.

INSTANTIATION OF THE COMPONENTS

The geometric modeling kernel is SMLib [121]. MATLAB Simscape is used as environment for development of the multiphysics simulation model for vehicle performance evaluation. The intelligent modeling system extracts the dynamics



```
8 -SimMechanics
  1 -multirotor          "multirotor"
  2 -nodeID              ""
  3 -bodies
    1 -Body              Name="Base"
    2 -Body              Name="Arm1"
    3 -Body              Name="Arm2"
    4 -Body              Name="Arm3"
    5 -Body              Name="Arm4"
  4 -joints
    1 -SimpleJoint       Name="Six-DoF"
    2 -SimpleJoint       Name="Weld1"
    3 -SimpleJoint       Name="Revolve1"
  ......
  5 -grounds
  6 -subsystems
    1 -Subsystem         Name="Rotor1"
    ......
      4 -bodies
        1 -Body          Name="Blade1"
        2 -Body          Name="Blade2"
      5 -joints
        1 -SimpleJoint   Name="Weld1"
    ......
    2 -Subsystem         Name="Rotor2"
    3 -Subsystem         Name="Rotor3"
    4 -Subsystem         Name="Rotor4"
```

```
1 -name              "B1"
2 -nodeID            "B1"
3 -status            ""
4 -mass              10
5 -massUnits         "g"
6 -inertia           1,0,0,0,1,0,0,0,1
7 -inertiaUnits      "g*cm^2"
```

```
1 -Primitive
  1 -name            "revolute"
  2 -nodeID          ""
  3 -referenceFrame  "WORLD"
  4 -axis            0,0,1
```

```
1 -Body              Name="Blade1"
  1 -name            "B1"
  2 -nodeID          "B1"
  3 -status          ""
  4 -mass            2.25
  5 -massUnits       "g"
  6 -inertia         1,0,0,0,1,0,0,0,1
  7 -inertiaUnits    "g*cm^2"
```

```
1 -Primitive
  1 -name            "weld"
  2 -nodeID          ""
  3 -reference...    "WORLD"
  4 -axis            0,0,0
```

Figure 3.21: The XML file produced by the intelligent modeling system for dynamics modeling

attributes from the classes and generates an XML file. The XML file can be compiled in MATLAB Simscape to construct the multiphysics simulation model for further evaluation. The MIM of the arm assembly are firstly instantiated as components.

As is shown in Figure 3.22, the arm is instantiated as a cone in the geometric aspect and a body block in MATLAB for the dynamics. The body block has two ports which are specified by the joint definition in the MIM (Figure 3.18). Moreover, the geometric shape of the rotor is instantiated from the aerodynamic configuration (Figure 3.17), especially for the definition of the airfoil. There is a local design and analysis cycle in this case. The aerodynamic forces and torque are searched from the database based on the shape of the airfoil. Then, the results are saved as tables which are used to initialize the blade model in the dynamics analysis. It shows that the intelligent modeling system organizes the analyses in a proper structure by capturing the expertise from simulation experts (Figure 3.6). Furthermore, because the motors are standard components, their characteristics are specified by the manufacturer. It is assumed that the manufactures provide the geometries and other

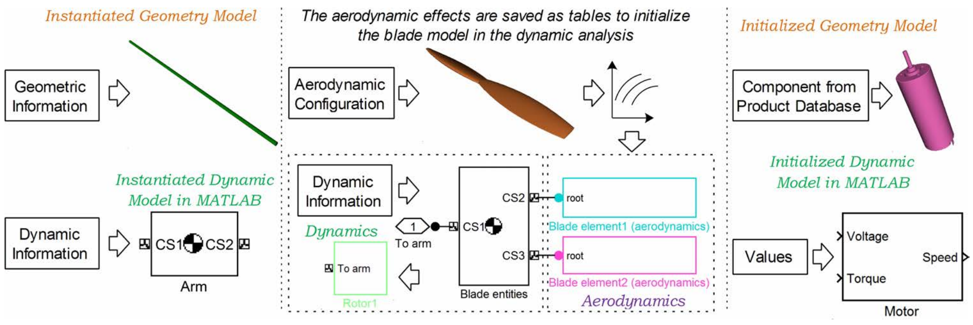

Instantiated Geometry Model
Geometric Information
Instantiated Dynamic Model in MATLAB
Dynamic Information
CS1
CS2
Arm
The aerodynamic effects are saved as tables to initialize the blade model in the dynamic analysis
Aerodynamic Configuration
Dynamic Information
Dynamics
To arm
Rotor1
1
To arm
CS1
CS2
CS3
Blade entities
root
Blade element1 (aerodynamics)
root
Blade element2 (aerodynamics)
Aerodynamics
Initialized Geometry Model
Component from Product Database
Initialized Dynamic Model in MATLAB
Values
Voltage
Torque
Speed
Motor

Figure 3.22: The instantiated and initialized components from the multiphysics information model (MIM)

characteristics for the motors. Thus, in this example it is not necessary to model the motors in the MIM in detail. The MIM only has to store the serial numbers and the characteristics of the motors in a data file which can be used for initializing the motor models.

Besides modeling the geometries and dynamics characteristics, the multiphysics simulation model of the multirotor UAV also consists of a propulsion system (motors), electrical energy supply (battery packages), aerodynamic components (inflow model), embedded control systems (automatic flight controls) and scripts required for executing simulations (trim algorithm, definition of test maneuvers, flight conditions). Because all of these components and simulation scripts are independent from the topology structure of the multirotor UAV, they are predefined in separate simulation libraries. When the top level configuration is determined by the user, the inference engine uses index numbers to search for the components in the libraries and requests the dynamics modeling kernel to initialize them.

Instantiation of the Submodels

The instantiated/initialized components are assembled to construct a complete arm assembly model, which is shown in Figure 3.23. As for the mechanical engineering, the components are connected by the assembling references. For example, the motor and rotor are positioned by referring to the endpoint of the arm and the shaft of the motor, respectively. On the contrary, the relative motion are more critical than the geometric constraints for the dynamics analysis. As is shown in Figure 3.23, the rotor is driven by the motor to rotate around the Z axis of the endpoint of the arm in the view of dynamics despite the fact that the rotor is fixed on the shaft of the motor in real world. Therefore, the dynamics components are linked by the joints in the dynamics domain. In MATLAB, the model blocks are linked by the associated ports. The source code

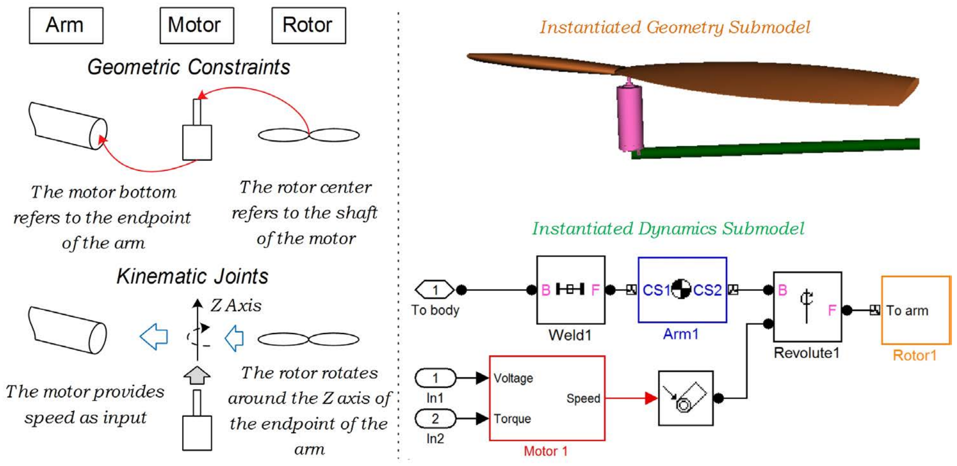

Arm
Motor
Rotor
Geometric Constraints
The motor bottom refers to the endpoint of the arm
The rotor center refers to the shaft of the motor
Kinematic Joints
Z Axis
The motor provides speed as input
The rotor rotates around the Z axis of the endpoint of the arm
Instantiated Geometry Submodel
Instantiated Dynamics Submodel
1
To body
B
F
Weld1
CS1
CS2
Arm1
Revolute1
To arm
Rotor1
1
In1
2
In2
Voltage
Torque
Speed
Motor 1

Figure 3.23: The instantiated submodels both in mechanical engineering and dynamics domain

of the MIM also contains the attributes to specify the ports for the objects. As can be seen in Figure 3.17, 3.18 and 3.19, one port is used to connect with parent component, such as the ":port-ref" for the rotor and arm. Therefore, when the MIM is instantiated as components, it is possible to integrate with more components from multiple domains, like a controller or the inflow model, by connecting the ports of model blocks.

3.4.6. CONSTRUCTION OF THE MULTIPHYSICS SIMULATION MODEL

In order to construct the overall multiphysics simulation model, the intelligent modeling system has to set up a connect with MATLAB through the communication framework. As can be seen in Figure 3.24, the intelligent modeling system generates a script file which contains the structure of the multiphysics simulation model based on the knowledge and specifications of the simulation expert. Then, it sets up connections with MATLAB and imports the submodels produced previously.

Next, the multiphysics simulation model of the physical plant is constructed by integrating the dynamics model together with the components from other domains. These are the atmospheric model (environment library), rotor inflow model (aerodynamics library), motor model (propulsion library) and the lookup tables for computing the sectional aerodynamics forces and moments acting on a blade element (aerodynamics library). The inputs are motor speed commands and the outputs are roll, pitch, yaw angles and vertical position. An impression of the multiphysics simulation model for a quadrotor UAV is shown in Figure 3.25. Because the XML file of the arm assembly can be repeatedly written by the

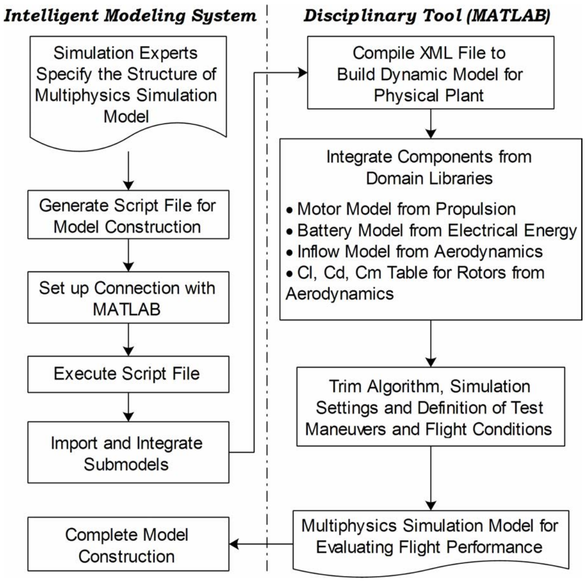

Intelligent Modeling System
Disciplinary Tool (MATLAB)
Simulation Experts Specify the Structure of Multiphysics Simulation Model
Generate Script File for Model Construction
Set up Connection with MATLAB
Execute Script File
Import and Integrate Submodels
Complete Model Construction
Compile XML File to Build Dynamic Model for Physical Plant
Integrate Components from Domain Libraries
• Motor Model from Propulsion
• Battery Model from Electrical Energy
• Inflow Model from Aerodynamics
• Cl, Cd, Cm Table for Rotors from Aerodynamics
Trim Algorithm, Simulation Settings and Definition of Test Maneuvers and Flight Conditions
Multiphysics Simulation Model for Evaluating Flight Performance

Figure 3.24: Construction of multiphysics simulation model by the intelligent modeling system

intelligent modeling system, it is possible to construct the multiphysics simulation model for the multirotor UAV with N rotors.

3.5. SUMMARY

E/E systems should be concurrently developed with other engineering disciplines in the design of complex engineering products such as aircraft. Hence, this is an multidisciplinary design problem. MDO is a promising technique to approach this problem. To support the MDO process, various researchers have introduced methods and tools that enable the use of mono disciplinary analysis tools within an MDO framework without user interference. Other research studies have focused on how to transform models from one discipline to another. For the concurrent design of E/E systems, however, multiphysics simulation models are required. Maintaining consistency between parameters, design variables etc. across different engineering disciplines is essential, especially for the development of control software of E/E systems.

Therefore, an intelligent modeling system is proposed that can automatically generate multiphysics simulation models to support the MDO process by using a KBE approach. This system has to overcome three challenges. First, it must deal with the problem of system coupling and interactions across disciplines in the process of multiphysics simulation model generation. Second, it has to represent design knowledge from multiple

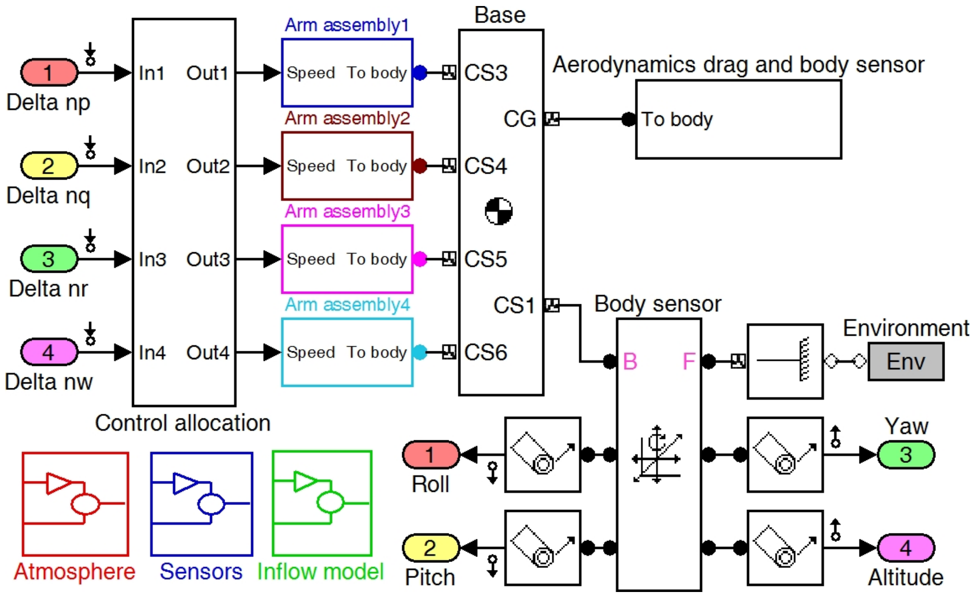


Figure 3.25: Impression of the multiphysics simulation model for a quadrotor UAV

engineering domains in an appropriate format that ensures consistency across the different disciplines, especially when considering the use of control software for E/E systems and its relation with the physical system. Third, it must be able to generate multiphysics simulation models with the right level of fidelity for the problem at hand. This can be low, medium or high fidelity, depending on; which parameters need to be calculated, which accuracy of the result is required, which computational speed is needed within the design optimization and finally depending on which parameters are available.

The intelligent modeling system is an application which consists of three software modules, responsible for; (1) knowledge acquisition (the "inference engine"), (2) knowledge application (the "MIM") and (3) the communication with analysis tools (the "communication framework"). How these three modules provide solutions for the above challenges is discussed in detail.

It may be required that mono disciplinary analysis tools are used before the multiphysics simulation model can be developed. The intelligent modeling system specifies the sequence of these analyses by collecting knowledge from e.g., textbooks, simulation experts and design experts. A calculation sequence may also be required within the multiphysics simulation model. This sequence is also defined by the system. Next, connections are created by the communication framework with target tools and the analyses are sequentially performed. Moreover, in order to select appropriate components from simulation libraries, the intelligent modeling system defines logical expressions to capture the expertise from simulation experts.

To ensure consistency, a unified knowledge representation, the MIM, is used. The different characteristics of the complex system to be designed are represented in this information model. Physical entities of the product are modeled as classes and non-physical elements (such as control software) are modeled with functions. Every class (or function) is assigned with attributes (or variables) to represent different aspects of the physical entities (or non-physical information). The challenges of system coupling and the interactions across disciplines are taken into account by means of the knowledge acquisition process.

Next, the intelligent modeling system instantiates the MIM into submodels with the modeling kernels. Finally, the intelligent modeling system constructs the multiphysics simulation model. This model is composed of the physical plant, control systems and files required for executing the simulations, such as trim algorithms, simulation configurations, etc.

The methodology is demonstrated by the automatic generation of multiphysics simulation models for a multirotor UAV. There are five coupled disciplines in this use case. It is assumed that a top-level configuration is prescribed by the design expert. This top level configuration, the design space, is the input to the proposed KBE system. For illustration it is shown how; (1) the system selects the appropriate control architecture when the topology of the multirotor UAV is altered and (2) how the level of fidelity of the rotor inflow model is selected.

After the knowledge acquisition phase, the intelligent modeling system successfully constructs the multiphysics information model. It defines classes with attributes to capture geometric information. These classes also have attributes required for modeling of the dynamics, such as rigid bodies, joints, mass and inertia. Other design knowledge is modeled in data files. Finally, submodels are instantiated and these are integrated into a complete multiphysics simulation model. In addition, script files needed to perform simulations are also created. For this test case, MATLAB Simscape is used as environment for multiphysics simulation. The final model can be used to evaluate the flight performance of the multirotor UAV when executing complex maneuvers. In this chapter, the modeling of the physical elements of the multiphysics simulation model is treated. The control system development also plays a key role for this test case. This development and the simulation of maneuvers is treated in the next chapter.

4

Automated Control System Design to Enable Flying Qualities Evaluation in MDO

4.1. Introduction

In order to evaluate the performance of the overall system, including E/E systems, it is necessary to concurrently design the control system which regulates the behavior of physical plant. In the previous chapter, an intelligent modeling system has been introduced that can automatically generate a multiphysics simulation model, excluding its control system. This intelligent modeling system was used to model the physical entities of a multirotor UAV, in terms of its geometry, dynamics, aerodynamics, propulsion, electrical power supply, etc. However, in order to evaluate the flight performance of this inherently unstable UAV, a control system is required that regulates the behavior of the multiphysics simulation model.

As explained in Chapter 1, electronic control systems used on present day passenger cars can be extremely complex. For inherently unstable and unmanned aerial vehicles, even more functions are electronic controlled. One crucial element of the complete E/E systems on an aircraft is the automatic flight control system. In order to achieve desired (level 1) and consistent handing qualities throughout the operational flight envelope, the design of a flight control system for unstable and unmanned aircraft is a difficult task which results in substantial cost and time [36]. The variables of the control system are related to the properties of physical plant and the architecture of the control system is related to the configuration of the design. This poses a challenge when top level configuration changes of the physical design are made since the control system architecture has to be modified and variables have to be tuned again. Within an MDO process this can occur thousands of times. Furthermore, even though there is a direct relation between the physical elements of the design and the control system parameters, there is not a one to one mapping between physical design parameters and variables of the control software. Therefore, the design of control system is not as straight forward as the modeling of physical entities. This is especially the case for the unstable and unmanned aerial vehicle with highly non-linear characteristics. It takes much time and effort to the configure the simulation model, determine an appropriate control system architecture and control method, tune the parameters of the control system and to ensure a consistent control system throughout the whole design envelope (which can be in the order of thousands designs).

Performing simulations with the multiphysics simulation model is a highly repetitive activity during the whole development process of the E/E systems. When the physical design is changed, simulation components representing the physical elements of the design have to be reconfigured. The initial conditions of the simulation model, e.g., the equilibrium position at the start of a maneuver, have to be recalculated as well. Appropriate test maneuvers have to be defined based on the current configuration of the design and the top level requirements. Script files must be developed that interpret all the simulation results. Finally, script files must be developed that can be used to determine linear models of the physical plant required for control system design. For the analysis and tuning of the control system, typically a large set of simulations must be conducted as well. All these repetitive processes are time consuming and prone to errors.

Model based inversion control technique allows the engineer to develop ideal controller by inversing the physical simulation model [118, 119]. An additional benefit is that if a controller is inversed from the physical plant, it is not necessary to tune the control variables. It is possible to use this technique because it potentially saves time and effort for the development of the control system when many configurations have to be evaluated, especially within the conceptual and preliminary design phase. However, the key to obtain a model based inversion control system is a set of sequential procedures, which is a different approach from that required for modeling the physical systems (Chapter 3). When applied on a real vehicle, linear model based inversion control can have robustness problems. This is an important disadvantage of the approach. However, that is not an issue when applied in a purely simulation based MDO framework. Unlike some multi-variable feedback control techniques, a model based inversion control system does not solve the control allocation problem. This problem deals with the question how to distribute a control effort over a set of actuators. For example, which physical flight control surfaces on a fixed wing aircraft should move when a pitch command is given. So, a separate control allocation scheme or algorithm has to be selected, ideally one that can also easily be applied within an MDO framework.

Because the procedures required to create the overall automatic flight control system, a model based inversion control system combined with a control allocator, to support MDO of unstable and unmanned aircraft, can be viewed as a special model type, it is possible to automate it by using the KBE approach. In this chapter, the intelligent modeling system is extended to include these procedures. The main challenges are summarized below:

- Specification of strategies and parameters for the development of control systems;
- Configuration of simulation initial conditions and other simulation settings for the entire design envelope (automated simulation);
- Automated design of control system for the E/E systems.

This chapter is structured as follows. First, a brief review of model based inversion control and control allocation is presented. Second, the methodology required for control system development process automation by the intelligent modeling system is covered. Finally, as test case, the intelligent modeling system is used to automatically design the flight control system for multirotor UAVs.

4.2. A Brief Review of Model Based Inversion Control and Control Allocation

The model based inversion control is widely used and proven useful for ensuring a consistent design envelope for the control system. Typically, the physical system is viewed as a black box, which is driven by the inputs and produce the response based on its inherent characteristics. As can be seen in Figure 4.1, through inversion of the physical system, the desired performance can be used as the inputs and reversely calculated the new inputs to feed the original system. Thus, it is not necessary to tune the parameters when using the inversion control method because the relationship from the inputs to outputs has been captured by the inversion controller. The only challenge is how to get the inversion model of the original system, which can be solved for most engineering applications.

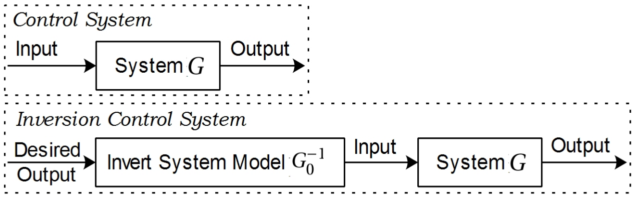


Figure 4.1: Schematic diagram of inversion control system

Sahani and Horn [119] describe the detailed process how the model based inversion control technique can be applied on multirotor helicopter. Devasia et al. [122] applies the inversion-based control approach to construct a bounded input trajectory for solving tracking control problems. Looye and Joos [123] use the dynamic inversion approach to obtain a linear outer loop controller. The controller is then tuned by multiobjective optimization method. Moreover, Tandale and Valasek [124] propose a fault-tolerant structured adaptive model inversion control, which is demonstrated for the problem of fault tolerance to actuator failures on redundantly actuated systems. Furthermore, the model inversion control has been applied for nonlinear tracking problem, such as missiles by Ru et al. [125] and Kim and Jang [126]. Further, adaptive model inversion control is proposed to integrate the inversion control with neural network and tested for helicopter [119], tilt-rotor aircraft [127] and twin rotor [128].

It should be pointed out that model based inversion control has some disadvantages when applied in a realistic environment. It is possible that the revision model of the system can't reflect the actual relationship from the outputs to the inputs due to non-linear characteristics. Because it is not straightforward to revise the actual system, it is therefore possible to design a different type of control law for the final design. This control law only needs to be tuned once.

On the other hand, Petersen and Bodson [129] propose an interior-point algorithm for control allocation, which is tested for computational efficiency and accuracy using linear models of aircraft. Other researchers also propose methods for solving the control allocation problem, such as a weighted pseudo-inverse method by Bordignon [130], a fixed-point iteration method by Roberts and Sutton [131]. Bodson [132] also formulates two control allocation problems: a direct allocation method and a mixed optimization methods. In this research study, the power distribution for the multirotor UAV is allocated through direct Daisy chain, which is simple but suitable for our test case.

4.3. SYSTEM MODELING

The system G in Figure 4.1 is the flight dynamics model of multirotor UAV. Bouabdallah and Siegwart derive the equations of motion of a quadrotor UAV (equation 4.1 [133]) based on Newton-Euler formalism [134], where a rigid body under external forces applied to its center of mass and expressed in body coordinate. The equation 4.1 takes a series of forces and moments due to different physical effects into account, such as body gyro effect, propeller gyro effect, actuators action, inertial counter-torque, counter-torque unbalance, friction along the x or y axis, hub forces and moments in forward or sideward flight, etc.

$$\begin{cases} I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h(\sum_{i=1}^{4} H_{yi}) + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi}, \\ I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) + J_r\dot{\phi}\Omega_r + l(T_1 - T_3) - h(\sum_{i=1}^{4} H_{xi}) + (-1)^{i+1}\sum_{i=1}^{4} R_{myi}, \\ I_{zz}\ddot{\psi} = \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) + J_r\dot{\Omega}_r + (-1)^{i}\sum_{i=1}^{4} Q_i + l(H_{x2} - H_{x4}) + l(-H_{y1} + H_{y3}) \\ m\ddot{z} = mg - (c\psi c\phi)\sum_{i=1}^{4} T_i \\ m\ddot{x} = (s\psi s\phi + c\psi s\theta c\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi} - \frac{1}{2}C_x A_c \rho \dot{x}|\dot{x}|^1 \\ m\ddot{y} = (-c\psi s\phi + s\psi s\theta c\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi} - \frac{1}{2}C_y A_c \rho \dot{y}|\dot{y}| \end{cases} \tag{4.1}$$

Compared to a complete multiphysics simulation model in Chapter 3 (see Figure 3.25), the system G in this research study is more about the dynamics model of multirotor UAV and several submodels related to the aerodynamics. The system G model can be more complex when more aerodynamics effects are taken into account. Figure 4.2 shows the system G model of a quadrotor UAV with the inputs of motor speed increments due to commands and the outputs of roll, pitch and yaw attitudes and vertical position. Next, the model based inversion control approach is applied on this model to construct the inversion controllers for a whole design envelope of multirotor UAVs.

[1] *c*: cos, *s*: sin

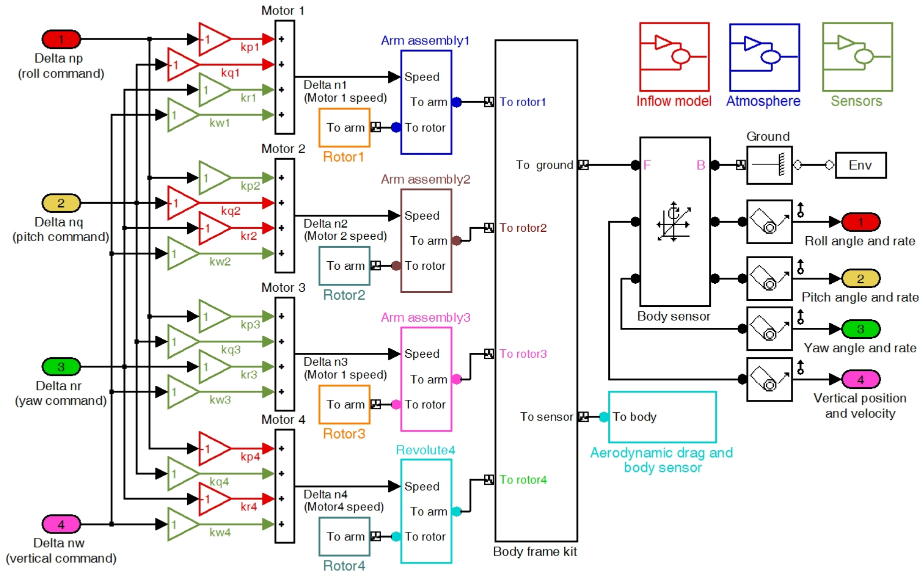


Figure 4.2: The multiphysics simulation model of a quadrotor UAV prepared for model based inversion control approach

4.4. Process Automation for Control System Development

Although model based inversion control is a promising method to accelerate the MDO process, the design of control system still involves many other activities, which is shown in Figure 4.3. Firstly, the inference engine of the intelligent modeling system has to collect related design knowledge, like the strategies for control allocation or trimming. Secondly, the MIM captures the procedure of model based inversion control and generate the simulation model of the physical plant. Thirdly, the intelligent modeling system sets up connections with the analysis tools, imports the simulation model and applies the model inversion method approach through the communication framework.

4.4.1. Knowledge Acquisition for Automated Control System Design

As shown in Figure 4.4, the inference engine of the intelligent modeling system searches the knowledge related to control system design based on the inputs of both simulation experts and control system design experts. First, the control architecture is determined. In this case it is a model based inversion control system combined with a simple Daisy chain approach to control allocation. Second, simulation strategies are determined, such as the trim algorithm used,

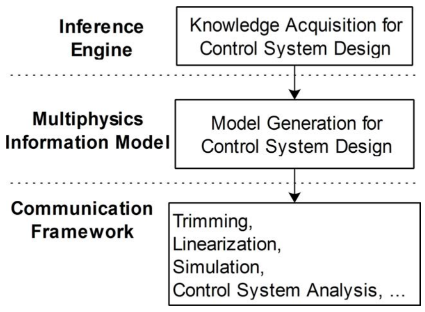

Inference Engine
Knowledge Acquisition for Control System Design
Multiphysics Information Model
Model Generation for Control System Design
Communication Framework
Trimming,
Linearization,
Simulation,
Control System Analysis, ...

Figure 4.3: Process automation by the intelligent modeling system for model based inversion control approach

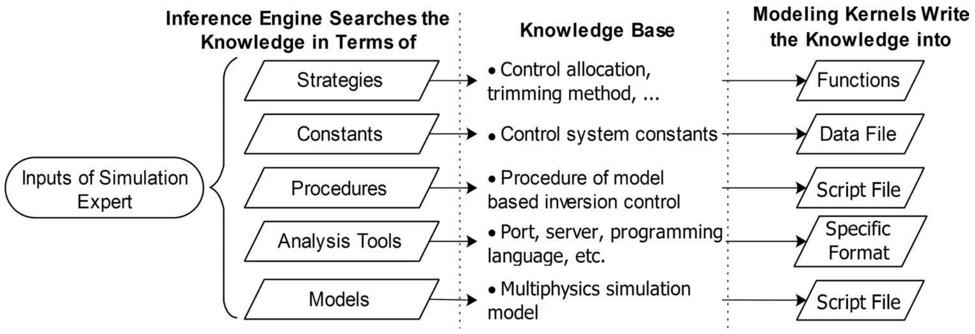

Inference Engine Searches the Knowledge in Terms of
Knowledge Base
Modeling Kernels Write the Knowledge into
Inputs of Simulation Expert
Strategies
• Control allocation, trimming method, ...
Functions
Constants
• Control system constants
Data File
Procedures
• Procedure of model based inversion control
Script File
Analysis Tools
• Port, server, programming language, etc.
Specific Format
Models
• Multiphysics simulation model
Script File

Figure 4.4: Collection of knowledge for control system design by the inference engine

the method for numerical integration of time domain simulations and the method to obtain linear models around a specific operating point. Third, flight conditions (e.g., aircraft weight and c.g., position) and operating conditions (e.g., altitude and turbulence level) are specified. At the same time, the inference engine collects the procedures for developing the control system from the knowledge base. Then, based on the analysis tools to be communicated with, the inference engine searches the knowledge for the communication framework, such as the port, server of the analysis tool. In the meanwhile, the inference engine also specifies the simulation object, such as the dynamic model, which is going to be instantiated from the MIM.

4.4.2. WRITING KNOWLEDGE INTO A SPECIFIC FORMAT

After the knowledge acquisition process conducted by the inference engine, the intelligent modeling system calls the modeling kernels to write the knowledge into specific formats, as shown in Figure 4.4.

Basically, the modeling kernels convert the knowledge of control strategies, constants, procedures and information related to the analysis tool (port, server etc.) into functions, data file and scripts, respectively. The modeling kernels also instantiate the simulation models from the MIM and write them into their specific format. For sake of brevity, the generation of files is not discussed in detail because it is similar to how the modeling kernels generate the geometry model, dynamics model and data file presented in Chapter 3. Finally, all files are sent to the communication framework.

4.4.3. COMMUNICATION WITH ANALYSIS TOOLS

A communication framework is set up to establish the connections between the intelligent modeling system and the target analysis tool. The module not only exchanges the data and information among (distributed) software tools, but also controls the overall process. As mentioned in Chapter 2, a communication framework which connects the intelligent modeling system with MATLAB, has been developed in previous research [33, 92]. This module is built based on the Carlos Ungil's common lisp interface to MATLAB [93].

As can be seen in Figure 4.5, the communication framework starts the target analysis tool through operation system commands. Then, a connection is set up through the common server between the intelligent modeling system and target

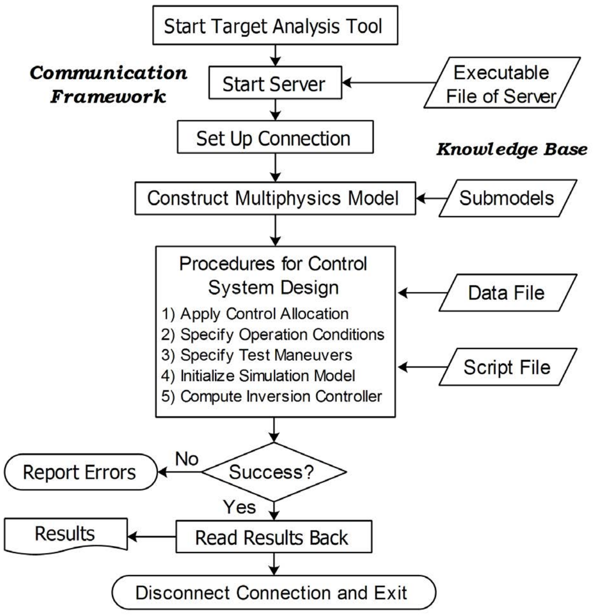

Start Target Analysis Tool
Communication Framework
Start Server
Executable File of Server
Set Up Connection
Knowledge Base
Construct Multiphysics Model
Submodels
Procedures for Control System Design
1) Apply Control Allocation
2) Specify Operation Conditions
3) Specify Test Maneuvers
4) Initialize Simulation Model
5) Compute Inversion Controller
Data File
Script File
Success?
No
Report Errors
Yes
Read Results Back
Results
Disconnect Connection and Exit

Figure 4.5: Communication with analysis tools for control system design

disciplinary tool. Next, the communication framework imports the multiphysics simulation models and performs the procedural files sequentially. Finally, if all procedures are successfully performed, the communication framework disconnects and reads the results back to the intelligent modeling system. Otherwise, a pop-up window with an error message is presented to the user of the system.

In summary, the intelligent modeling system automates the design process by three main steps (Figure 4.3). It will be shown how the intelligent modeling system automatically designs the overall automatic flight control system for the multirotor UAVs.

4.5. AUTOMATED DESIGN OF FLIGHT CONTROL SYSTEM FOR THE MULTIROTOR UAV

4.5.1. MANEUVER TRACKING CONTROL STRATEGY

The simulation experts have to determine which type of control laws are required for a specific physical system, which is similar to the component selection in Chapter 3. The multiphysics simulation models of the multirotor UAVs will be tested in several flight maneuvers to evaluate the handing qualities. A maneuver tracking control system is necessary to guide the simulation models along the maneuvers within boundaries and time limit. Proportional plus velocity plus acceleration (PVA) control is the most common control strategy applied in practice [135]. It is also practicable to helicopters. In this research study, a maneuver tracking controller [136] is derived from the PVA approach. An overview of the flight control system is shown in Figure 4.6. For instance, an input command u_x which controls the vehicle to move in the X direction is represented by:

$$u_x = K_x e_x + K_{vx}\Delta\dot{x} + K_{ax}\Delta\ddot{x} \tag{4.2}$$

$$e_x = x - x' \tag{4.3}$$

The displacement in X, Y, Z direction (X', Y', Z') and yaw motion (D') constitute the feedback which are sequentially deducted by the desired response (X, Y, Z, D) and their derivatives. The errors are scaled by gains and then added together as the command signals for pitch, roll, yaw motion and altitude control.

4.5.2. CONTROL ALLOCATION

It is straightforward to change the topology structure of multirotor UAV from one to N rotors in the web-based interface. However, few multirotor UAVs have odd rotors because extra mechanism are required to balance the yaw forces.

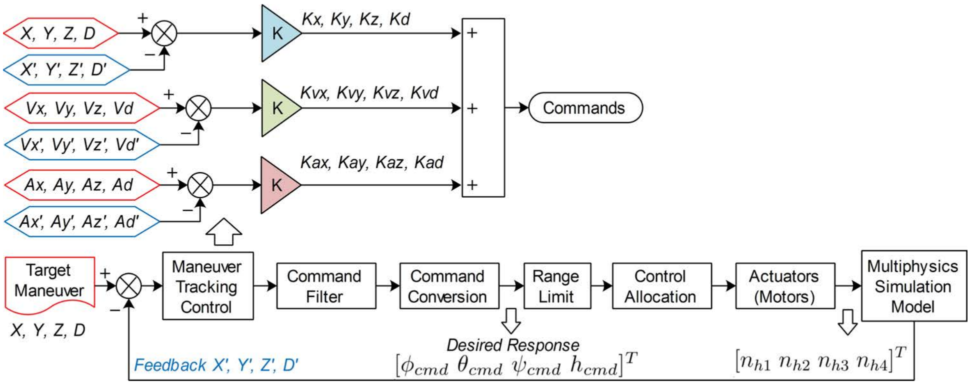


Figure 4.6: The overview of the maneuver tracking control system architecture for a quadrotor UAV

Moreover, the multirotor UAV need four degrees of freedom to control yaw, pitch, roll and thrust. Therefore, the quadrotor UAV is the most common multirotor UAV both in the research area and the market.

Take quadrotor UAV and sixrotor UAV for example, the control allocation for the rotors is quite different for these configurations. As can be seen in Figure 4.7, the rotors on the diagonal for the quadrotor UAV rotate in clockwise direction and the other two rotate in the opposite direction. As for the sixrotor UAV, the rotors on the diagonal and the adjacent rotors rotate reverse direction. Thus, the motion control for the quadrotor UAV and the sixrotor UAV are also different. Typically, the motor speed Δn is calculated by adding the speed of increment due to the commands of roll Δn_p, pitch Δn_q, yaw Δn_r and altitude Δn_w. The coefficients of k_{np}, k_{nq}, k_{nr} and k_{nw} decide whether the rotor speed is increasing or decreasing due to the input commands.

$$\Delta n = k_{np} \times \Delta n_p + k_{nq} \times \Delta n_q + k_{nr} \times \Delta n_r + k_{nw} \times \Delta n_w \tag{4.4}$$

In order to pitch or roll, two rotors' speed are increasing while the other two are decreasing for the quadrotor UAV, which is shown in Figure 4.7. However, when the sixrotor UAV pitches, the rotor 1 and 2 decrease the speed while the rotor 4 and 5 increase the same amount and the rotor 3 and 6 keep the hovering speed. When the sixrotor UAV rolls, all the rotor 2, 3, 4 decrease the speed and the rotor 1, 5, 6 increase. In other words, the pith and roll motion are unequal for the sixrotor UAV, which makes the control system even more difficult than the quadrotor UAV.

Take pitch as an example, two functions are proposed to compute the motor speed (also the rotor speed) for the multirotor UAV with N rotors. The number of rotors N is variable. Two distinct classes of multirotor UAVs can be defined,

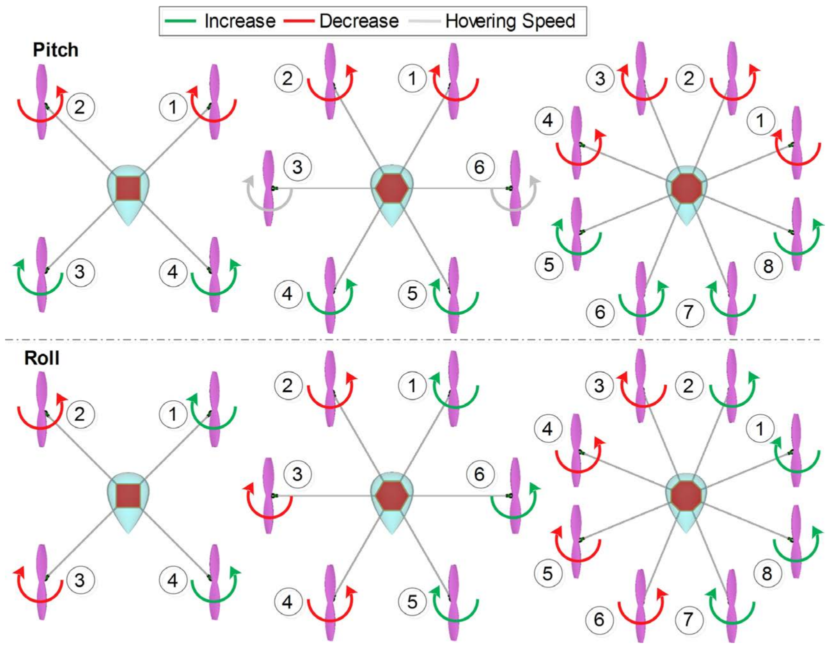
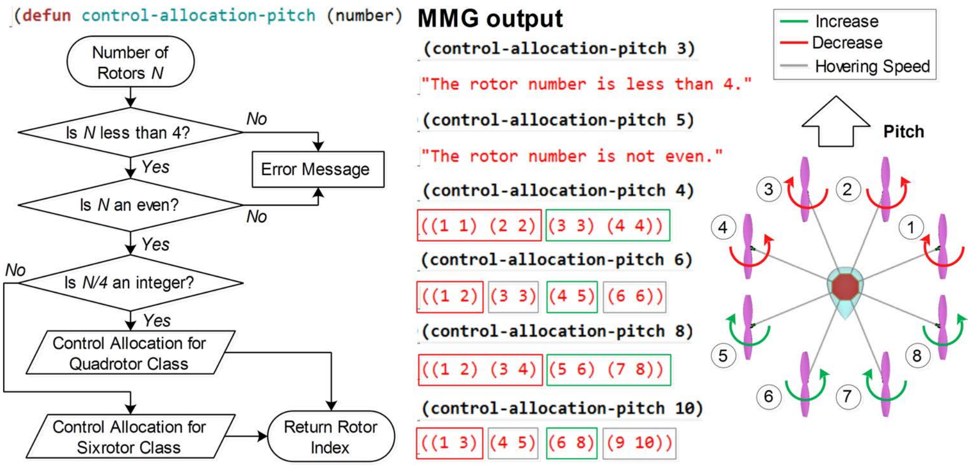


Figure 4.7: The control allocation strategies for the multirotor UAVs with four, six or eight rotors

the class of quadrotor UAVs (multiples of four rotors - 4, 8, 12) and the class of sixrotor UAVs (all other configurations with an even number of rotors). As can be seen in Figure 4.7, the rotors are numbered in anticlockwise direction with index number i which is computed by equation 4.5. Finally, the rotor index is multiplied with coefficient vector in equation 4.6 (or equation 4.7) to decide whether the rotor is increasing or decreasing due to pitch commands. For example, the coefficient of the first rotor is -1, which means the speed of the first rotor is always decreasing due to pitch commands, no matter it's quadrotor UAV or six rotor UAV.

$$i = \frac{N}{4} \tag{4.5}$$

$$\Delta n_q = \begin{pmatrix} -1 & -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} r_{i=1,\ldots,i} \\ r_{i=i+1,\ldots,2i} \\ r_{i=2i+1,\ldots,3i} \\ r_{i=3i+1,\ldots,4i} \end{pmatrix} \tag{4.6}$$



```
(defun control-allocation-pitch (number)
```

```
(control-allocation-pitch 3)
"The rotor number is less than 4."
(control-allocation-pitch 5)
"The rotor number is not even."
(control-allocation-pitch 4)
((1 1) (2 2) (3 3) (4 4))
(control-allocation-pitch 6)
((1 2) (3 3) (4 5) (6 6))
(control-allocation-pitch 8)
((1 2) (3 4) (5 6) (7 8))
(control-allocation-pitch 10)
((1 3) (4 5) (6 8) (9 10))
```

Figure 4.8: Specification of control allocation strategies for the multirotor UAVs with four or six rotors

$$\Delta n_q = \begin{pmatrix} -1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{i=1,\ldots,i+2} \\ r_{i=i+3,\ldots,2i+3} \\ r_{i=2i+4,\ldots,3i+5} \\ r_{i=3i+6,\ldots,4i+6} \end{pmatrix} \tag{4.7}$$

The intelligent modeling system uses a function in the MIM to capture the strategy of control allocation due to pitch commands for the quadrotor UAVs and sixrotor UAVs, which is shown in Figure 4.8. Basically, the rotors are classified into four groups which are multiplied with the coefficient vector in equation 4.6 or equation 4.7. Firstly, error messages are given when the number of rotors is less than 4 or not an even. Secondly, if the $N/4$ is an integer, equation 4.6 is used to calculate the index number for the rotors of the quadrotor UAV, otherwise equation 4.7 is selected for the sixrotor UAV. The function of "control-allocation-pitch" in the MIM is tested for various inputs. It returns a list of lists which indicates the start and end rotor index for each group.

Therefore, the intelligent modeling system can allocate rotors for the multirotor UAV with eight, ten, or N rotors by capturing the knowledge of control allocation. This is to say, the intelligent modeling system can assign the control strategies for the complex system with different configurations or topology structures.

4.5.3. ASSIGNMENT OF CONSTANTS FOR CONTROL SYSTEM

It is also necessary to assign values for the parameters and constants. For example, although it is possible to give an step command to the multirotor UAV, the actual input is calculated by a transfer function (equation 4.8) which considers the system delay due to assembly clearances and other factors. Other parameters should also be considered, such as the motor speed, altitude, initial velocity, etc.

$$\frac{1}{0.25s+1} \tag{4.8}$$

As mentioned in the methodology, the inference engine collects the values for the parameters and constants from the knowledge base according to the inputs of simulation and control system design experts. Then, the values can be written by the modeling kernels into data format, such as a CSV file. The CSV file, e.g., contains the flight condition (altitude, speed) and aircraft configuration (weight, c.g., etc.) for which the trim solution must be found.

4.5.4. AUTOMATED TRIMMING FOR SPECIFIED DESIRED FLIGHT CONDITION

Trimming is a process to find specified desired starting point for the simulation, where a dynamic system is in a steady state. Needless to say, trimming is a repetitive process. Although the software tool, like MATLAB, can help us to search for the trim points, the simulation expert still has to specify the inputs, outputs, states and state derivatives and conditions for the system. Usually, the simulation expert makes an initial equilibrium guess for the trimming functions in MATLAB and check the response of the system. This process is repeated until a nontrivial equilibrium point is found.

By capturing the procedural information of trimming, the intelligent modeling system can automatically find the flight conditions for many design configurations. First, as can be seen in Figure 4.9, the inference engine specifies desired flight condition based on the top-level configuration provided by the simulation expert. For example, according to the configuration of multirotor UAV, the inference engine can calculate the motor speed when it is hovering at specific altitude. Secondly, the MIM provides the information required to generate the physical plant model with a dedicated MATLAB script. Third, the multiphysics model are constructed in the analysis tools and trimmed to check whether the desired the flight conditions are met. If the nontrivial equilibrium point is found, the results are saved for the further model linearization. Otherwise, the whole process is repeated from the inference engine again. Typically, the quality of trimming is ensured by analysis tools. The intelligent modeling system is responsible for determining an initial flight condition based on current configuration and automation of the whole process for the entire design envelop.

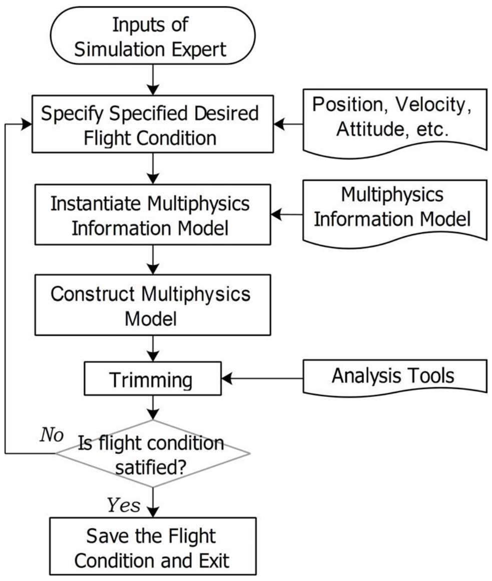

Inputs of Simulation Expert
Specify Specified Desired Flight Condition
Position, Velocity, Attitude, etc.
Instantiate Multiphysics Information Model
Multiphysics Information Model
Construct Multiphysics Model
Trimming
Analysis Tools
No
Is flight condition satified?
Yes
Save the Flight Condition and Exit

Figure 4.9: Automated trimming by the intelligent modeling system

4.5.5. INTEGRATION OF MODEL BASED INVERSION CONTROL

The maneuver tracking control system has 12 gains in Figure 4.6. If all the gains are manually tuned, it will take too much time to support the MDO process. Model based inversion control is suggested by researchers as a solution to design consistent control system without having to tune the gains [118, 119]. In order to clearly explain how the intelligent modeling system automates the whole process, the design of the control system for the multirotor UAV with four rotors (quadrotor UAV) is discussed as an example. First, a simple low order linear model of the quadrotor is required:

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{w} \end{pmatrix} = A \begin{pmatrix} p \\ q \\ r \\ w \end{pmatrix} + B \begin{pmatrix} \Delta n_p \\ \Delta n_q \\ \Delta n_r \\ \Delta n_w \end{pmatrix} \tag{4.9}$$

The linear model, consisting of the A and B matrix, is obtained in three consecutive steps. First, the nonlinear simulation model is trimmed in a predefined flight condition. Second, a high order linear model is obtained by numerical perturbation of the nonlinear simulation model. Finally, the order of the high order linear model is reduced. The inputs to the low order linear model of the quadrotor UAV model are the motor speed commands. The states of the

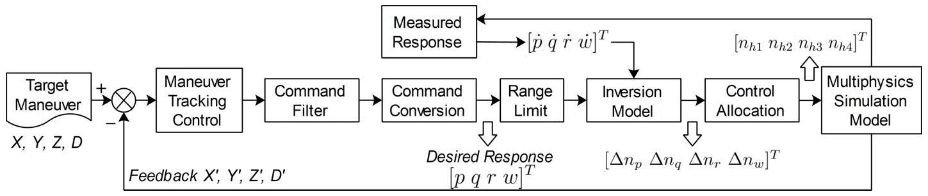

Measured Response
$[\dot{p}\ \dot{q}\ \dot{r}\ \dot{w}]^T$
$[n_{h1}\ n_{h2}\ n_{h3}\ n_{h4}]^T$
Target Maneuver
X, Y, Z, D
+
−
Maneuver Tracking Control
Command Filter
Command Conversion
Range Limit
Inversion Model
Control Allocation
Multiphysics Simulation Model
Desired Response
$[p\ q\ r\ w]^T$
$[\Delta n_p\ \Delta n_q\ \Delta n_r\ \Delta n_w]^T$
Feedback X', Y', Z', D'

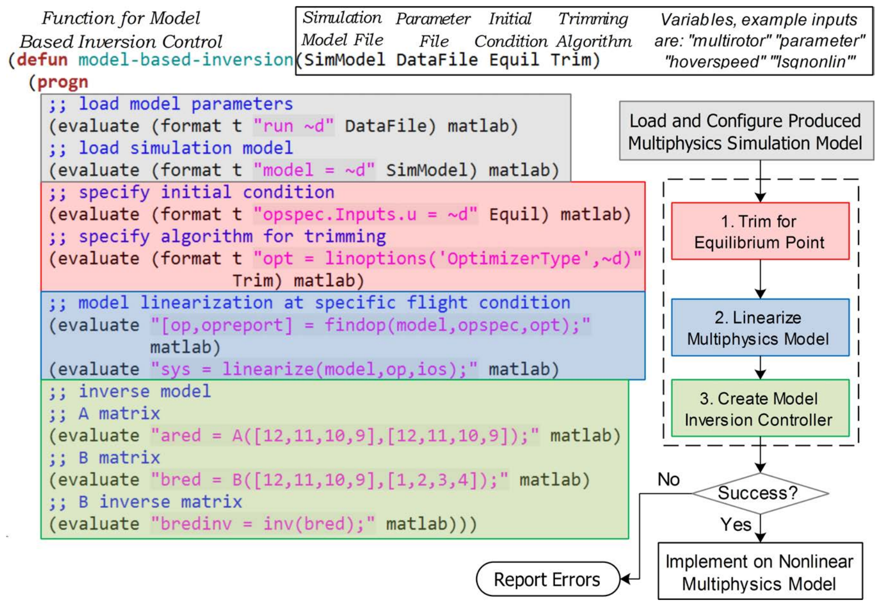
Figure 4.10: The overall architecture of maneuver tracking control system based on model based inversion control approach

linear model are the angular rates (roll rate, pitch rate and yaw rate) and the vertical velocity. These four states should be controlled to obtain a flight control system with a so called "rate command" response type [137]. For a given state vector and a desired time rate of change of the state vector, the required motor speed can be calculated by inversing the above equation as:

$$\begin{pmatrix} \Delta n_p \\ \Delta n_q \\ \Delta n_r \\ \Delta n_w \end{pmatrix} = B^{-1} \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{w} \end{pmatrix} - B^{-1} A \begin{pmatrix} p \\ q \\ r \\ w \end{pmatrix} \tag{4.10}$$

The desired angular rates and vertical speed are the top level inputs to the controller. These commands are transferred into a desired time rate of change of the state vector by means of a command filter and a compensator. A detailed description of this procedure can be found in Ref. [119]. Next, a model inversion controller can be built based on equation 4.10 and implemented on the original linear or nonlinear simulation model. Thus, it is not necessary to tune the parameters of the control system when using the model based inversion control method. Only the low order linear model is required. The overall structure of the control system is presented in Figure 4.10.

The key to developing a model based inversion control system is a set of sequential procedures. It starts with trimming the multiphysics simulation model in a predefined flight condition, next a linear model is obtained by means of numerical perturbation of the model, the linear model is reduced in order and this is followed by the creation of model inversion controller. Finally, the model inversion controller is implemented on the multiphysics simulation model. All these procedures are captured by the MIM and saved as a script file. The whole process is presented in Figure 4.11. After the model generation, the multiphysics model is trimmed to find the equilibrium point, where the multirotor UAV is hovering at altitude of 0 m. Then, the multiphysics model of multirotor UAV is linearized at the equilibrium point with the inputs of motor speed and the outputs of velocity elements (equation 4.9). Next, the model


Function for Model Based Inversion Control
Simulation Model File
Parameter File
Initial Condition
Trimming Algorithm
Variables, example inputs are: "multirotor" "parameter" "hoverspeed" "'lsqnonlin'"
Load and Configure Produced Multiphysics Simulation Model
1. Trim for Equilibrium Point
2. Linearize Multiphysics Model
3. Create Model Inversion Controller
Success?
No
Yes
Report Errors
Implement on Nonlinear Multiphysics Model

```
(defun model-based-inversion(SimModel DataFile Equil Trim)
 (progn
  ;; load model parameters
  (evaluate (format t "run ~d" DataFile) matlab)
  ;; load simulation model
  (evaluate (format t "model = ~d" SimModel) matlab)
  ;; specify initial condition
  (evaluate (format t "opspec.Inputs.u = ~d" Equil) matlab)
  ;; specify algorithm for trimming
  (evaluate (format t "opt = linoptions('OptimizerType',~d)"
                  Trim) matlab)
  ;; model linearization at specific flight condition
  (evaluate "[op,opreport] = findop(model,opspec,opt);"
          matlab)
  (evaluate "sys = linearize(model,op,ios);" matlab)
  ;; inverse model
  ;; A matrix
  (evaluate "ared = A([12,11,10,9],[12,11,10,9]);" matlab)
  ;; B matrix
  (evaluate "bred = B([12,11,10,9],[1,2,3,4]);" matlab)
  ;; B inverse matrix
  (evaluate "bredinv = inv(bred);" matlab)))
```

Figure 4.11: Function defined in the multiphysics information model (MIM) to capture the procedure information of model based inversion control approach

inversion controller is obtained by inversing the linear model (equation 4.10). Finally, if success, the model inversion controller is implemented on the full non-linear multiphysics model of the quadrotor UAV together with other control elements, which is illustrated in Figure 4.10. Otherwise, the intelligent modeling system will report an error message to the simulation experts. This can for example happen when the system tries to trim the design in a flight condition for which it does not have sufficient power.

4.6. DESIGN OF MULTIROTOR UAVS WITH CONFIGURATION AND DIFFERENT TOPOLOGIES

The intelligent modeling system has been proposed to model the physical entities of multirotor UAV in Chapter 3. As mentioned, there are almost unlimited design possibilities for the multirotor UAVs with N rotors and various configurations. Furthermore, the motion of the multirotor UAV is coupled, which makes the control system more complex than normal aircraft. In order to push to its limit, the intelligent modeling system is tested to generate the multiphysics simulation models and consistent control systems for the multirotor UAVs with configuration and different topologies.

4.6.1. SIMULATION CONFIGURATIONS

In this research study, every multirotor UAV is composed of nine components, such as the motor, propeller, gears, frame, base, cover, landing gear, battery and controller. These components are linked to thirty-two decision variables for modeling the physical entities and configuring the simulation models. Because the propeller of the multirotor UAV is a standard part which is provided by dedicated manufacturers, the decision variable for the propeller is a serial number, which is an integer. Eleven propellers are selected with sizes ranging from 5 up to 18 *inch*. The motor is treated in a similar way with eleven selections from small to big motor. Other components are defined by dimensional variables, such as length, width, height, etc.

Moreover, if the number of rotors is fixed, the configuration of multirotor UAV is mainly decided by dimensional variables and the selection of the components. However, the topology of the multirotor UAV can be changed by selecting a different number of rotors, which is a challenge for the dynamic modeling and simulation. Four multirotor UAVs are selected as a comparison (Table 4.1). They are classified into two groups according to the size: propeller ID 2 and motor ID 4 (configuration 1); propeller ID 10 and motor ID 10 (configuration 2). Then, the multirotor UAVs with four or six rotors are also grouped, which are the quadrotor UAVs and the sixrotor UAVs.

Configuration		1	2	1	2
Variable	Unit				
Motor ID	-	4	10	4	10
Propeller ID	-	2	10	2	10
Number of rotors	-	4	4	6	6
Payload up to	g	80	2400	140	3620
Propeller size	$inch$	6×3	10×5.5	6×3	10×5.5
Propeller weight	g	6	40	6	40
Motor speed	rpm	10600	6400	10600	6400
Motor weight	g	19	88	19	88
Frame size	mm	225	426	225	426
Frame weight	g	93	338	105	390
Gross weight	g	277	1053	320	1302

Table 4.1: Table of quadrotor UAVs and sixrotor UAVs in two configurations (there is a significant difference in payload)

4.6.2. Construction of Multiphysics Simulation Model for Multirotor UAVs

In order to challenge the proposed KBE system, the attribute "arm-number" (see Figure 3.14) is set from four to six. It is assumed that the simulation expert has completed the top-level configuration and written a script file including the necessary components and steps. The dynamics model of the physical plant is firstly instantiated from the MIM in MATLAB through the communication framework. Then, the inflow model and atmospheric module are indexed from the aerodynamic libraries and added to the simulation model. Next, the multiphysics simulation model is created for at a specific operating condition and subsequently linearized to get the model inversion controller. The multiphysics simulation model is integrated with the maneuver tracking control system and the model inversion controller and finally initialized with the test maneuvers (see Figure 4.12 and 4.13).

4.6.3. Flying Qualities Evaluation by Means of Virtual Flight Test Maneuvers

Various test maneuvers can be found in literature to evaluate the flying qualities of aircraft [137]. Three specific test maneuvers are selected for the multirotor UAVs. These are a lateral reposition maneuver, a hovering turn and a descending 360° circle. The first two maneuvers are described in ADS-33 [137].

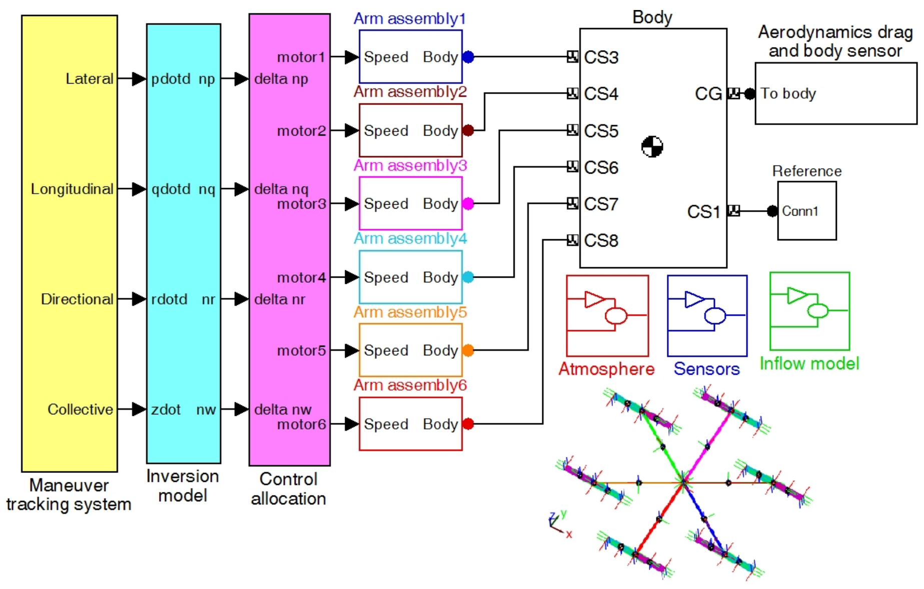


Figure 4.12: The multiphysics simulation model for a sixrotor UAV

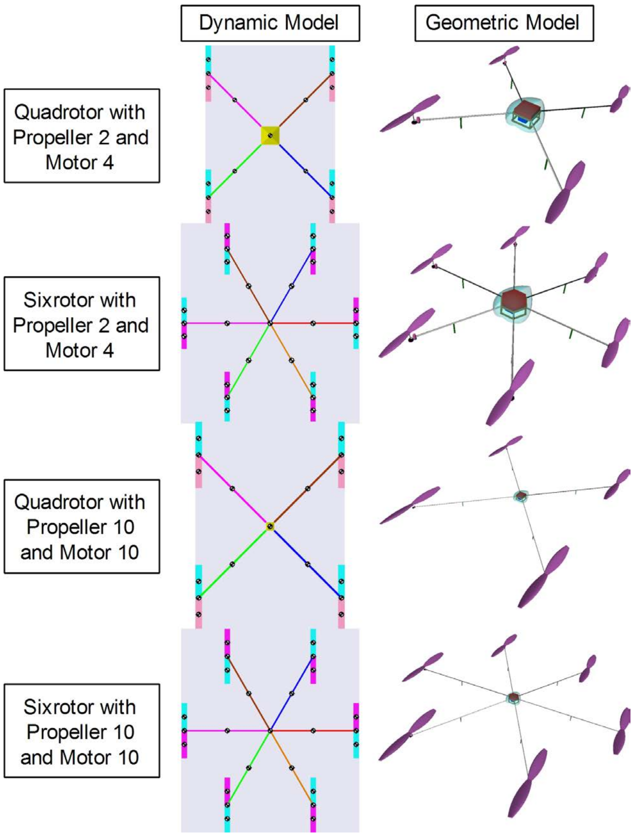

Dynamic Model
Geometric Model
Quadrotor with Propeller 2 and Motor 4
Sixrotor with Propeller 2 and Motor 4
Quadrotor with Propeller 10 and Motor 10
Sixrotor with Propeller 10 and Motor 10

Figure 4.13: The geometric model and dynamic model instantiated from the MIM for the quadrotor UAVs and the sixrotor UAVs with different configurations

Since ADS-33 is created for manned rotorcraft, dimensions of the maneuver and reference speeds and angles are scaled. The descending 360° circle is a maneuver that starts with a straight and level flight condition at a constant speed. Next, the model aircraft performs a gentle 360° descending circle with a specific altitude change and the maneuver is completed with straight and level flight in the original heading.

The results show that all configurations successfully complete the maneuvers with different performance levels (see Figure 4.14, 4.15, 4.16 and 4.17). The vertical motion of the different configurations is similar but there are significant differences in the lateral and longitudinal motion. All sixrotor UAVs have more oscillations around the target trajectory before straight flight is achieved. The quadrotor UAVs only overshoot once. This can be explained by the fact that a sixrotor UAV inherently has more cross coupling due to its

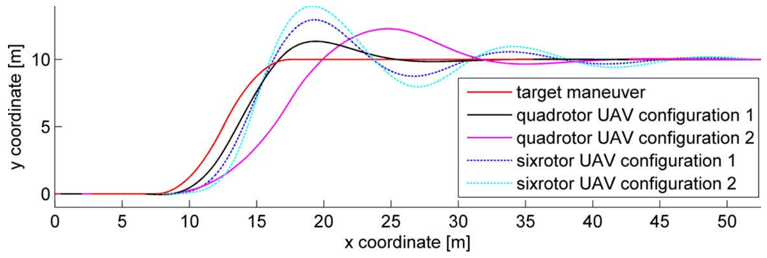


Figure 4.14: Flying qualities of quadrotor UAVs and sixrotor UAVs in lateral reposition maneuver (xy plane)

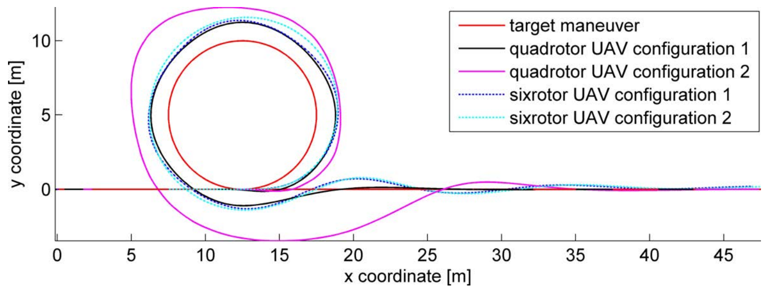


Figure 4.15: Flying qualities of quadrotor UAVs and sixrotor UAVs in descending 360° circle maneuver (xy plane)

geometric configuration. Moreover, as for the same amount of command inputs, the sixrotor rolls more than the pitch motion due to the control allocation strategy in Figure 4.7. The quadrotor UAV configuration 2 has the largest deviation from the target circle. Compared to quadrotor UAV configuration 1, it has a larger mass and inertia. This has a direct effect on pitch and roll control power. Moreover, the deceleration for the quadrotor UAV is a more challenging task because the speed of four motors has to be reallocated while a specific altitude must be kept. However, when the number of rotors is six, enough power is generated to counteract the effect of inertia. Therefore, the mass and inertia have less effects on the sixrotor UAVs than the quadrotor UAVs in this test case. Finally, it should be pointed out that all of the designs are controlled effectively and pushed to their performance limits whilst executing the maneuvers even though there are large differences among the four configurations. In other words, the flight control systems are successfully built by the MIM for all designs and this is an enabling factor for the use of MDO on such a vehicle.

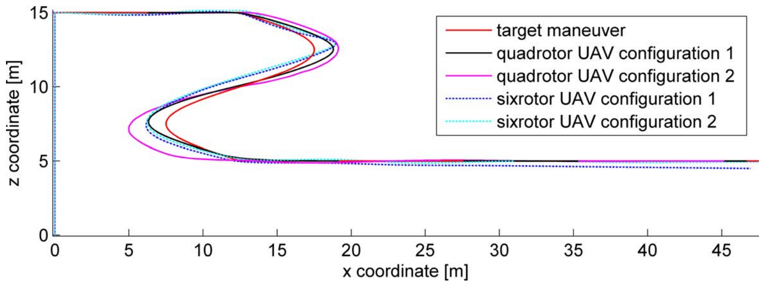


Figure 4.16: Flying qualities of quadrotor UAVs and sixrotor UAVs in descending 360° circle maneuver (xz plane)

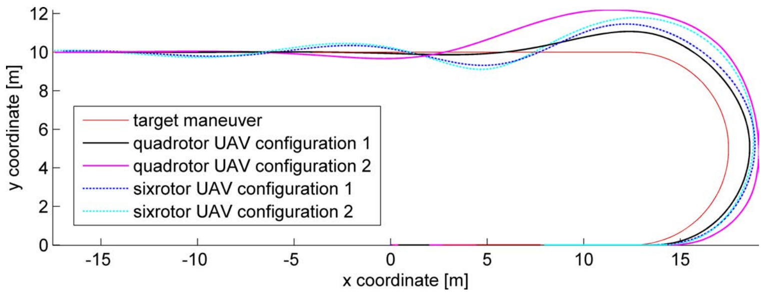


Figure 4.17: Flying qualities of quadrotor UAVs and sixrotor UAVs in hovering turn maneuver (xy plane)

4.7. SUMMARY

A complete multiphysics simulation model is composed of the multiphysics model of the physical plant, the corresponding control systems and simulation configurations. The multiphysics model has been produced by the intelligent modeling system in previous chapter. Hence, this chapter proposes methods and tools to automate the design of control system for the evaluation of test maneuvers in the MDO.

Due to extreme complexity of the E/E systems, there are three challenges for the automated design of control systems. It costs much time and effort to specify the strategies and parameters for the establishment of control systems, configure the simulation initial conditions and design a consistent control system. Although model based inversion is a promising method to accelerate the design process for the control systems, it needs methods and tools which supports the process automation not only limited to the E/E systems but also for the MDO framework.

Besides modeling the physical entities, the intelligent modeling system can also automate the design process within and out of the intelligent modeling system environment. In this research study, it automates the development of control systems by capturing the process information from the model based inversion control approach. Firstly, it collects related knowledge of the control systems, such as the strategies for control allocation, the values for parameters and constants, the procedures for the model based inversion control, and the information for analysis tools. Secondly, it models the collected knowledge into functions, data files and script files. It also generates the multiphysics model of the physical plant for the further simulation in the analysis tools. Thirdly, the intelligent modeling system connects with the analysis tools and performs the script file which contains the procedures of model based inversion control approach to trim and linearize the multiphysics model together with other simulation configurations and finally inversely get the model based inversion controller. The inversion controller can be implemented on the linear and nonlinear multiphysics model to construct the multiphysics simulation models which are used for the evaluation of maneuvers in the MDO.

The intelligent modeling system is tested to design the control systems for the multirotor UAVs with configuration and different topologies. After the simulation configurations, the intelligent modeling system generates both of the geometry models and the dynamic models integrated with aerodynamic components (the inflow models) and electric power systems (the motors) for the physical plants of the multirotor UAV configurations. Then, the model inversion controllers are obtained by trimming, linearizing and inversing the multiphysics models at every equilibrium point for each multirotor UAV configuration. Finally, the multiphysics models are integrated with corresponding model inversion controllers and simulation configurations and then simulated in three test maneuvers. The results show that all the multirotor UAVs are controllable though they are quite different in configuration and topology structure. It can be expected that the intelligent modeling system can ensure a consistent control system for the entire flight envelope of multirotor UAV.

In summary, the intelligent modeling system cannot only model the physical entities in multiple domains but also automatically design the control systems for the evaluation of maneuvers in the MDO. The performance of the E/E systems can quickly provide feedback to the MDO framework, which in turns establishes a concurrent design environment together with other disciplines.

5 Generation of Consistent Control Software Components for Serving MDO

5.1. Introduction

The performance of the E/E systems can be evaluated when the multiphysics model of the physical plant is integrated with a flight control system (Chapter 3 and Chapter 4). Usually, code generation is subsequently performed to test the control systems in a more realistic environment.

As mentioned in Chapter 1, there are millions of lines of software code for the control systems embedded on road vehicles or aircraft. Moreover, the consistent design of the E/E systems, the development of the associated software code and the integrated design of these systems with the mechanical system is a very time consuming task [97]. On one hand, when the control system is represented by source code, it is not straightforward to map the attributes of physical entities with the variables or parameters in the control software. The software developer may make mistakes, such as multiple usage of the same variable, due to ambiguous or similar definitions. If the comments are not written in detail, it is difficult to recall the functionality for a specific piece of the source code in a later stage according to the "curve of forgetting" [138]. On the other hand, the programming itself is also in high risk of errors because a typical code is very lengthy. Programmer may wrongly set or reset the variables. It is also difficult to avoid typing errors. As mentioned in Chapter 1, software errors constitute more than 70% of errors during the software development associated to control technology.

There are many patents and research studies related to automated generation of software code [139–142]. Moreover, many commercial analysis tools, such as MATLAB, are integrated with the functionality of code generation from block diagrams. In this research study, methods and tools are proposed to automatically generate the software code using the KBE approach. In order to avoid software errors and produce the control software efficiently, the focus is on two challenges:

- Generation of consistent control software from the top level physical configuration of the product and the control architecture to the software components;
- Automated generation of software code for the control system which is easy to understand.

This chapter is organized as followed. Various methods used for automated generation of software components (or automatic code generation) are reviewed. Then, extensions to the intelligent modeling system are proposed to provide consistent control software in support of the MDO problem. Next, the proposed method and tool are demonstrated for two test cases; (1) to automatically produce the flight control software components for the multirotor UAV and (2) the generation of control software for the anti-lock braking system (ABS) of a novel road vehicle configuration.

5.2. A BRIEF REVIEW OF AUTOMATIC CODE GENERATION

Many researchers propose methods and tools for the automated generation of software code. Budinsky et al. [139] propose a tool that automates the implementation of design patterns and then generates all the pattern-prescribed code. Mery and Singh [140] develop a translation tool which generates target programming language code from Event-B formal specification. Pastor et al. [141] use an OO-method approach which provides a precise, conventional graphical notation to obtain a system description at the problem space level. An abstract model which is extracted from the system description determines the software representations from the conceptual modeling constructs. Mozumdar et al. [142] propose a framework that can model sensor network components and automatically generate the code for different platforms based on Simulink, Stateflow and the Embedded Coder. Javed et al. [143] develop a method that transforms the sequential diagrams into platform specific test cases with the model-to-model and model-to-text two steps.

There are also many patents about automatic code generation. Savage et al.

[144] generate the software code from the source metadata obtained by analysis of the data sources and the structure entities of a generic data repository for execution in an EDM application. Henninger et al. [145] use an object model to automatically map information between an object-oriented application and a structured database by capturing all of the semantics (implications), such as inheritance and relationships among objects. Perycz et al. [146] propose a system and method for software modularization and automatic code generation for embedded systems. Sadiq [147] provides a system and method that provide a means for application infrastructure services to insert specific programming code (XML documents) in the generated code.

It should be pointed out that the software components are built based on the control systems which are used to regulate the behavior of physical entities in real world. Although some commercial software tools, like MATLAB, support automatic code generation from block diagrams, all above methods and tools start from the aspect of software development instead of the design process for engineering applications. In other words, the relationship from the top level configuration of the physical design to the control system and finally to the associated software components has not been taken into account in theses research studies. The structure of the software is not based on the physical configuration of the product and the control architecture. Although there are methods and tools that can support mechatronic design, including the physical modeling and code generation in one development environment, they cannot provide high fidelity model to support the MDO problem as explained in Chapter 1. Thus, there is a gap between the software development and the analyses of other disciplines.

In the previous chapters it was shown how the intelligent modeling system constructs the multiphysics simulation model which integrates the design knowledge from multiple engineering domains, such as dynamics, aerodynamics, control systems, etc. In this chapter, the intelligent modeling system is further developed to set up connections from the physical modeling of the complex systems, the control architecture to the generation of control software components. On one hand, the variable definitions and the values for the parameters in the control software are based on the physical configuration and the control architecture. On the other hand, the intelligent modeling system can also automatically update the software structure, such as the example of the different control allocation strategies required for either quadrotor UAVs or sixrotor UAVs in Chapter 4, through defining the functions for generic code generation.

5.3. AUTOMATIC CODE GENERATION BY INTELLIGENT MODELING SYSTEM

Typically, the control software components consist of three models: the data model, behavior model and real-time model. The data model contains the values for the variables and parameters in the software components. The behavior model represents the calculations, strategies and control logic. Lastly, the real-time model is the test harness for the combination of the data model and behavior model [148].

As can be seen in Figure 5.1, the intelligent modeling system can generate all the submodels for the control software components. First, it extracts the variable definitions and values from the MIM in terms of physical modeling and the control architecture and then it instantiates them into a specific data format, such as a CSV file. Second, the MIM defines functions to capture the software components, such as control logic, for the behavior model, which are usually saved in source code (similar to the generation of XML file in Chapter 3). Third, the intelligent modeling system can also provide the test harness, like the multiphysics simulation models in Chapter 4, to evaluate the software functions.

5.3.1. GENERATION OF DATA MODEL

It should be pointed out that the data model (values) of control software components can come either from the physical entities or from the architecture of the control system. On one hand, the MIM defines classes to represent the physical entities (Chapter 3). Every class is associated with several attributes to capture the different characteristics of the physical entities. Similar to writing the XML file required for modeling the dynamics of the vehicle in Chapter 3, the intelligent modeling system defines functions to extract the values of attributes

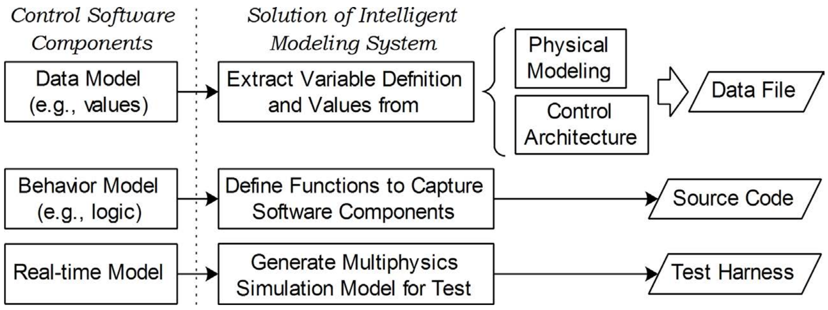

Control Software Components
Solution of Intelligent Modeling System
Data Model (e.g., values)
Extract Variable Defintion and Values from
Physical Modeling
Control Architecture
Data File
Behavior Model (e.g., logic)
Define Functions to Capture Software Components
Source Code
Real-time Model
Generate Multiphysics Simulation Model for Test
Test Harness

Figure 5.1: Generation of data model, behavior model and real-time model for control software components by intelligent modeling system

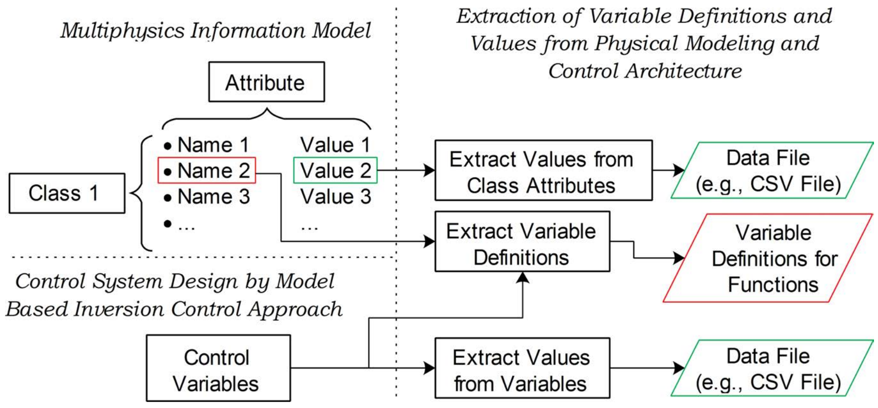

Multiphysics Information Model
Extraction of Variable Definitions and Values from Physical Modeling and Control Architecture
Attribute
Class 1
• Name 1
• Name 2
• Name 3
• ...
Value 1
Value 2
Value 3
...
Extract Values from Class Attributes
Data File (e.g., CSV File)
Extract Variable Definitions
Variable Definitions for Functions
Control System Design by Model Based Inversion Control Approach
Control Variables
Extract Values from Variables
Data File (e.g., CSV File)

Figure 5.2: Extraction of class definitions from the multiphysics information model (MIM) to generate the data model by intelligent modeling system

from the classes level by level. Moreover, the intelligent modeling system collects the attribute names to define the variables in the software components, as shown in Figure 5.2. On the other hand, the intelligent modeling system extracts the variable definitions and values from the control variables in the control architecture. It is assumed that the values are already obtained during the control system design, such as for the model based inversion control approach presented in Chapter 4. Finally, the intelligent modeling system saves the variable definitions for the construction of the behavior model and writes the corresponding values into a data file which can be used for testing the control software components. Therefore, the MIM links the physical modeling and the control system design with the development of software components.

5.3.2. GENERATION OF BEHAVIOR MODEL

In Chapter 3, the ability to produce an XML file for the analysis of the vehicle dynamics by the intelligent modeling system was demonstrated. Again, the intelligent modeling system writes the software components in strings, which can be defined in any programming languages. It is suggested that every piece of the source code should be a single, unambiguous, authoritative representation within the system, following the don't repeat yourself (DRY) principle [149] . Thus, as can be seen in Figure 5.3, the intelligent modeling system divides one software component into basic elements which can be manipulated and reorganized according to the requirements. The basic elements which have input and output variables are written in pieces of source code. Every basic element deals with one step for the whole software component, such as a mathematical computation or a logical evaluation. The

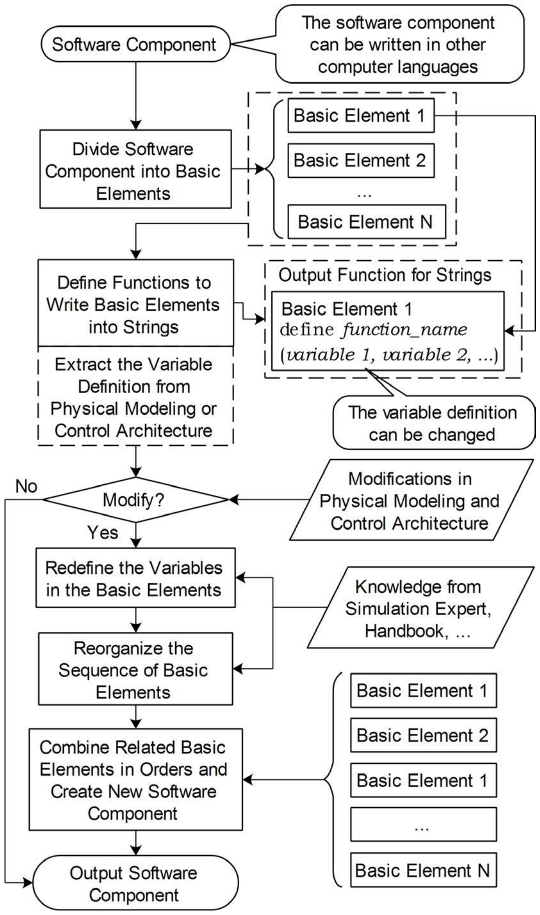

Software Component
The software component can be written in other computer languages
Divide Software Component into Basic Elements
Basic Element 1
Basic Element 2
...
Basic Element N
Define Functions to Write Basic Elements into Strings
Output Function for Strings
Basic Element 1
define *function_name*
(*variable 1, variable 2, ...*)
Extract the Variable Definition from Physical Modeling or Control Architecture
The variable definition can be changed
No
Modify?
Yes
Modifications in Physical Modeling and Control Architecture
Redefine the Variables in the Basic Elements
Knowledge from Simulation Expert, Handbook, ...
Reorganize the Sequence of Basic Elements
Basic Element 1
Basic Element 2
Basic Element 1
...
Basic Element N
Combine Related Basic Elements in Orders and Create New Software Component
Output Software Component

Figure 5.3: The generation of new software component (behavior model) by intelligent modeling system

intelligent modeling system uses output functions to write the basic elements of software components into strings. Thus, it is possible to target almost any programming language. However, there are still two challenges for the code generation.

On one hand, because the variable definitions are directly linked with the physical model and the control architecture, they must be automatically updated in the software components as well. This challenge can be solved by most OO languages, such as Common Lisp, which has internal functions to freely replace and modify the contents in the strings. As illustrated by Figure 5.3, the intelligent modeling system can directly extract the variable definitions from the classes of physical entities (or the control architecture) to define the variables in the basic element. Because the basic element is encapsulated as a black box, the intelligent modeling system can give different variable definitions for the same element to create a sequence of similar functions. In other words, when the variable definitions of the physical model and control architecture are modified, the intelligent modeling system can automatically update the corresponding variables in the software components.

On the other hand, the structure of the software component should also adapt to the modifications in the top level physical configurations and the control architecture. Because the software component is divided into basic elements, every time when the topology of physical entities (or control architecture) is extended, the intelligent modeling system will redefine the variables in the basic elements, reorganize the order of the basic elements and finally combine them together to create a new software component. In other words, the same basic element can be repeatedly instantiated by different variables. As for different basic elements, the intelligent modeling system can collect the knowledge from simulation experts, control system design experts or handbooks to reorganize them in a specific order for creating the new software components.

The real-time model is in fact the multiphysics simulation model. This was model was described in the previous chapter and will not be discussed again. It will be shown how the intelligent modeling system automatically generates the software components for two test cases:

- Generation of software components for the multirotor UAVs with different control architectures;
- Generation of a logic controller for the ABS on road vehicles.

The objectives of these two test cases are different. Because the data model (values) of the control system can be automatically obtained by using the model based inversion control approach, the first test case more focuses on how the intelligent modeling system changes the software structure for different control architectures. The second test case shows that how the intelligent modeling system automatically adjusts the data model of the control software to adapt various the physical configurations.

5.4. Generation of Software Components for Multirotor UAVs

5.4.1. Software Component for Rotor Speed Allocation

As mentioned in Chapter 3, many different multirotor UAV configurations are possible due to the availability of many different subcomponents, technologies and control architectures. Thus, the generation of control software components must be able to adapt to various design possibilities.

For instance, the motion of a quadrotor UAV is achieved by applying a rotational speed difference among the different rotors (also the motor speed). As indicated in equation 4.4, the rotor speed Δn is calculated based on the sum of the speed increment due to the roll Δn_p, pitch Δn_q, yaw Δn_r and heave Δn_w commands.

For example, two control allocation strategies for the pitch motion of a quadrotor UAV can be defined. This is shown in Figure 5.4. If the quadrotor UAV is in hovers and a pitch command is introduced (rolling motion about the z-axis) in order to start moving in the X direction, the speed of motor 1 and 2 should be smaller than motor 3 and 4 for configuration 1. For configuration 2, the speed of rotor 1 decreases somewhat while the speed of rotor 3 increases the same amount and rotor 2 and 4 maintain their original speed.

The rotor speed (also the motor speed) resulting from roll, pitch, yaw and altitude commands can be calculated by equation 5.1 and 5.2 for configurations 1 and 2, respectively. In this example, the coefficient matrix is multiplied with the vector of commands (k_{np}, k_{nq}, k_{nr} and k_{nw}) to obtain the speed for every individual rotor. The values in the coefficient matrix determine whether the rotor speed is increasing (+1), decreasing (-1), or maintained (0) constant as a result of

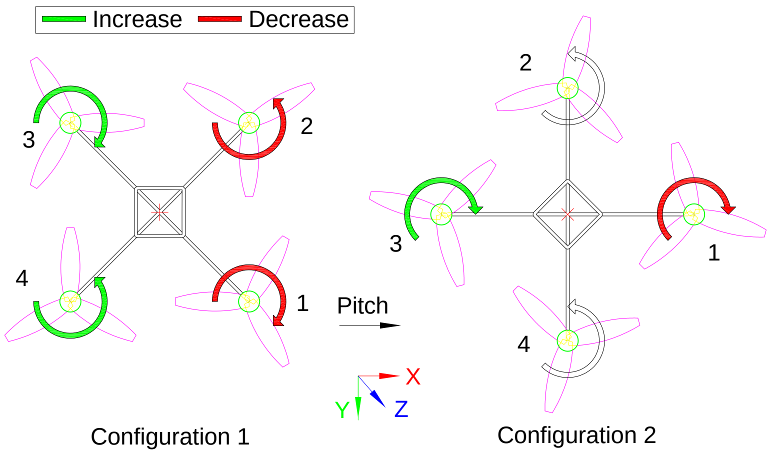


Figure 5.4: Different control allocation strategies for pitch motion

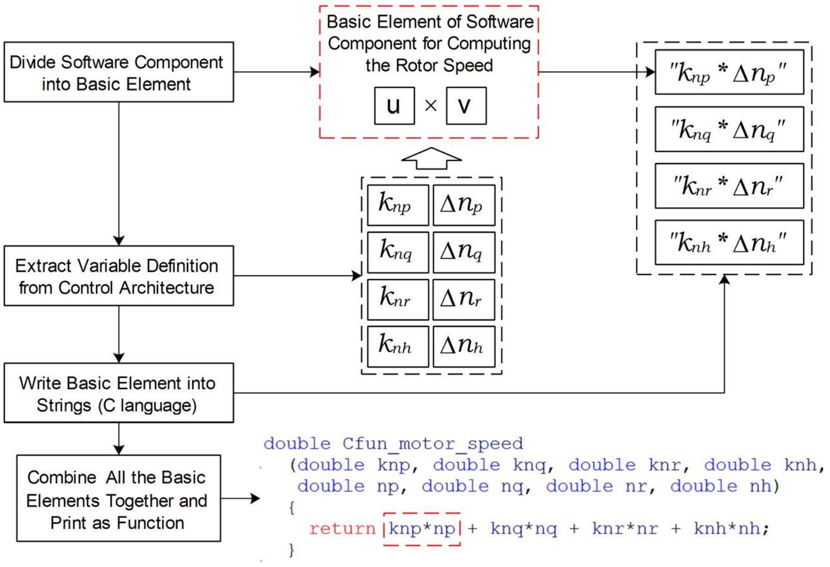
a positive input command.

$$\begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix} = \begin{pmatrix} -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \Delta n_p \\ \Delta n_q \\ \Delta n_r \\ \Delta n_w \end{pmatrix} \tag{5.1}$$

$$\begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & 1 \\ 1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 1 \\ -1 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \Delta n_p \\ \Delta n_q \\ \Delta n_r \\ \Delta n_w \end{pmatrix} \tag{5.2}$$

The intelligent modeling system saves the values in the coefficient matrices as data models in this case. When the user of the design system decides which control allocation strategy should be used for the quadrotor UAV, the intelligent modeling system searches for the proper coefficient matrix and writes the data model as CSV file which can be used for testing the software components.

Moreover, the behavior model is a summing function which adds all the speed changes due to the different commands. As shown in Figure 5.5, the basic element is the product of the speed increment due to a specific command and the corresponding coefficient. The definition of variables is extracted from the control architecture in Figure 4.6 (Chapter 4). Then, the basic element is printed


Divide Software Component into Basic Element
Basic Element of Software Component for Computing the Rotor Speed
u × v
"$k_{np} * \Delta n_p$"
"$k_{nq} * \Delta n_q$"
"$k_{nr} * \Delta n_r$"
"$k_{nh} * \Delta n_h$"
k_{np} Δn_p
k_{nq} Δn_q
k_{nr} Δn_r
k_{nh} Δn_h
Extract Variable Definition from Control Architecture
Write Basic Element into Strings (C language)
Combine All the Basic Elements Together and Print as Function

```
double Cfun_motor_speed
   (double knp, double knq, double knr, double knh,
    double np, double nq, double nr, double nh)
   {
     return knp*np + knq*nq + knr*nr + knh*nh;
   }
```

Figure 5.5: Basic elements and final software component (C code) for computing the rotor speed

by the intelligent modeling system four times with three "+" inserted among them. Finally, the intelligent modeling system creates a generic function as output (written in C code), to compute the speed for all rotors.

In this example, there are four pairs of inputs and outputs specified in Figure 4.6. It is possible that the multirotor UAV with N rotors may have more degrees of freedom than the quadrotor UAV. In this case, the intelligent modeling system can automatically extend the structure of the software component by instantiating the basic element also N times. This is to say, the intelligent modeling system can automatically generate proper software components even when the topology of the physical system (or control architecture) changes significantly.

Finally, the intelligent modeling system writes the generic function in the C language, which can be compiled in MATLAB to build a so-called S-function. In order to check whether the S-function works the same as the original sum block in MATLAB, a comparison is given in next section.

5.4.2. TEST OF SOFTWARE COMPONENTS

A descending 360° circle is used to test the software components and to compare it to the original control system. An arbitrary quadrotor UAV (propeller ID 2 and motor ID 4) is selected as test case. The C code produced by the intelligent modeling system is compiled in MATLAB to build an S-function

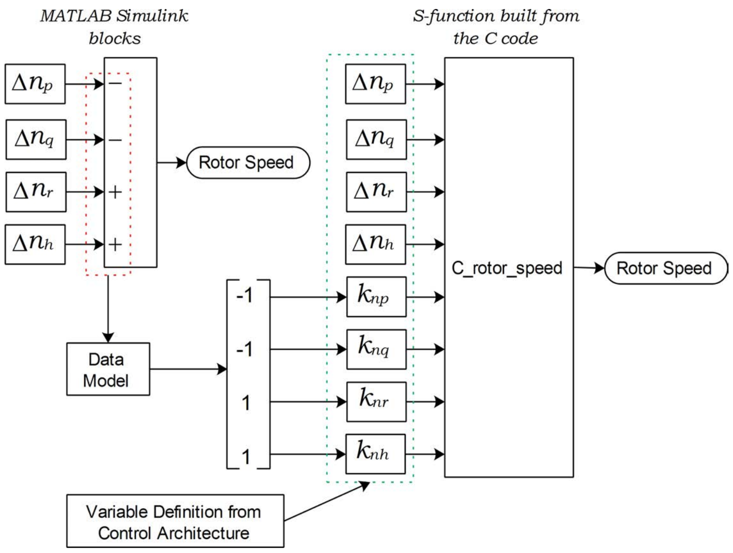


Figure 5.6: Comparison of control allocation strategies modeled by MATLAB Simulink blocks or the S-function built from the C code

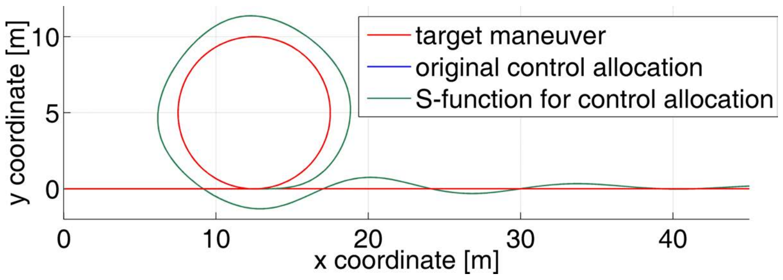


Figure 5.7: Flying qualities of quadrotor UAVs equipped with different control allocation modules (MATLAB Simulink blocks or the S-function built from the C code) in descending 360° circle maneuver (xy view)

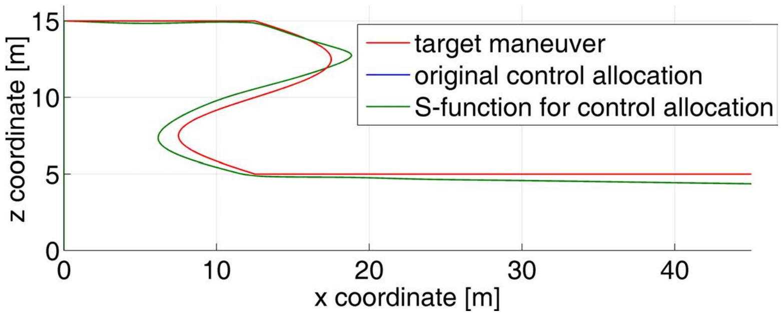


Figure 5.8: Flying qualities of quadrotor UAVs equipped with different control allocation modules (MATLAB Simulink blocks or the S-function built from the C code) in descending 360° circle maneuver (xz view)

which is used to compare with the original Simulink blocks used for control allocation. As can be seen in Figure 5.6, the S-function accepts the inputs from the coefficient matrix and the speed increments due to the commands. The variable definitions in the final S-function are obtained from the control architecture.

The simulation results are illustrated in Figure 5.7 and 5.8. The results show that the S-function performs exactly the same as the original Simulink block in MATLAB. Thus, the C code produced by the intelligent modeling system can be embedded on actual hardware, such as a real quadrotor UAV.

5.5. GENERATION OF CONTROL SOFTWARE FOR ABS ON A PASSENGER CAR

A novel electric vehicle configuration named A-line, where passengers are seated in line in order to have a vehicle with low aerodynamic drag [33], is used as test case. Because of its narrow track, it is necessary to check the required braking distance to ensure safety. The braking distance is limited to 70 m, when the vehicle starts to brake at 100 km/h [96]. The ABS is modeled specifically for A-line configurations. However, the braking distance is affected by many factors related to physical design parameters, such as the wheelbase, front axle load ratio, hydraulic pressure, etc. The software components of the ABS must keep pace with the overall design process in relation to disciplinary analyses. The intelligent modeling system is used to automatically generate the software components (C code) of the ABS for the A-line series. The objective is twofold. The first objective is to verify that the C code produced by the intelligent modeling system behaves the same as the original control system built by the MATLAB Simulink blocks. The second objective is to verify that the intelligent modeling system automatically adjusts the control software components of the ABS to (discrete) changes in the physical design of the A-line configurations.

5.5.1. Control System of the ABS

The block diagram of the ABS control system is presented in Figure 5.9. The physical plant includes the master cylinder, electromagnetic valve, brake actuator, wheel and sensors. The heart of the ABS is a logical controller. This controller accepts wheel speed derivative (wheel angular acceleration) and actual relative slip which are compared with the inherent parameters of wheel speed derivative thresholds $a1$, $a2$ and the desired relative slip to compute the control logic for the master cylinder to increase, hold or decrease hydraulic pressure. The basic theory of this logical controller can be found in Ref. [150]. The hydraulic pressure is increasing when the actual wheel speed derivative is lower than $a1$, while decreasing when it is higher than $a2$. In any case, if the actual relative slip is lower than desired relative slip, the hydraulic pressure will be decreased to avoid locking the wheel early.

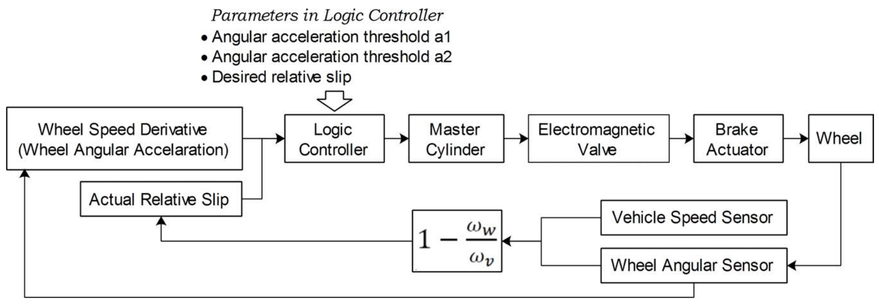

Parameters in Logic Controller
• Angular acceleration threshold a1
• Angular acceleration threshold a2
• Desired relative slip
Wheel Speed Derivative (Wheel Angular Accelaration)
Logic Controller
Master Cylinder
Electromagnetic Valve
Brake Actuator
Wheel
Actual Relative Slip
$1 - \frac{\omega_w}{\omega_v}$
Vehicle Speed Sensor
Wheel Angular Sensor

Figure 5.9: The block diagram of the ABS control system (ω_w - wheel angular speed, ω_v - vehicle speed divided by wheel radius))

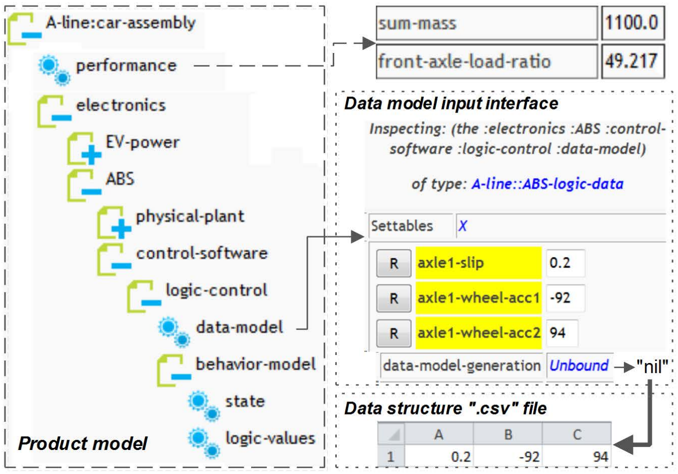
5.5.2. Data Model Generation

It is assumed that the user specifies the values in the logic controller for one A-line model. This is shown in Figure 5.10. It is also assumed that this A-line model has a wheelbase of 2400 mm, track of 1000 mm, gross mass of 1100 kg , front axle load ratio of 49.21%, etc.

Because the top level configuration is subject to significant changes in the conceptual design phase, the challenge is how to automatically set the values in the logic controller of the ABS for the whole design envelope. The wheel speed derivative thresholds $a1$ and $a2$ are determined by five factors: axle load, wheel moment of inertia, hydraulic pressure, hydraulic delay and tire-road friction coefficient. An increase in these factors can have a positive effect (axle load, hydraulic pressure and tire-road friction co-efficient) while other factors have a negative effect (wheel moment of inertia and hydraulic delay). For example, if the front axle is loaded more, the $a1$ for the first axle will be given a larger value, which has been defined in equation 5.3. In other words, when the load on the first axle increases 1%, the wheel speed derivative threshold $a1$ for the first axle will also increase one percent.

$$a1 = a1' \times (1 + (\frac{G \times R}{G' \times R'} - 1) \times 100\%) \tag{5.3}$$

where $a1$ is the new value of the wheel speed derivative threshold, $a1'$ is the old value of the wheel speed derivative threshold, G is the new value of gross weight, G' is the old value of gross weight, R is the new value of front axle load ratio, R' is the old value of front axle load ratio.



sum-mass	1100.0
front-axle-load-ratio	49.217

Inspecting: (the :electronics :ABS :control-software :logic-control :data-model)

of type: A-line::ABS-logic-data

Settables	X
R axle1-slip	0.2
R axle1-wheel-acc1	-92
R axle1-wheel-acc2	94
data-model-generation	Unbound

	A	B	C
1	0.2	-92	94

Figure 5.10: Generation of data model by intelligent modeling system for ABS software components

The intelligent modeling system can capture the knowledge, such as equation 5.3, to automatically set the values for the wheel speed derivative threshold $a1$ and $a2$ when the top level configuration changes. As is shown in Figure 5.10, the wheel speed derivative thresholds $a1$ and $a2$ are -92 and 94 rad/s^2, respectively. As mentioned, the selection of the initial values is based on the consideration of vehicle gross weight (1100 kg) and front axle load ratio (49.2%). The intelligent modeling system will increase the wheel speed derivative threshold $a1$ for the first axle due to more load transferred, if the front axle load ratio changes to 51%. Finally, when the simulation expert clicks "data-model-generation" in Figure 5.10, the status "Unbound" will change to "nil" which means the intelligent modeling system writes the values into a data file. Further test case will show the results of ABS maneuvers for an A-line model with different axle load ratios.

5.5.3. BEHAVIOR MODEL GENERATION

The behavior model of the logic controller is responsible for switching phases and outputting control logic. As can be seen in Figure 5.11, it compares actual wheel speed derivative with the thresholds in the data model. When the threshold is reached, the phase will switch to the next one. Otherwise, it will keep current phase until the condition is triggered. Every phase corresponds to one control logic which is used to operate the hydraulic brakes. The logic values are saved in a separate file similar to the data model. In this example, the

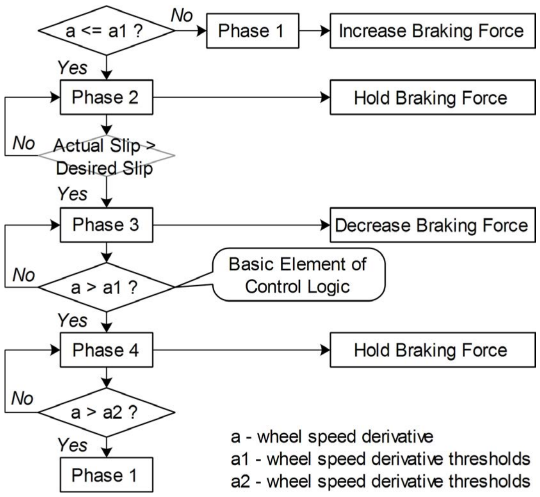


Figure 5.11: Generation of behavior model by intelligent modeling system for anti-lock braking system (ABS) software components

intelligent modeling system divides the control software of the ABS into logic expressions which are the basic elements. More complicated control logic can be built by inserting more logic expressions, if necessary. Finally, the intelligent modeling system produces a complete file in C code, which includes the four phases in Figure 5.11.

5.5.4. Configuration of A-line Models

In order to test the data model and behavior model of the ABS software components, the intelligent modeling system generates a multiphysics simulation model of the road vehicle in MATLAB Simscape. The multiphysics simulation model includes a physical plant, consisting of the vehicle dynamics modeled using multi rigid-body dynamics, mathematic expressions for the pump, electromagnetic valve, brake actuator, etc., and the control system of the ABS. The control system is originally built by MATLAB Simulink blocks and later replaced by the generated C code.

It is assumed that the user specifies the top level configurations for two A-line models, which are listed in Table 5.1. Most parameters are the same except for the front axle load ratio, which are 49.2% and 51% for the concept 1 and 2, respectively. It is expected that the A-line models start to brake with an initial speed of 100 km/h on a straight path.

5.5.5. Comparison of Behavior Models

In this section, the C code and the data model produced by the intelligent modeling system constitute the logic controller. This controller is compared to the original controller built by MATLAB Simulink blocks. The logic controllers are simulated using the test harness (multiphysics simulation model). In order to exclude the effect from the rear wheels on the braking performance, only the front wheels are equipped with the ABS logic controller.

Parameters	Unit	Concept 1	Concept 2
Gross weight	[kg]	1000	1000
Number of axles	-	2	2
Front axle load ratio	-	49.2%	51%
Wheelbase	[mm]	2400	2400
Track	[mm]	1000	1000
Tire	-	205/55 R16	205/55 R16
Initial velocity	[km/h]	100	100
Hydraulic pressure	[MPa]	1	1
Hydraulic delay	[s]	0.01	0.01

Table 5.1: Parameters for the A-line models with different front axle load ratios

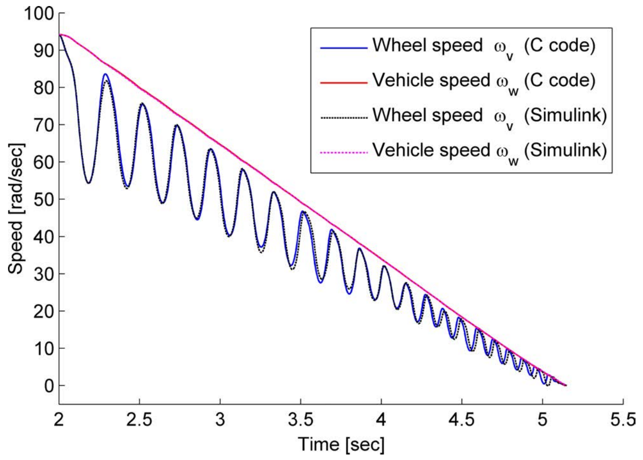


Figure 5.12: Comparison of anti-lock braking system (ABS) logic controller modeled by the C code generated by intelligent modeling system or MATLAB Simulink blocks

As is shown in Figure 5.12, the results show that both wheel speed ω_v and vehicle speed ω_w are almost identical for the logic controllers built by the C code and the Simulink blocks. There are very few differences at the end of the maneuver due to the accumulation of small errors. The reason for this is that the logic controller built from the C code is simulated in the discrete time domain while the Simulink model is tested in continuous time domain.

5.5.6. Simulation and Comparison

The intelligent modeling system combines the logic controller built from the C code with the multiphysics model to construct the complete multiphysics simulation model for the A-line series. The multiphysics simulation model is tested for A-line models with a front axle load ratio 49.2% and 51% in Table 5.1. The vehicles start to brake at 2 s with initial velocity of 100 km/h. When the vehicle speed is lower than 5 km/h, the ABS is switched off.

In order to show the results clearly, only the front wheels are compared with the vehicle speed in Figure 5.13. The results illustrate that although two configurations have different front axle load ratios, both of them shows reasonable results in the test maneuver, where the front wheels gradually stop after nine braking cycles. Moreover, the relative slip in Figure 5.14 shows than both of the front wheels and the rear wheels are neither locked early nor released too much during the simulations. The reason is that the intelligent modeling system captures the knowledge from equation 5.3 and automatically computes the values for the wheel speed derivative thresholds $a1$ and $a2$ when the front axle load ratio changes from 49.2% to 51%. Consequently, when the

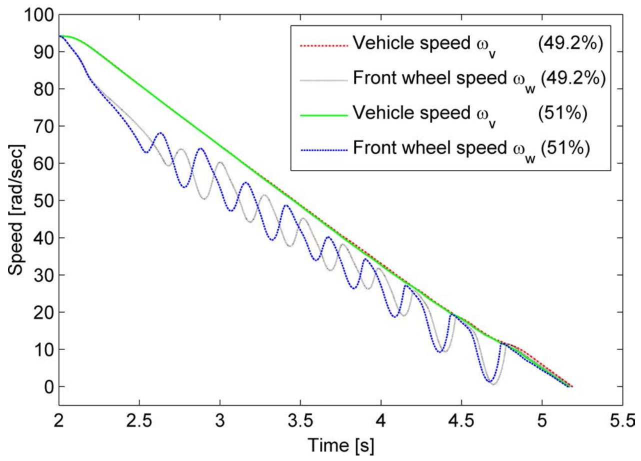


Figure 5.13: Vehicle speed vs. front wheel speed (A-line models with front axle load ratio 49.2% or 51%)

front axle is given more load, the wheel speed derivative threshold $a1$ for the front wheels increases from -92 to -95 while decreases from -80 to -77 for the rear wheels.

The adjustment of the wheel speed derivative thresholds $a1$ and $a2$ for the two concepts are summarized in Table 5.2. Therefore, no matter how many concepts are evaluated in the conceptual design phase, the intelligent modeling system can always provide a proper data model for the control software

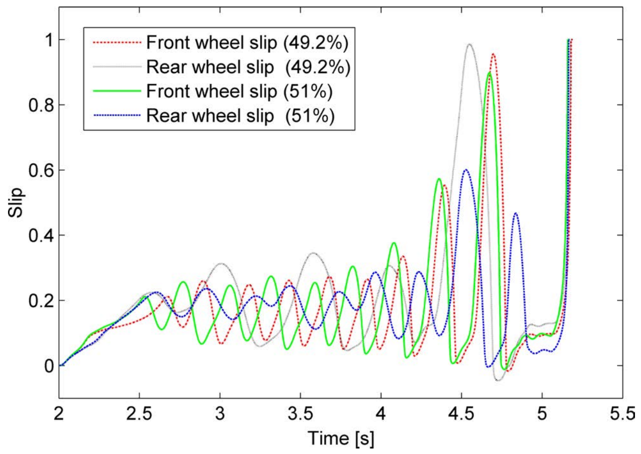


Figure 5.14: The relative slip for the A-line models with front axle load ratio 49.2% or 51% braking from 100 km/h

Front axle load ratio	Front wheels		Rear wheels	
	$a1$	$a2$	$a1$	$a2$
49.2% (Concept 1)	-92	94	-80	0
51% (Concept 2)	-95	94	-77	0

Table 5.2: The adjustment of wheel speed derivative thresholds $a1$ and $a2$ for different front axle load ratio 49.2% or 51%

component of the ABS by capturing the knowledge from simulation experts or handbooks.

5.6. Summary

Control software components are embedded on the real physical system. It must be tested whether these components perform as desired in all possible operating conditions. Control software can be very lengthy and complex. Many researchers therefore work on the topic of automatic code generation. Various commercial software tools can also generate source code from block diagrams. However, the automated generation of software components is not optimal. There is a gap between the software components (structure and variables) with the other engineering disciplines related to the physical design and the overall control architecture. Furthermore, the use of automatic code generation in the early design stages for products with highly integrated E/E systems, such as unstable UAVs is not optimal.

It is proposed to automatically generate consistent control software components to support the MDO process using the KBE approach. This technique is implemented in the intelligent modeling system. Typically, software components are composed of three submodels: the data model, the behavior model and the real-time model. As for the data model, the intelligent modeling system extracts variable definitions and values from the physical system (or the control architecture) to specify the variables in the control software components. The variable definitions and values may come from the classes used for modeling the physical entities in Chapter 4 or from the variables and parameters in the control system (Chapter 5). Moreover, the intelligent modeling system divides the software components into basic elements which can be reorganized according to the requirements. It writes the basic elements into strings, (targeted to any computer language). The contents in the basic elements, such as variable definitions, can also be replaced. Finally, the data model and behavior model are tested on the real-time model which in this case is the multiphysics simulation model.

This capability of the intelligent modeling system is demonstrated on two test cases: The generation of software components for the flight control system of the multirotor UAVs and for the ABS of a novel road vehicle configuration. When the control architecture of the multirotor UAV changes, the intelligent modeling system can automatically select the appropriate basic software elements and combine them together to produce a new software component that solves the control allocation problem. For testing, the intelligent modeling system generates a piece of C code for the multirotor UAV which is compiled in MATLAB to build an S-function. The S-function is used to compare the original control system built by MATLAB Simulink blocks with the C code for a complex test maneuver. Results show that the S-function performs exactly the same as the original control system, which validates the C code produced by the intelligent modeling system.

In the second test case, the intelligent modeling system generates the source code (C code) for the ABS logic controller used on road vehicles. It directly links the values in the software component (logic controller) with the physical configuration of the road vehicle. Again, the C code is compared with the MATLAB Simulink blocks for a straight braking maneuver with an initial speed of 100 km/h. There are only very small differences between the C code and MATLAB Simulink blocks due to different sampling methods used.

Finally, the automatic development of the logic controller compiled as C code is tested in a design case: two road vehicle configurations with different front axle load ratios. The intelligent modeling system successfully captures the associated design knowledge and automatically generates properly working C code which is easy to understand. In summary, the concept of the MIM connects the physical modeling, control system design with the generation of software components, which can provide a consistent control software component to support the development of E/E systems. Because the whole process can be automatically completed, it is possible to integrate the proposed methods and tools in an MDO framework. This approach can be very valuable if physical design changes are made to an existing vehicle which has lengthy and complex control software (millions of lines of code).

6

Design and Optimization of a Multirotor Helicopter

6.1. Introduction

In previous chapters, it has shown how the intelligent modeling system models the physical plant of the complex engineering systems across multiple domains, automatically designs the control system and produces consistent control software components. In this chapter, the intelligent modeling system is verified to design and optimize a multirotor UAV which is a typically mechatronic product. It is expected to find the optimum solutions for fulfilling the top level requirements from potential customers while satisfying all the design constraints in the mean while. The intelligent modeling system is challenged by following requirements:

- Generation of consistent design representations for multidisciplinary analysis;
- Solving the problem of complexity for modeling, simulation and evaluation of the E/E systems;
- Automation of repetitive design activities for the whole optimization process.

It is assumed that the potential customer wants an multirotor UAV with four rotors (quadrotor UAV) which can carry the most payload and cost the least. Moreover, the target multirotor UAV should flight faster than 4 m/s and 2 m/s for the maximum climb rate and maximum speed, respectively. Furthermore, the flying qualities of the possible designs are expected to level 1 or level 2 in several complex test maneuvers, such as hovering turn, lateral reposition, vertical maneuver and descending 360° circle. The definition of the levels are referred to Ref. [137].

6.2. COMPLEX AND REPETITIVE DESIGN TASKS

The design and optimization of multirotor helicopter is a complex and repetitive design task. On one hand, in order to evaluate the flight performance of the multirotor UAV, the intelligent modeling system has to generate the multiphysics simulation model. which is a complex task. As can be seen in Figure 6.1, the multiphysics simulation model is composed of the dynamics model of the physical plant, a model based inversion controller, several components from multidisciplinary libraries and the simulation configurations. There are many decision variables for the geometric modeling, dynamics modeling and simulation configurations. The intelligent modeling system has to ensure a consistent design representation from the mechanical design (geometric modeling) to the integration of E/E systems (dynamics modeling and control system design) and multidisciplinary components. Moreover, in order to deliver a reasonable design, the intelligent modeling system must ensure the components selected from multidisciplinary libraries are compatible during the model generation. Furthermore, every multiphysics simulation model is integrated with a model based inversion controller, tested in five flight maneuvers and checked by twenty-two design rules (constraints) both from the

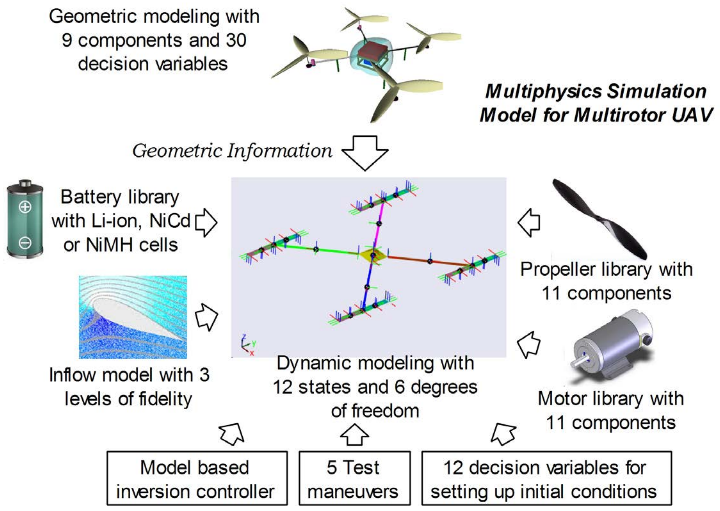

Geometric modeling with 9 components and 30 decision variables
Multiphysics Simulation Model for Multirotor UAV
Geometric Information
Battery library with Li-ion, NiCd or NiMH cells
Propeller library with 11 components
Inflow model with 3 levels of fidelity
Dynamic modeling with 12 states and 6 degrees of freedom
Motor library with 11 components
Model based inversion controller
5 Test maneuvers
12 decision variables for setting up initial conditions

Figure 6.1: The complexity of modeling the multirotor UAV in multiple domains

mechanical domain and the E/E systems. Therefore, the construction of multiphysics simulation model can be somewhat more complex than the multirotor UAV itself.

On the other hand, there are also many repetitive design activities during the optimization process from modeling, analysis and evaluation. As is shown in Figure 6.2, the optimization framework of the quadrotor UAV includes twenty-two design activities for each chromosome. After the generation of initial population, the geometry models of quadrotor UAV are firstly

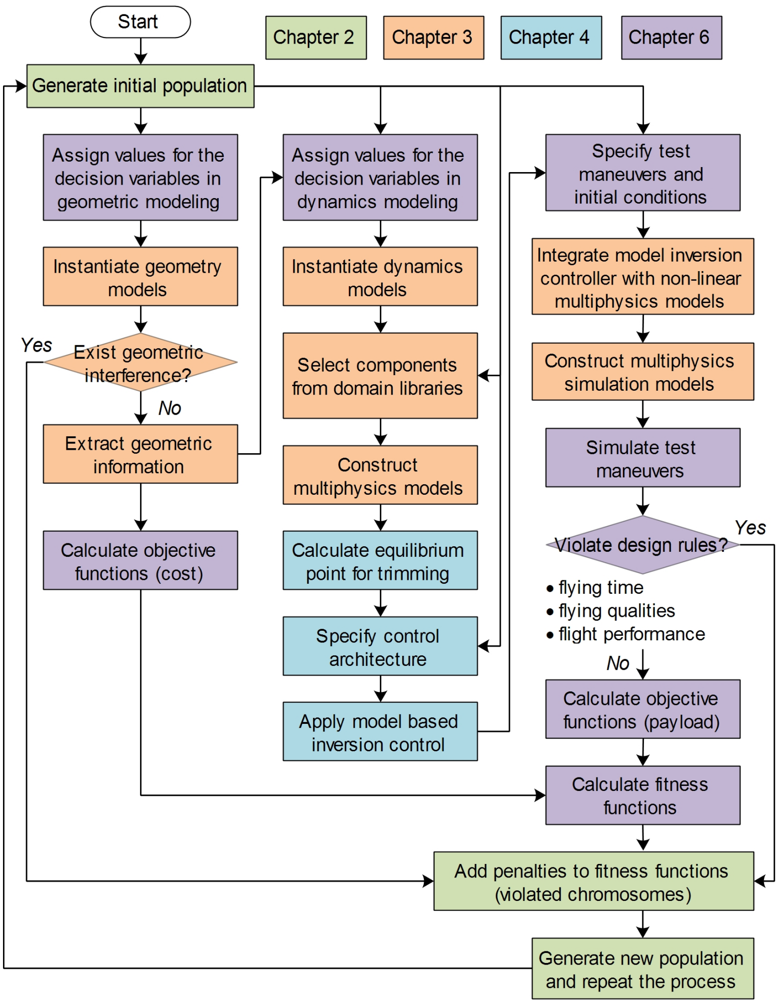

Start
Chapter 2
Chapter 3
Chapter 4
Chapter 6
Generate initial population
Assign values for the decision variables in geometric modeling
Instantiate geometry models
Exist geometric interference?
Yes
No
Extract geometric information
Calculate objective functions (cost)
Assign values for the decision variables in dynamics modeling
Instantiate dynamics models
Select components from domain libraries
Construct multiphysics models
Calculate equilibrium point for trimming
Specify control architecture
Apply model based inversion control
Specify test maneuvers and initial conditions
Integrate model inversion controller with non-linear multiphysics models
Construct multiphysics simulation models
Simulate test maneuvers
Violate design rules?
Yes
• flying time
• flying qualities
• flight performance
No
Calculate objective functions (payload)
Calculate fitness functions
Add penalties to fitness functions (violated chromosomes)
Generate new population and repeat the process

Figure 6.2: The repetitive design activities in different disciplines and software tools for the optimization

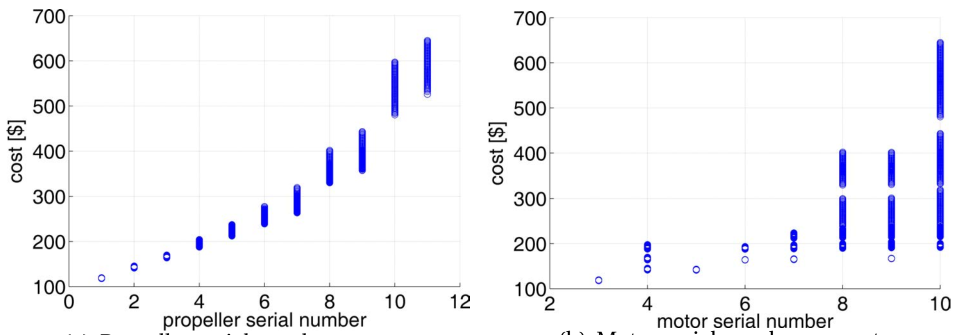
instantiated to check the geometric interference. The process may be terminated if there are some geometric interferences among the components. Otherwise, the dynamics models are built and integrated with the components from multidisciplinary libraries, such as the inflow model, propeller, motor, etc. Next, the intelligent modeling system calculates the equilibrium point and specifies the control architecture for applying the model based inversion control. Then, together with the simulation configurations (the test maneuvers and initial conditions), the model inversion controllers are integrated with the multiphysics models to construct the multiphysics simulation models which are evaluated in the flight maneuvers. The simulation results are checked by several design rules. If no rule violates, the fitness functions are calculated based on the cost and payload for the current design. On the contrary, the processes are terminated and penalties are added to the fitness functions.

It should be pointed out that the intelligent modeling system has to switch from the internal intelligent modeling system to the analysis tools and read the results back. Thus, the requirements from multidisciplinary analysis also somewhat increase the complexity of design and optimization. It can be expected it will cost much time and effort to manually model the multirotor UAV, simulate in test maneuvers and check all the constraints.

6.3. SPECIFICATION OF INITIAL CONCEPTS

First, the customer requirements are used to find the initial points by reversely calculating the search space. An abstract of the search space is built by the GA in the solver of the optimizer with the method in Section 2.2.6. When there is no design mission, the solver of the initiator randomly chooses the components with the GA method, like propeller, motor, frame, battery and other accessories to generate possible solutions. The cost of every possible solution is calculated by adding the price of all components together. Thus, it is enable to reversely find out the corresponding components if the cost is given.

Figure 6.3(a) and 6.3(b) shows the relationship from the propeller and motor to the cost of the quadrotor UAV. Every point represents a feasible solution, and all the unfeasible designs are eliminated by the natural selection of the GA. It should be pointed out that the propeller sold on the market is a standard part which is provided by manufactures. Eleven propellers are selected with sizes from 5 $inch$ up to 18 $inch$. Thus, the decision variable for the propeller is the serial number, which is an integer. The motor is treated in a similar way with eleven selections from small to big motor. As mentioned, the customer wants to pay the least while get the maximum payload. If the upper bounds for the cost is set to 500 \$, it can be seen than the combination of the propeller from one to ten and the motors from three to ten are able to achieve feasible solutions by



(a) Propeller serial number vs. cost (b) Motor serial number vs. cost

Figure 6.3: The corresponding relationship from the propellers, motors to the cost of quadrotor UAVs

reversely feeding this requirements with Figure 6.3(a) and 6.3(b). Therefore, every customer requirement is viewed as a criteria to specify the range of decision variables. By adding all the range together, the initial points and corresponding boundaries for the optimization are obtained, which are stored in the initiator.

6.4. OPTIMIZATION CONFIGURATION

Unlike transitional iterated design processes, the intelligent modeling system designs the quadrotor UAV in an optimized fashion. Thus, the requirements from the customer are converted into the objective functions and constraints, which are shown as follows:

$$min f_1(\vec{x}) = \sum_{i=1}^{n} C_i(\vec{x}_k), \qquad i = 1,2,...,9 \tag{6.1}$$

$$max f_2(\vec{x}) = T - g * \sum_{i=1}^{n} W_i(\vec{x}_k), \; i = 1,2,...,9 \tag{6.2}$$

$$\vec{x}_k^T = [x_{k1}, x_{k2}, ..., x_{k32}], \; k = 1,2,...,30 \tag{6.3}$$

$$g_j(\vec{x}_k) \leqslant 0, \qquad j = 1,2,...,22 \tag{6.4}$$

$$fitness(\vec{x}) = 0.5 rank(f_1(\vec{x})) + 0.5 rank(-f_2(\vec{x})) \tag{6.5}$$

As mentioned, the GA has been integrated within the optimizer of the intelligent modeling system. The optimization starts with the initial population of thirty chromosomes and loops for one thousand generations. Every chromosome has thirty-two decision variables which are used for initialing the model generation and simulation configurations. The cost and payload are given equal weight values in the fitness function (equation 6.5).

6.4.1. CALCULATION OF OBJECTIVE FUNCTIONS

As can be seen in Figure 6.4, the objective functions for the cost and payload are separately calculated and then combined as the fitness function. On one hand, the components are classified as standard and non-standard components. The cost for the non-standard components is computed by multiplying the weight with the cost for the material. The prices for the standard components are searched from the product database. On the other hand, it is assumed a payload for the current configuration which is evaluated in serial flight maneuvers. If all the requirements from the flight performance and flying qualities are satisfied, the payload value is saved. Otherwise, the GA adds penalties to the fitness function. Finally, the value of the fitness functions is obtained by ranking the cost and payload with proportional function.

6.4.2. EVALUATION OF CONSTRAINT FUNCTIONS

Typically, the design and optimization of multirotor UAV includes many constraints, which are classified in several disciplines in Table 6.1. Every chromosome (solution) in the population is evaluated by all the constraints to see whether it is feasible or not. As indicated in Section 2.2.4, the unfeasible solutions will be deleted in further generations due to the natural selection mechanism of the GA.

Take the evaluation of duration as an example (Figure 6.5), the battery capacity is calculated by multiplying the energy density specified by the simulation experts with the volume extracted from the geometric definition of

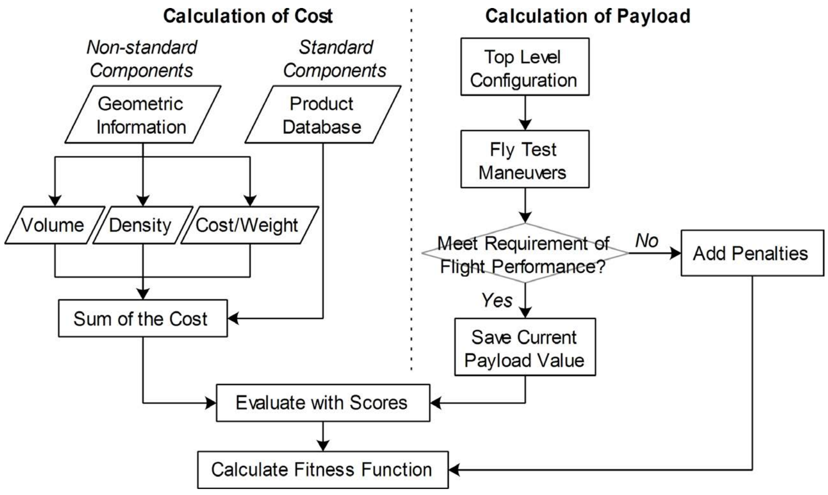


Figure 6.4: Calculation of objective functions for quadrotor UAVs

Category	Constraints
Mechanical domain	Geometric interference check for all close components
Dynamic domain	Limitation of roll/pitch/yaw rate
Control system design	System delays, bounds of input commands
Flying qualities	Flying qualities level 1 or level 2 in all test maneuvers
Flight performance	Minimum duration ≥ 10 min
-	Maximum climb rate ≥ 4 m/s
-	Maximum speed ≥ 2 m/s

Table 6.1: Part of constraints considered during the construction of multiphysics simulation model across multiple domains

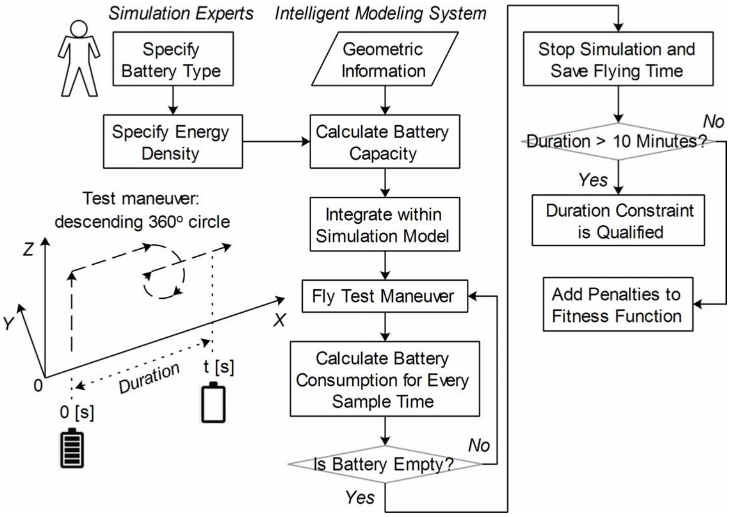

Simulation Experts
Intelligent Modeling System
Specify Battery Type
Geometric Information
Specify Energy Density
Calculate Battery Capacity
Test maneuver: descending 360° circle
Integrate within Simulation Model
Z
Y
X
0
Fly Test Maneuver
Duration
t [s]
0 [s]
Calculate Battery Consumption for Every Sample Time
Is Battery Empty?
No
Yes
Stop Simulation and Save Flying Time
Duration > 10 Minutes?
No
Yes
Duration Constraint is Qualified
Add Penalties to Fitness Function

Figure 6.5: Example of evaluation of constraint for duration

the battery package. Then, a amount of battery capacity is assigned to battery model in the multiphysics simulation model which is simulated in the test maneuver of descending 360° circle. Typically, the relationship from the speed of the multirotor UAV in every sample time to the battery consumption can be set up according to the method suggested by Larminie [151]. After a 360° descending circle, the multiphysics simulation model continuously flies in the X direction until the battery is empty at what time the simulation is stopped. Next, if the duration is larger than 10 minutes in this case, the chromosome (current solution) is passed for the duration constraint and checked for the others. Otherwise, it will be added with penalties to corresponding fitness function. Other constraints, such as mechanical interference check, are also treated in a similar way, which are not discussed in details.

6.5. VALIDATION OF MULTIPHYSICS SIMULATION MODEL

In order to ensure the level of fidelity of the multiphysics simulation model instantiated from the MIM is sufficient to be used in a design process and for control system design, it is compared to flight test data of a quadrotor UAV, the QS4 test bench, from literature. The QS4 test bench is a light weight vehicle with four propulsion groups. Every propulsion group is composed of a motor (29 g), a gear box (6 g) and a propeller (6 g) [152]. The same parameters are set, such as mass, dimensions, rotor speed, as the QS4 test bench to configure the multiphysics simulation model. However, there might be several differences between the QS4 test bench and the simulation model, like control law, inertia of the bodies and rotor flapping, which cannot be determined from the literature due to limited information (Table 6.2).

Initially, the multiphysics simulation model is compared to the QS4 test bench for a step input into the roll axis. As illustrated in Figure 6.6(a), the multiphysics simulation model of quadrotor UAV shows similar maneuver response as the QS4 test bench, which in turns demonstrates the dynamics characteristics and the control system of the multiphysics simulation model.

Then, both of the QS4 test bench and the multiphysics simulation model are tested in a more complex case, the four way-point maneuver (Figure 6.6(b)). The flying speed is set to 0.5 m/s for both of them. Compares to the QS4 test bench, the multiphysics simulation model shows a more smooth turning at the corners. However, both of them start and end at almost the same point which means they track the maneuver properly. Therefore, the multiphysics simulation model and model based inversion controller are validated to

Same configurations	Different configurations	
	QS4 test bench	Simulation model
Motor mass and speed	-	-
Propeller size and mass	-	-
Frame size and mass	-	-
Flying speed	-	-
Flying altitude	-	-
Test maneuver	-	-
-	Compensated controller	Inversion controller
-	Gearbox	Direct drive
-	No wind	Inflow model
-	Actual body inertia	Computed body inertia
-	rotor flapping	No flapping

Table 6.2: The simulation configurations for the QS4 test bench [152] and the multiphysics simulation model

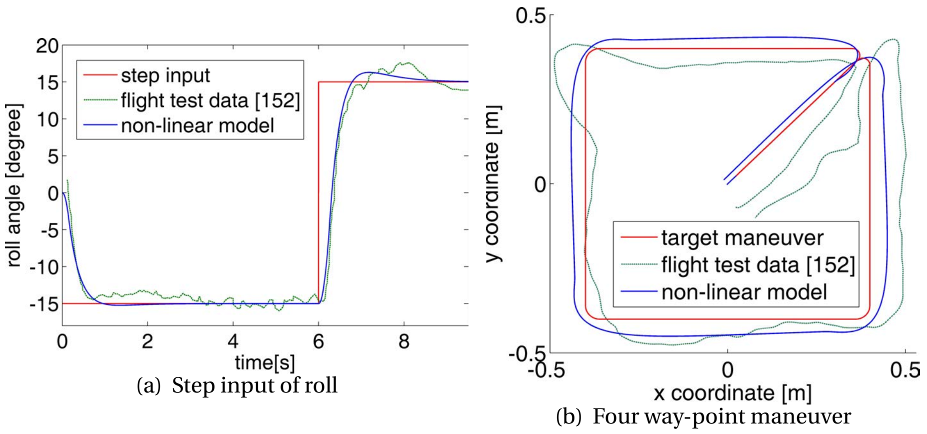


Figure 6.6: Validation of multiphysics simulation model with the flight test data of QS4 test bench

somewhat both in the dynamics of the multirotor UAV and the control systems.

6.6. EVALUATION OF FLIGHT PERFORMANCE AND FLYING QUALITIES

Thirty thousand design solutions are generated during the optimization process. Besides the mechanical constraints, the performance and flying qualities of every design has to be evaluated, for which the multiphysics model and corresponding control system is needed. Since it costs much time and effort to manually adjust the variables of the control system for all of these designs. Therefore, model based inversion control is integrated in the intelligent modeling system to automatically tune the control system for every possible solution. In order to clearly show the differences, three possible designs with size varying from small to large, are compared with each other (see Table 6.3). To demonstrate the difference in flight performance and handling qualities, one prescribed handling qualities requirement and one test maneuver are presented as example. The moderate-amplitude pitch attitude change handling qualities requirement is evaluated to determine the agility of the different designs in the pitch axis. In addition, the descending 360° circle is presented as example maneuver for evaluation of the “assigned” handling qualities. All configurations carry no payload during the test maneuvers.

The quadrotor should respond as rapidly as possible from one steady state to another when the desired pitch (roll) rate is set to a specific large value. The required pitch (roll) angle is set to vary from 5° to 25°. The results of three example quadrotor configurations are illustrated in Figure 6.7. It is apparent

Configuration		1	2	3
Variable	Unit			
Propeller ID	-	2	6	10
Motor ID	-	4	8	10
Propeller size	*inch*	6×3	10×5.5	16×5.5
Propeller weight	*g*	6	15	40
Payload up to	*g*	80	600	2400
Propeller cost	$	1.8	2.9	7.9
Motor speed	*rpm*	10600	8200	6400
Motor weight	*g*	19	27	88
Motor cost	$	10	20	45
Frame size	*mm*	225	377	426
Frame weight	*g*	93	230	338
Frame cost	$	13	25	42
Gross weight	*g*	277	563	1053
Total cost	$	141	170	346

Table 6.3: Example of three quadrotor UAV configurations from 30,000 design variants (there is a significant difference in payload)

that the largest quadrotor has much better agility (level 1) than the medium (level 2) and small one (level 3). This can be explained by the fact that the large quadrotor has more control power, even though the inertia is also larger. The maximum airspeed of the small quadrotor is limited due to its size. Therefore, it executes the flight test maneuver at a reduced airspeed.

The results show that all configurations have similar performance in the

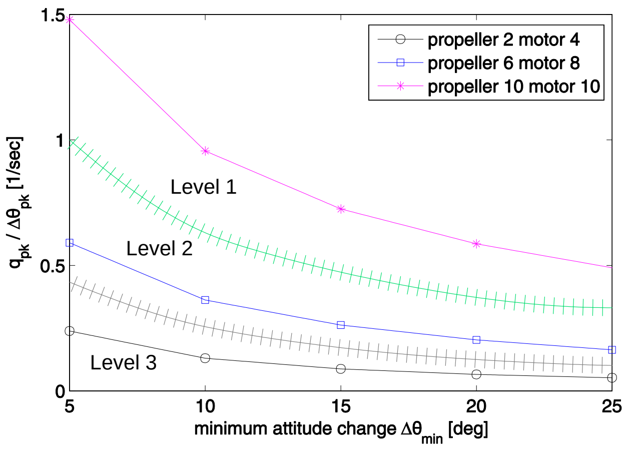


Figure 6.7: Moderate-amplitude pitch attitude change requirement in hovering for evaluating flight performance

vertical motion (Figure 6.8 and 6.9) but there are significant differences in the lateral and longitudinal motion, which can be seen in Figure 6.10, 6.11 and 6.12. First of all, the mass and inertia indeed have an effect on the control system of pitch and roll. Moreover, the deceleration for the quadrotor is a difficult task because the speed of four motors has to be reallocated while a specific altitude must be kept. Nevertheless, it can be observed that the three configurations complete the target maneuver almost completely within the limits of level 2 , which is adequate. Only the configuration with propeller ID 10 and motor ID 10 exceeds the level 2 boundary in the maneuver. Its fitness function therefore receives a penalty value by the GA. It should be noted that, all designs are able to be controlled regardless of the huge differences among the configurations. The control systems are successfully built by the model based inversion control method in every maneuver task element for all the designs. The results of flying qualities for three of the designs are summarized in Table 6.4.

Configuration	1	2	3
Propeller ID	2	6	10
Motor ID	4	8	10
Vertical maneuver	Level 1	Level 1	Level 1
Lateral reposition	Level 1	Level 2	Level 2
Hovering turn	Level 1	Level 2	Level 2
Descending 360° circle in xy plane	Level 1	Level 2	Level 3
Descending 360° circle in xz plane	Level 1	Level 2	Level 2
Final grade	Level 1	Level 2	Level 3

Table 6.4: Summary of flying qualities for three quadrotor UAV configurations

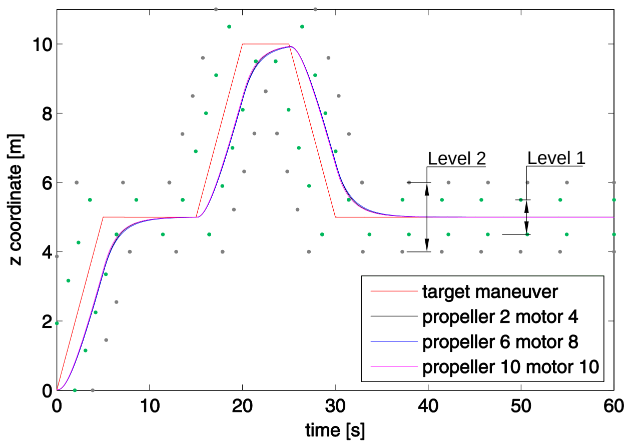


Figure 6.8: Flying qualities of three quadrotor UAV configurations in vertical maneuver (z plane)

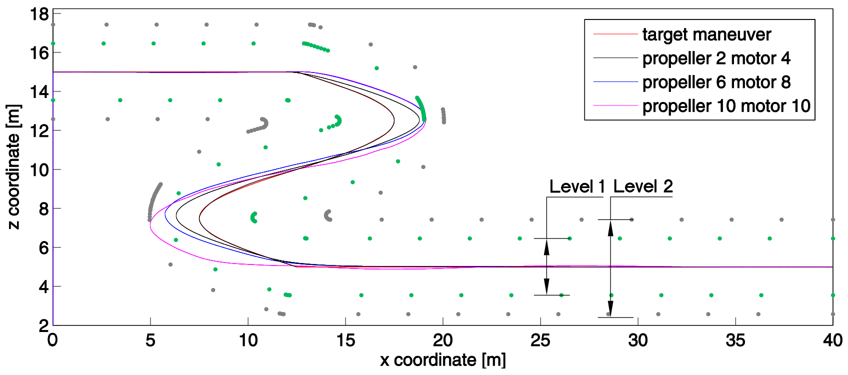


Figure 6.9: Flying qualities of three quadrotor UAV configurations in descending 360° circle maneuver (xz plane)

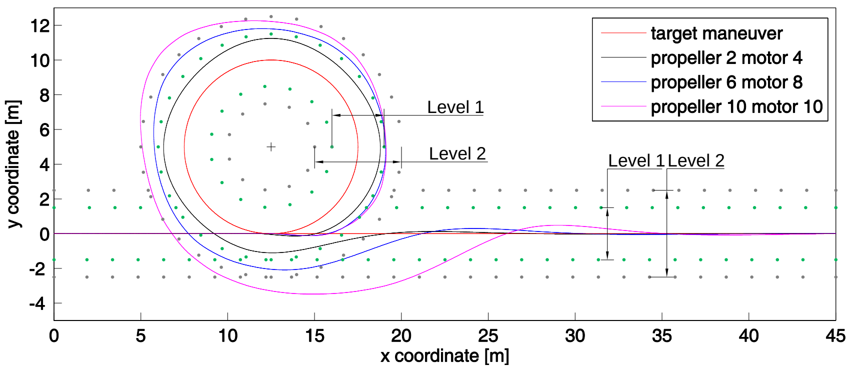


Figure 6.10: Flying qualities of three quadrotor UAV configurations in descending 360° circle maneuver (xy plane)

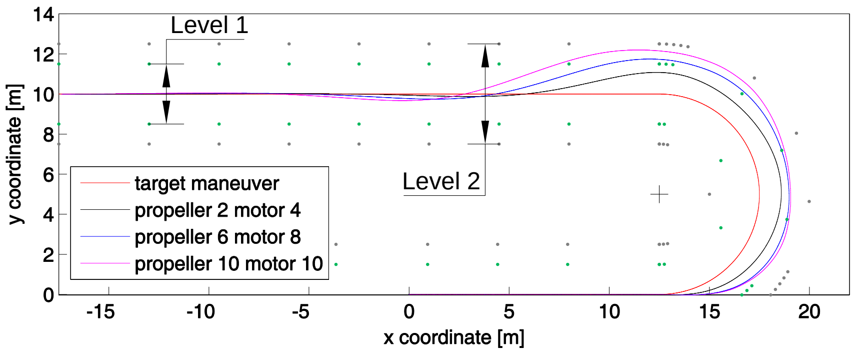


Figure 6.11: Flying qualities of three quadrotor UAV configurations in hovering turn maneuver (xy plane)

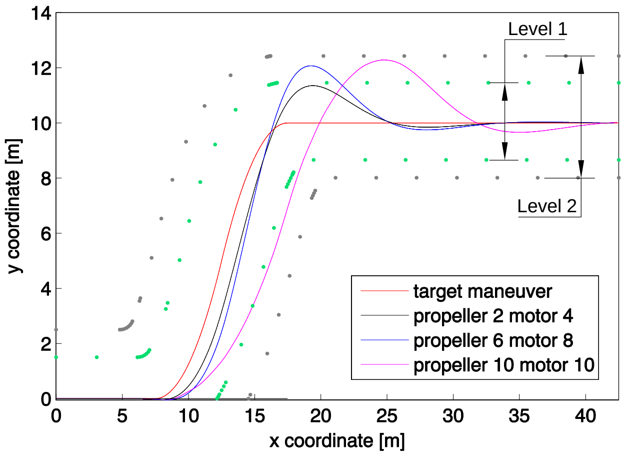


Figure 6.12: Flying qualities of three quadrotor UAV configurations in lateral reposition maneuver (xy plane)

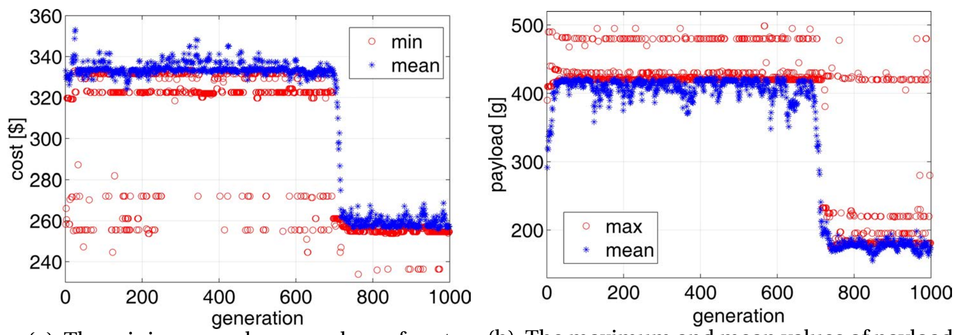
6.7. Optimization Results

After the specification of the initial points, bounds, objective functions, constraints, and test maneuvers, the intelligent modeling system starts to the find the optimum for two conflicting requirements. The evaluator of the optimizer will reduce the fitness values for the chromosomes violated in the constraints (which are infeasible solutions) by adding a penalty value. Next, new population are produced by the solver. The iteration stops when the user-specified maximum number of evolutions have been achieved (one thousand generation).

The optimization are convergent at two solutions: propeller ID 3 and motor ID 3 or propeller ID 5 and motor ID 6, which are shown in Figure 6.13(a) and 6.13(b). As can be seen in Figure 6.14, it reveals that either less money is spent but also less payload is carried on the quadrotor or a medium size design is selected and carry more. Moreover, it is not necessary to discuss which one is better, because it needs further information related to decision-making. In short, it shows that although the design space is large, the intelligent modeling system is able to automate the whole processes both for mechanical and electronic control system design, and finally achieve the optimum.

6.8. Summary

The intelligent modeling system, which is a KBE system, is verified in this chapter to design and optimize a multirotor UAV with four rotors for fulfilling the potential customer requirements. Two contrary optimization objectives are



(a) The minimum and mean values of cost (b) The maximum and mean values of payload

Figure 6.13: Comparison of the minimum (or maximum) values with the mean values

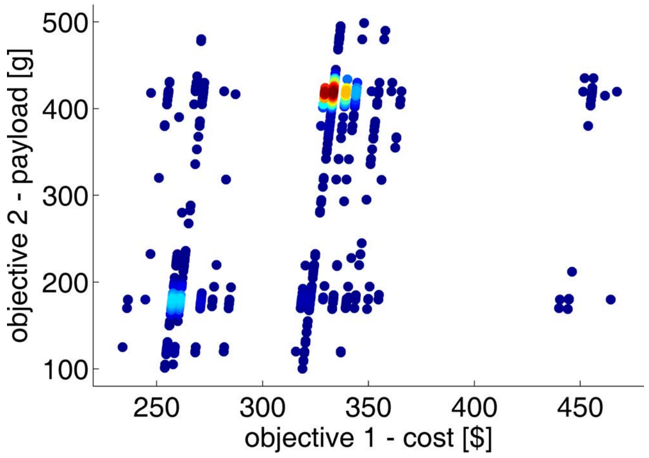


Figure 6.14: Results of objective functions during one thousand generations

proposed with many design constraints from mechanical design, flight performance, flying qualities, performance, etc. Next, the GA is selected in this research study to organize the optimization framework. It has shown how the intelligent modeling system calculates the objective functions and evaluates the design constraints in multiple domains. Soon after, it has been pointed out that the design and optimization of multirotor UAV is complex and repetitive tasks, which is the focus of this research study. Firstly, the multiphysics simulation model of multirotor UAV which is generated by the intelligent modeling system is compared with a real small quadrotor UAV. The multiphysics simulation model shows similar flight dynamics as the real test bench which in turn validates the control system as well. Secondly, thirty thousand designs are simulated in five maneuvers to evaluate the aircraft performance and flying qualities. Three of them are compared as an example. The results show that although the configuration of various multirotor UAVs might be quite different, the control systems are properly tuned by the intelligent modeling system to evaluate the maximum performance of the rotorcraft within its flight envelope. The model based inversion control is also somewhat validated by delivering reliable control systems for thirty thousand designs. This proves to be a powerful method in a design optimization because all the parameters in the control law directly depend on a linear model of the multiphysics simulation. Model based inversion control has some disadvantages when applied in a realistic environment. It is therefore possible to design a different type of control law for the final design. This control law only needs to be tuned once. Finally, the optimization process is convergent at two optimum solutions for the multirotor UAVs after one thousand generations.

The results show that although modeling of the multirotor UAV is a complicated task, the intelligent modeling system successfully generates the multiphysics simulation models for the entire design envelope of a complex engineering application. On one hand, the multiphysics simulation models can support the analyses in multiple domains, such as the evaluation of the flight controls and the estimation of electric energy consumption. On the other hand, the MIM are consistent among the disciplines, not only for modeling the mechanical components but also for designing the control systems and other E/E systems. Moreover, the intelligent modeling system integrates the E/E system development together with the analyses in other disciplines to achieve the benefits of the MDO. Furthermore, it also shows an automated fashion to design and optimize complex mechatronic system in terms of modeling, simulation and evaluation.

7 CONCLUSIONS AND RECOMMENDATIONS

7.1. CONCLUSIONS

NOWADAYS, more and more functions are electronic controlled on aircraft, which towards the more and even all electric aircraft. It has been found out that novel electric concept cannot be properly developed by traditional aircraft design method due to extreme complexity of the E/E systems. Moreover, because the aircraft design is a multidisciplinary task, the E/E systems should be concurrently developed with other disciplines in the design phase. However, the development of the E/E systems is a difficult task, which are summarized in the following three items:

- **Setting up simulation models** for the E/E systems is a complex and multidisciplinary task;
- It costs time and effort to **design consistent control systems** for the entire flight envelope;
- The development of **control software components** is prone to errors due to **inconsistency** from the top level configuration, control architecture to the software components.

A better result can be obtained when the software tools are capable of solving multidisciplinary problems not only limited to mechanical domain or E/E systems. New methods and tools are required to concurrently design the E/E systems together with the other engineering domains.

Therefore, this research proposes novel methods and tools to support the development of the E/E systems for unstable and unmanned aircraft by using the KBE approach. The objective of this research is to reduce the development time of complex engineering application by automating the design process of the E/E systems and create more consistent control system/software through the entire design envelope. Compares to related work, the proposed KBE system fills the gap between the mechanical design and the E/E systems, which is characterized by :

- Ensuring a consistent design representation both for the mechanical domain and the E/E systems;
- Synchronous design of mechanical systems, control systems and software components to enable the benefits of the MDO approach;
- Automating the mechatronic design both for modeling the mechanical components and developing the control systems.

A multirotor UAV configuration is used as test case to demonstrate the novel methods and tools described above. It takes a lot of time to manually develop a single multiphysics simulation model, representing a single design of the multirotor UAV, taking into account all discipline couplings and interactions between components. Moreover, as explained before, it may be necessary to develop multiple multiphysics simulation models for a single design in order to evaluate different design requirements. In order to represent the complete design envelope of the multirotor UAV (with highly different top level aircraft configurations) a vast number of different multiphysics simulation models is required. Furthermore, the multirotor UAV is inherently unstable and it is unmanned. It therefore requires a relatively complex control system which needs to be developed for each design under evaluation.

Because the development of this vehicle involves several disciplines and many subtasks, a novel intelligent modeling system is proposed to enable MDO for aircraft (and other vehicles) with a high level of E/E systems integration. The intelligent modeling system is an extension to the so-called DEE, a KBE system proven useful for conceptual and preliminary aircraft design. Unlike the original DEE designed for aircraft conceptual design, the intelligent modeling system focuses more on the multiphysics modeling aspects of the physical system, design of the control system and generation of software code for the E/E systems. Compares to other DEE systems, the intelligent modeling system generates the models to support the E/E system specifically in relation to the following three aspects:

- Automated generation of multiphysics simulation model to support the MDO (Chapter 3);
- Automated evaluation of inherent aircraft agility by performing maneuvers within an MDO framework (Chapter 4);
- Automated generation of consistent control software components (Chapter 5).

Then, the concept of a MIM which is the core of the new intelligent modeling system is proposed that can automatically generate multiphysics simulation models to support the MDO process by using a KBE approach. The intelligent modeling system is an application which consists of three software modules, responsible for; (1) knowledge acquisition (the "inference engine"), (2) knowledge application (the "MIM") and (3) the communication with analysis tools (the "communication framework"). This system has to overcome three challenges:

- It must deal with the problem of system coupling and interactions across disciplines in the process of multiphysics simulation model generation;
- It has to represent design knowledge from multiple engineering domains in an appropriate format that ensures consistency across the different disciplines, especially when considering the use of control software for E/E systems and its relation with the physical system;
- It must be able to generate multiphysics simulation models with the right level of fidelity for the problem at hand.

It may be required that mono disciplinary analysis tools are used before the multiphysics simulation model can be developed. The intelligent modeling system specifies the sequence of these analyses by collecting knowledge from e.g., textbooks, simulation experts and design experts. A calculation sequence may also be required within the multiphysics simulation model. This sequence is also defined by the system. Next, connections are created by the communication framework with target tools and the analyses are sequentially performed. Moreover, in order to select appropriate components from simulation libraries, the intelligent modeling system defines logical expressions to capture the expertise from simulation experts.

To ensure consistency, a unified knowledge representation, the MIM, is used. The different characteristics of the complex system to be designed are represented in this information model. Physical entities of the product are modeled as classes and non-physical elements (such as control software) are modeled with functions. Every class (or function) is assigned with attributes (or variables) to represent different aspects of the physical entities (or non-physical information). The challenges of system coupling and the interactions across disciplines are taken into account by means of the knowledge acquisition process.

Next ,the intelligent modeling system instantiates the MIM into submodels with the modeling kernels. Finally, the intelligent modeling system constructs the multiphysics simulation model. This model is composed of the physical plant, control systems and files required for executing the simulations, such as trim algorithms, simulation configurations, etc.

The methodology is demonstrated by the automatic generation of multiphysics simulation models for a multirotor UAV. There are five coupled disciplines in this use case. After the knowledge acquisition phase, the intelligent modeling system successfully constructs the MIM. It defines classes with attributes to capture geometric information. These classes also have attributes required for modeling of the dynamics, such as rigid bodies, joints, masses and inertia. Other design knowledge is modeled in data files. Finally, the MIM is instantiated by modeling kernels into submodels which are integrated into a complete multiphysics simulation model. In addition, script files needed to perform simulations are also created. For this test case, MATLAB Simscape is used as environment for multiphysics simulation. The final model can be used to evaluate the flight performance of the multirotor UAV when executing complex maneuvers.

The test cases in Chapter 3 show that the intelligent modeling system can generate the MIM which integrates the knowledge from multiple domains to represent the physical entities and non-physical information while considering the system coupling and the interactions among the domains. Then, the multiphysics simulation model can be automatically instantiated for serving the analyses in MDO.

Subsequently, a complete multiphysics simulation model is composed of the multiphysics model of the physical plant, the corresponding control systems and simulation configurations. Hence, Chapter 4 proposes methods and tools to automate the design of control system for the evaluation of test maneuvers in the MDO.

Due to extreme complexity of the E/E systems, there are several challenges for the automated design of control systems. It costs much time and effort to specify the strategies and parameters for the establishment of control systems, configure the simulation initial conditions and design a consistent control system. Although model based inversion a promising method to accelerate the design process for the control systems, it needs methods and tools which support the process automation not only limited to the E/E systems but also for the MDO framework.

Besides modeling the physical entities, the intelligent modeling system can also automate the design process within and out of the intelligent modeling system environment. In this research study, it automates the development of control systems by capturing the process information from the model based inversion control approach. Firstly, it collects related knowledge of the control systems, such as the strategies for control allocation, the values for parameters and constants, the procedures for the model based inversion control, and the information for analysis tools. Secondly, it models the collected knowledge into functions, data files and script files. It also generates the multiphysics model of the physical plant for the further simulation in the analysis tools. Thirdly, the intelligent modeling system connects with the analysis tools and performs the script file which contains the procedures of model based inversion control approach to trim and linearize the multiphysics model together with other simulation configurations and finally inversely get the model based inversion controller. The inversion controller can be implemented on the linear and nonlinear multiphysics model to construct the multiphysics simulation models which are used for the evaluation of maneuvers in the MDO.

The intelligent modeling system is tested to design the control systems for the multirotor UAVs with configuration and topology differences. After the simulation configurations, the intelligent modeling system generates both of the geometry models and the dynamics models integrated with aerodynamic components (the inflow models) and electric power systems (the motors). Then, the model inversion controllers are obtained by trimming, linearizing and inversing the multiphysics models at every equilibrium point for each multirotor UAV configuration. Finally, the multiphysics models are integrated with corresponding model inversion controllers and simulation configurations and then simulated in three test maneuvers. The results show that all the multirotor UAVs are controllable though they are quite different in terms of configuration and topology structure. It can be expected that the intelligent modeling system can ensure a consistent control system for the entire flight envelope of multirotor UAV.

In short, the intelligent modeling system cannot only model the physical entities in multiple domains but also automatically design the control systems for the evaluation of maneuvers in the MDO. The performance of the E/E systems can quickly provide feedback to the MDO framework, which in turns establishes a concurrent design environment together with other disciplines.

Next, control software components are required for testing whether these components perform as desired in all possible operating conditions. Control software can be very lengthy and complex. There is a gap between the software components (structure and variables) with the other engineering disciplines related to the physical design and the overall control architecture.

The intelligent modeling system is proposed to automatically generate consistent control software components to support the MDO in Chapter 5. In order to avoid software errors and produce the control software efficiently, the focus is on two challenges:

- Generation of consistent control software from the top level physical configuration of the product and the control architecture to the software components;
- Automated generation of software code for the control system which is easy to understand.

Typically, software components are composed of three submodels: the data model, the behavior model and the real-time model. As for the data model, the intelligent modeling system extracts variable definitions and values from the physical system (or the control architecture) to specify the variables in the control software components. The variable definitions and values may come from the classes used for modeling the physical entities in Chapter 4 or from the variables and parameters in the control system (Chapter 5). Moreover, the intelligent modeling system divides the software components into basic elements which can be reorganized according to the requirements. It writes the basic elements into strings, (targeted to any computer language). The contents in the basic elements, such as variable definitions, can also be replaced. Finally, the data model and behavior model are tested on the real-time model which in this case is the multiphysics simulation model.

This capability of the intelligent modeling system is demonstrated on two test cases: The generation of software components for the flight control system of the multirotor UAVs and for the ABS of a novel road vehicle configuration. When the control architecture of the multirotor UAV changes, the intelligent modeling system can automatically select the appropriate basic software elements and combine them together to produce a new software component that solves the control allocation problem. For testing, the intelligent modeling system generates a piece of C code for the multirotor UAV which is compiled in MATLAB to build an S-function. The S-function is used to compare the original control system built by MATLAB Simulink blocks with the C code for a complex test maneuver. Results show that the S-function performs exactly the same as the original control system, which validates the C code produced by the intelligent modeling system.

In the second test case, the intelligent modeling system generates the source code (C code) for an ABS logic controller used on road vehicles. The MIM links the values in the software component (logic controller) with the physical configuration of the road vehicle. Again, the C code is compared with the MATLAB Simulink blocks for a straight braking maneuver with an initial speed of 100 km/h. There are only very small differences between the C code and MATLAB Simulink blocks due to different sampling methods used.

Next, the automatic development of the logic controller compiled as C code is tested in a design case: two road vehicle configurations with different front axle load ratios. The intelligent modeling system successfully captures the associated design knowledge and automatically generates properly working C code which is easy to understand.

In summary, the core of the intelligent modeling system, the MIM, connects the physical modeling, control system design with the generation of software components, which can provide a consistent control software component to support the development of E/E systems. Because the whole process can be automatically completed, it is possible to integrate the proposed methods and tools in an MDO framework. This approach can be very valuable if physical design changes are made to an existing vehicle which has lengthy and complex control software (millions of lines of code).

Finally, the intelligent modeling system for the development of the E/E systems is verified in Chapter 6 to design and optimize a multirotor UAV with four rotors for fulfilling the potential customer requirements. Two contrary optimization objectives are proposed with many design constraints from mechanical design, flight performance, flying qualities, performance, etc. Next, the GA is selected in this research study to organize the optimization framework. It has shown how the intelligent modeling system calculates the objective functions and evaluates the design constraints in multiple domains. The optimization starts from thirty initial solutions and loops for one thousand generations. Every solution is evaluated in five flight maneuvers and checked by many design constraints. The results show that although modeling of the multirotor UAV is a complicated task, the intelligent modeling system successfully generates the multiphysics simulation models for total thirty thousand designs with different configurations. On one hand, the multiphysics simulation models can support the analyses in multiple domains, such as the evaluation of the flight controls and the estimation of electric energy consumption. On the other hand, the multiphysics simulation models are consistent among the disciplines, not only for modeling the mechanical components but also for designing the control systems and other E/E systems.

In summary, the proposed KBE system concurrently develops the control software components of the E/E systems together with other disciplines for a multirotor UAV. Compare to manually modeling in the analysis tool, the proposed method and tools save much time and effort to model the complex systems across multiple domains. Moreover, it can ensure consistent control systems and software components for the whole design envelope, which avoids errors due to repetitive design activities. Furthermore, the performance of the E/E systems can quickly provide feedback to the MDO framework, which in turn establishes a concurrent design environment together with other disciplines. Finally, it also shows an automated fashion to design and optimize complex mechatronic system, which can be integrated within the MDO framework.

Although this research study applies the concept of MIM to model the mechatronic products, mainly for the mechanical design and control systems, the concept itself doesn't restrict to cover limited disciplines. This concept can be used to represent almost any complex systems which requires the analyses in multiple domains. It can be expected that the proposed KBE system is a promising tool for developing the MEA and even the AEA concepts in future.

7.2. RECOMMENDATIONS

This research study proposes a new concept, the MIM, to represent physical and non-physical information of the physical systems. The objective is to use one information model to ensure all the design representations across multiple domains are consistent throughout the MDO process. The multirotor UAV which is a typical mechatronic product is selected as the test case. Both the geometry and dynamics features are captured by defining the classes with attributes in the MIM. Next, the MIM is instantiated into a geometry model and multi-body dynamics model by modeling kernels. These models represent the physical plant of the multirotor UAV. Then, every multiphysics model is associated with a flight control system which is developed by applying model based inversion control approach. Because both the geometry model and multiphysics simulation model including flight control system are obtained from one source, the consistency of all the design representations are ensured.

Moreover, the stability and robustness of the control system is of key importance when embedded on the real vehicle. Moreover, in a real-life applications, the control system will be subject to noise. However, in the current study, the aim of the control system is to enable the evaluation of the inherent flying qualities of a large number of different designs within an MDO framework. For each different design and for each operating point (flight condition), a dedicated control system is developed automatically. Thus, the control system is used only within a simulation environment without the presence of noise. Modeling errors with respect to the real life vehicle therefore do not have to be considered during the design optimization and robust stability requirements can therefore be less stringent. Nevertheless, the control system must be stable on the nonlinear simulation model, which differs from the linear model on which the control system design is based. Since the linear model has a close correlation with the nonlinear model and a successful control system was designed for each of the quad rotor designs (in the order of 30.000 controllers) this is considered a statistical proof of robust stability.

Furthermore, a disadvantage of model based inversion control approach is that it only works well on a real-life application when the simulation model is highly accurate. Whereas, in other control methods, tuning of gains can be employed on the real vehicle. However, within an MDO process, this is not a problem since the model inversion controller is only used to control the simulation model. Thus, the use of model inversion control within an MDO process, enables the evaluation of the flying qualities of a large number of unstable aircraft configurations in the conceptual and preliminary design phase. Once the final design is obtained, there are two options for the control system design to be embedded on the real vehicle. A different more traditional control technique, which is more easy to certify and which requires tuning of gains, can be employed. The design of this control system only has to be conducted once and the simulation model required for the design is readily available. The second option is to use the model inversion control system. However, the accuracy of the simulation model has to be high enough to ensure the controller will perform well on the real-life vehicle.

Needless to say, the complexity of the MIM will increase when the complex products with more parts and assemblies and an integrated design with many relations across the product architecture. In order to handle such complexity, the classes (or functions) in the MIM must be carefully defined. The *don't repeat yourself* (DRY) principle can be applied to define the classes (or functions) with single and unambiguous definitions. Moreover, when the classes (or functions) are highly coupled with other definitions in the MIM, they should be considered and treated as a whole component. For example, there could be class definitions both for the wings and the fuselage of the aircraft, respectively. Since there are many interactions among the wings and the fuselage in the design phase, they can be integrated in a more larger class, like "wing-fuselage". Every time when the simulation expert wants to evaluate a new aircraft concept, he or she can directly refer to the class of "wing-fuselage" with minimum modifications instead of setting up the complex interactions among the wings and fuselage again.

In summary, the key of the MIM concept is that it focuses on capturing the intrinsic properties of physical systems by the KBE approach and a specific format of representation is avoided. This feature is quite different from traditional modeling languages where the information of the physical systems is coupled with the instantiation method. By defining the MIM, a mirror of the physical system is created in the computer environment, which can be used at any time regardless of the modeling kernels, software and hardware requirements. This will be a universal method for modeling a wide range of products for which there is a high level of domain integration.

Nevertheless, the MIM concept also has several limitations. First, the MIM concept is not suitable when the design mission only relates to one or two domains. It will take a significant amount of time and effort to collect the design knowledge from multiple engineering domains and the process information from the analysis tools and then to develop a working KBE system. It would be more straightforward to model the engineering system with other modeling languages rather than creating a MIM and then instantiating it. Therefore, a trade-off is necessary between the benefits of the proposed methodology and the time and effort required to implement the method in a software application. Nevertheless, engineers should be encouraged to use the proposed approach because the design framework implemented in a software application can be repeatedly used for further product development. Thus, the cost is temporary but a long-term benefit can be obtained when the complete lifespan of designing complex mechatronic systems is considered. Second, additional skills are required from the engineers developing the product. Engineers with expertise of a single discipline are now required to also have expertise in the field of software development and mathematics. Third, although it is able to generate many design solutions and their variants by instantiating the MIM with different inputs, human creativity is still needed to obtain truly novel solutions.

A

Automated Generation of High Fidelity Multiphysics Simulation Models for Truck-trailer Combinations

A.1. INTRODUCTION

The general aim of this appendix is to demonstrate that the techniques for the automated development of multiphysics simulation models can be used to obtain high-fidelity models and can be applied very broadly, to almost any type of vehicle.

Previously, several trailer manufacturers, universities and one axle manufacturer conducted a project named "FORWARD" in the Netherlands [54]. The objective of this project was to measure and identify real-life loading conditions on trailers during everyday operation and to use these for calculation of the fatigue on critical components, enabling weight reduction by re-design. A whole range of different trailer configurations was considered in the project, such as deep loaders, conventional trailers, trailers for transport of liquid cargo, double decker trailers, etc. Therefore, a generic multiphysics simulation model was required to predict the dynamic loads on particular elements of this range of vehicles when subjected to a range of maneuvers.

However, high-fidelity simulation of a large range of truck-trailer combinations with highly different designs is a challenging task. As can be seen in Figure A.1, a typical truck-trailer combination can be modeled by several main components, such as structural elements (suspension, truck cabin, etc.), aerodynamic forces, tire models, powertrain, steering system and a model representing the driver. For different trailer designs, different modeling

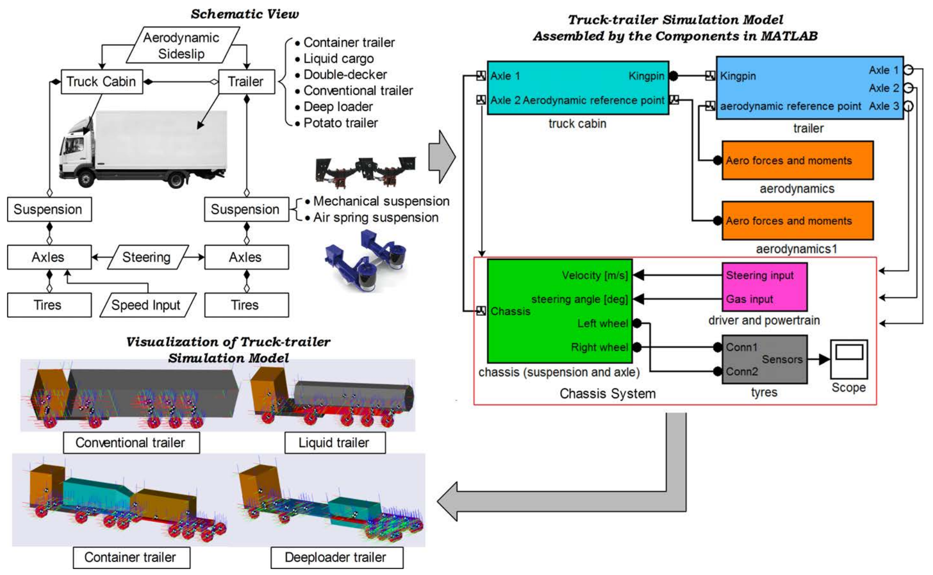


Figure A.1: Schematic view of truck-trailer combinations

components are required. For example, a leaf-spring suspension is modeled in a different manner than an air spring. In order to rapidly prototype a specific truck-trailer combination, a generic multiphysics simulation model is required. Moreover, in order to accurately predict the dynamic loads, high fidelity simulation models are necessary, especially for the suspension system.

Therefore, a complete example is given in this section to show how intelligent modeling system instantiates the MIM to generate both the data file and the model file for modeling the suspension systems. Next, the resulting high fidelity suspension model is integrated within the generic whole truck-trailer model and validated by comparison with data obtained from field test. The objectives of this test case are the following:

- Instantiation of a customized model from the MIM;
- High fidelity prediction of loads on critical components by the multiphysics simulation model;
- Validation of multiphysics simulation model with data from an actual field test.

MATLAB Simscape [112] is selected as the simulation environment in this test case. Typically, software tools supporting multiphysics capabilities can be classified into two groups. On one hand, some software tools are specialized for solving multiple simultaneous physical phenomena by utilizing the finite element method. ANSYS Multiphysics [153], COMSOL Multiphysics [154], ADINA [155], FEATool [156] are examples of commercial software packages for simulating multiphysics phenomena. On the other hand, other software packages, like Dymola [110] (which is based on the Modelica language [109]) and MATLAB Simscape [112], allow the user to assemble complex multiphysics models of the physical systems by selecting the pre-defined model blocks from multiple domain libraries. As mentioned in Section 3.3.3, it should be noticed that which modeling kernel (software package) is applied by the intelligent modeling system is determined by the target data (or file) format and the analysis tool. An advantage of using MATLAB Simscape is that new components can be created from basic mathematical equations and then integrated with existing model blocks to construct the multiphysics simulation model. This feature facilitates the modeling of linear or nonlinear characteristics for the suspension system.

A.2. CUSTOMIZED MULTIPHYSICS SIMULATION MODEL COMPONENTS

In order to accurately predict dynamic loading conditions, detailed high fidelity suspension models are required. The suspension of a trailer is typically composed of an air spring and a shock absorber. Due to its nonlinear characteristics, it is not straightforward to model the air spring and the shock absorber by simply using components from existing modeling libraries in MATLAB Simscape. Therefore a new component in the library must be created in this test case.

The reaction force of the spring and damper system of the suspension is calculated by equation A.1, which was obtained from handbooks.

$$f = k \times x + D \times D_v \tag{A.1}$$

The spring and damper coefficients, represented by k and D are both nonlinear functions of their position. Based on this equation, a new component (Simulink block) is created which can be selected by the intelligent modeling system when needed (see Figure A.2). It is assumed that the simulation/suspension expert defines the nonlinear stiffness and damper coefficients. In this case, they are represented by lookup tables. This information is collected by the MIM of the intelligent modeling system. Next, the intelligent modeling system instantiates the MIM into three model files at once. It generates data files which contain stiffness and damping coefficients. Moreover, source code is produced. This code, written in the MATLAB Simscape language, records the mathematical representation of the spring and damper system (equation A.1). Finally, a script is created which includes the commands to further operate the data files and the source code in the MATLAB environment. Next, the intelligent modeling system sets up a direct connection with MATLAB through the communication framework. The source code is first instantiated as a model block and then initialized by the data files. Finally, the new model block of a nonlinear spring and damper is integrated with the overall vehicle suspension model to create reaction forces on the vehicle body and unsprung weight. The nonlinear spring and damper model block is saved in the libraries, to allow for repeated use in other cases.

The complete multiphysics simulation model which can be generated automatically was developed and implemented according to the methodology described in Chapter 3.

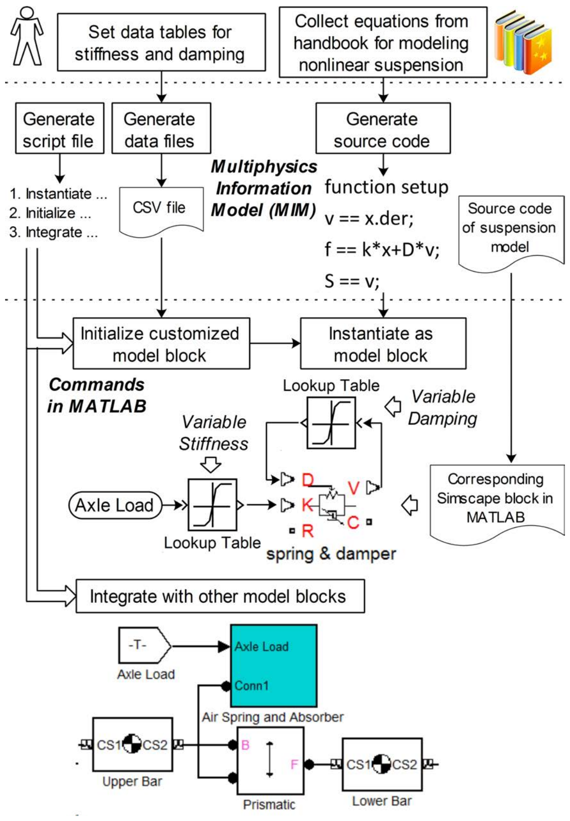


Figure A.2: Example of customized model instantiated from the multiphysics information model (MIM)

A.3. VALIDATION OF GENERIC HIGH FIDELITY MULTIPHYSICS SIMULATION MODEL

The multiphysics simulation model including the nonlinear suspension model is validated by comparison with data obtained from field tests. Two wheels of the trailer were instrumented with wheel force transducers to allow direct measurement of forces and moments on the wheel. The test maneuver is a double lane change in which the truck-trailer first steers to the left lane and then steers back to its original lane. Results are displayed in Figure A.3 and Figure A.4.

Results show that the vehicle dynamics (yaw rate) and all forces and moments acting on the tires are accurately predicted by the multiphysics simulation model. Because the multiphysics simulation model can be created and analyzed in an automated fashion, it is possible to integrate this tool within an MDO framework.

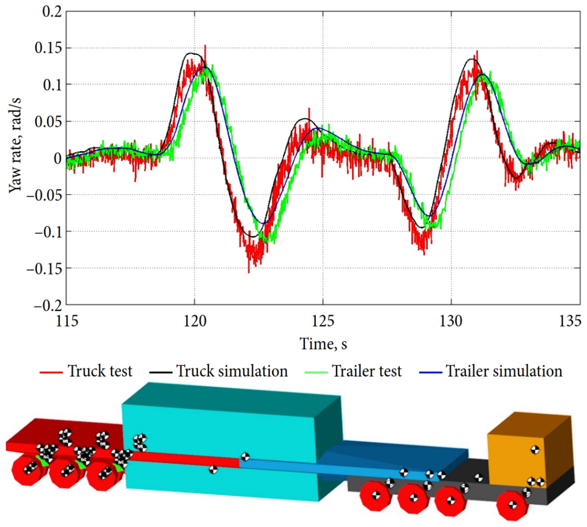


Figure A.3: Yaw rate during lane change maneuver [54]

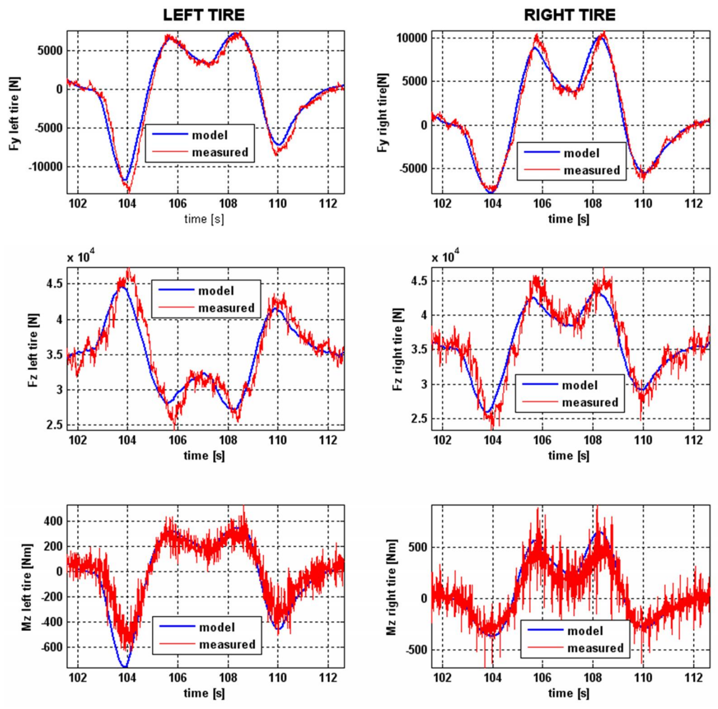


Figure A.4: Wheel forces validation during lane change maneuver

REFERENCES

[1] J. S. Cloyd, *Status of united states air force's more electric aircraft initiative,* IEEE AES System Magazine **4**, 17 (1998).

[2] N. Shah, W. Ho, M. Nordby, and A. Hall, *More electric aircraft F/A I8 cost benefits study,* Tech. Rep. (Air Force Contract F33657-90-D-0027, 1991).

[3] M. Eicke, E. Hodge, W. Price, P. Ravaris, and S. Talati, *Advanced secondary power system study for the F16,* Technical Report CRDL 34002, Air Force Contract F33657-84-C-0247 (Wright Patterson AFB, OH, 1984).

[4] M. Provost, *The more electric aero-engine: a general overview from an engine manufacturer,* in *International Conference on Power Electronics, Machines and Drives* (2002) pp. 246–251.

[5] D. Blanding, *Subsystem design and integration for the more electric aircraft,* in *5th International Energy Conversion Engineering Conference and Exhibit (IECEC)* (2007).

[6] E. T. James, *Computers take flight: a history of NASA's pioneering digital fly-by-wire project* (NASA Dryden Flight Research Center, 2000).

[7] W. Cao, B. C. Mecrow, G. J. Atkinson, J. W. Bennett, and D. J. Atkinson, *Overview of electric motor technologies used for more electric aircraft (mea),* IEEE Transactions on Industrial Electronics **59**, 3523 (2012).

[8] G. Raimondi, T. Sawata, M. Holme, A. Barton, J. Coles, P. Mellor, and N. Sidell, *Aircraft embedded generation systems,* in *Power Electronics, Machines and Drives* (2002).

[9] E. Ganev, *High-performance electric drives for aerospace more electric architectures part i - electric machines,* in *IEEE Power Engineering Society, IEEE General Meeting* (2007) pp. 1–8.

[10] M. Cronin, A. Hays, F. Green, N. Radovcich, C. Helsley, and W. Rutchik, *Integrated eigital/electric aircraft concepts study,* Tech. Rep. (NASA Contractor Report 3841, 1985).

[11] P. J. Masson and C. A. Luongo, *High power density superconducting motor for all-electric aircraft propulsion,* IEEE Tansactions on Applied Superconductivity **15**, 2226 (2005).

[12] A. Hoffman, I. Hansen, R. Beach, R. Plencher, R. Dengler, K. Jefferies, and R. Frye, *Advanced secondary power system for transport aircraft,* Tech. Rep. (NASA Technical Paper 2463, Washington, DC: NASA, 1985).

[13] Dailymail.co.uk, *Forget electric cars - airbus unveils plans to fly battery-powered planes within the next 20 years,* `http://www.dailymail.co.uk/` (2014).

[14] M. Grady, *Eads unveils electric, aerobatic, four-engine airplane,* `http://www.avweb.com/` (2010).

[15] ENFICA-FC, *Environmentally friendly inter city aircraft powered by fuel cells (enfica-fc),* `http://www.enfica-fc.polito.it/` (2011).

[16] T. E. Noll, J. M. Brown, M. E. Perez-Davis, S. D. Ishmael, G. C. Tiffany, and M. Gaier, *Investigation of the helios prototype aircraft mishap volume I mishap report,* Tech. Rep. (NASA Langley Research Center, Hampton, VA 23681-2199, 2004).

[17] A. S. Gohardani, G. Doulgeris, and R. Singh, *Challenges of future aircraft propulsion: a review of distributed propulsion technology and its potential application for the all electric commercial aircraft,* Progress in Aerospace Sciences **47**, 369 (2011).

[18] V. Dijakovic, *A future of electric airplanes?* `http://www.solarimpulse.com/` (2013).

[19] D. industry daily staff, *ER/MP gray eagle: enhanced MQ-1C predators for the army,* `https://www.defenseindustrydaily.com/warrior-ermp-an-enhanced-predator-for-the-army-03056/` (2015).

[20] E. Products, *Unmanned aerial vehicles block diagram,* `http://www.electronicproducts.com/Electromechanical_Components/Motors_and_Controllers/Unmanned_Aerial_Vehicles_Block_Diagram.aspx` (2015).

[21] R. E. Perez and H. H. Liu, *Multidisciplinary optimization framework for control-configuration integration in aircraft conceptual design,* Journal of Aircraft **43**, 1937 (2006).

[22] J. R. Martins and A. B. Lambe, *Multidisciplinary design optimization: a survey of architectures,* AIAA Journal **51**, 2049 (2013).

[23] Kehrer and WT, *Design evolution of the boeing 2707-300 supersonic transport,* in *43rd AGARD Flight Mechanics Panel Meeting, Aircraft Design Integration and Optimization,* edited by AGARD (1973).

[24] W. Birkenstock, *Unstable aircraft design: the computer at the controls,* Flug Revue **9**, 74 (1999).

[25] T. Dermis, J. Nalepka, D. Thompson, and D. Dawson, *Fly-by-light: the future of flight control technology,* Tech. Rep. (Technology Horizons Magazine, Air Force Research Laboratory Air Vehicles Directorate, 2005).

[26] Anonymous, *Current state of the art on multidisciplinary design optimization (MDO)* (American Institute of Aeronautics and Astronautics, 1991).

[27] A. Rosenblatt and G. F. Watson, *Concurrent engineering,* IEEE SPECTRUM **5**, 22 (1991).

[28] H. Takeuchi and I. Nonaka, *The new product developement game,* Harvard Business Review **64**, 137 (1986).

[29] A. Kusiak, *Concurrent engineering automation, tools and techniques* (John Wiley & Sons, Inc., 1993).

[30] P. Mawhinney, M. Price, R. Curran, E. Benard, A. Murphy, and S. Raghunathan, *Geometry-based approach to analysis integration for aircraft conceptual design,* in *AIAA 5th Aviation, Technology, Integration, and Operations Conference (ATIO)* (2005) pp. 1–9.

[31] J.Weber, *Automotive development processes* (Springer-Verlag Berlin Heidelberg, Berlin, 2009).

[32] NHTSA, *Flat file copies of NHTSA/ODI database,* `http://www-odi.nhtsa.dot.gov/downloads/` (2011).

[33] F. Tian and M. Voskuijl, *Knowledge based engineering to support automotive conceptual design and automatic control software development,* in *Proceedings of the FISITA 2012 World Automotive Congress Lecture Notes in Electrical Engineering*, 194, Vol. 6, edited by SAE-China and FISITA (Springer Berlin Heidelberg, 2012) pp. 393–405.

[34] J. A. Weimer, *The role of electric machines and drives in the more electric aircraft,* in *IEEE International Electric Machines & Drives Conference (IEMDC)*, Vol. 1 (2003) pp. 11–15.

[35] E. aviation safety agency, *Certification specifications for large aeroplanes CS-25,* European aviation safety agency (2007).

[36] T. Yomchinda and J. F. Horn, *Handling qualities assessment of a model inversion controller for a tiltrotor aircraft,* in *The 3rd International Basic Research Conference on Rotorcraft Technology* (2009) pp. 193–206.

[37] H. Weule, D. Spath, and H.-J. Schelberg, *Object-oriented programming of plc based on iec 1131,* Production Engineering **1**, 197 (1994).

[38] D. Spath and R. Landwehr, *Three-dimensional programming and simulation of PLC-controlled manufacturing systems,* International Journal for Manufacturing Science and Production **4**, 189 (2001).

[39] N. Kyura and H. Oho, *Mechatronics-an industrial perspective,* IEEE/ASME Transactions on Mechatronics **1**, 10 (1996).

[40] Alciatore and D. G., *Introduction to mechatronics and measurement systems* (McGraw-Hill, 2007).

[41] W. Bolton, *Electrical control systems in mechanical and electrical engineering* (Addison-Wesley Longman, 1999).

[42] M. M. da Silva, O. Bruls, J. Swevers, W. Desmet, and H. V. Brussel, *Computer-aided integrated design for machines with varying dynamics,* Mechanism and Machine Theory **44**, 1733 (2009).

[43] S. Behbahani and C. W. de Silva, *Mechatronic design quotient as the basis of a new multicriteria mechatronic design methodology,* IEEE/ASME Transactions on Mechatronics **12**, 227 (2006).

[44] J. J. Granda, *The role of bond graph modeling and simulation in mechatronics systems,* Mechatronics **12**, 1271 (2002).

[45] J. van Amerongen, *Mechatronic design,* Mechatronics **13**, 1045 (2003).

[46] A. A. A. Cabrera, M. S. Erden, M. J. Foeken, and T. Tomiyama, *High level model integration for design of mechatronic systems,* in *2008 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (IEEE/ASME MESA 2008)* (IEEE Intelligent Transportation Systems Society, Beijing Friendship Hotel, Beijing, China, 2008) pp. 12–15.

[47] A. A. Cabrera, M. Foeken, O. Tekin, K. Woestenenk, M. Erden, B. D. Schutter, M. van Tooren, R. Babuska, F. van Houten, and T. Tomiyama, *Towards automation of control software: a review of challenges in mechatronic design,* Mechatronics **20**, 876 (2010).

[48] A. A. Cabrera, K. Woestenenk, and T. Tomiyama, *An architecture model to support cooperative design for mechatronic products: a control design case,* Mechatronics **21**, 534 (2011).

[49] A. A. Cabrera, T. van Beek, H. Komoto, and T. Tomiyama, *Architecture-centric design approach for platform development,* (Springer, New York, 2014) Chap. 17, pp. 419–447.

[50] P. Bhattacharya, N.-A. S. Welakwe, and R. Makanaboyina, *Integration of CATIA with modelica,* in *Proceedings of the 5th International Modelica Conference* (2006) pp. 671–675.

[51] R. Sinha, C. J. Paredis, and P. K. Khosla, *Integration of mechanical CAD and behavioral modeling,* in *Proceedings of the IEEE/ACM Workshop on Behavioral Modeling and Simulation* (2000) pp. 31–36.

[52] A. Biahmou, A. Frohlich, and J. Stjepandic, *Improving interoperability in mechatronic product developement,* in *Proceedings of the International Conference on Product Lifecycle Management* (2010) pp. 510–521.

[53] G. L. Rocca, *Knowledge based engineering: between AI and CAD. review of a language based technology to support engineering design,* Advanced Engineering Informatics **26**, 159 (2012).

[54] K. Kural, M. Voskuijl, T. Fengnian, and J. Pauwelussen, *Determination of representative loading conditions for effective semitrailer design,* Transport **29**, 363 (2014).

[55] P. Sainter, K. Oldham, and A. Larkin, *Achieving benefits from knowledge-basedengineering systems in the longer term as wellas in the short term,* in *Proceedings International Conference on Concurrent Enterprising* (2000).

[56] G. Blount, S. Kneebone, and M. Kingston, *Selection of knowledge-based engineering design applications,* Journal of Engineering Design **6**, 31 (1995).

[57] C. Chapman and M. Pinfold, *The application of knowledge based engineering approach to the rapid design and analysis of an automotive structure,* Journal of Advances in Engineering Software **32**, 903 (2001).

[58] W. J. Verhagen, P. Bermell-Garcia, R. E. van Dijk, and R. Curran, *A critical review of knowledge-based engineering: An identification of research challenges,* Advanced Engineering Informatics **26**, 5 (2012).

[59] G. L. Rocca, *Knowledge based engineering techniques to support aircraft design and optimization*, Ph.D. thesis, Department of Aerospace Design, Integration & Operations, Faculty of Aerospace Engineering, Delft University of Technology (2011).

[60] C. A. Coello Coello, *Evolutionary algorithms for solving multi-objective problems* (Kluwer Academic/Plenum, 2002).

[61] A. A. Hopgood, *Intelligent systems for engineers and scientists*, 2nd ed. (CRC Press, 2000).

[62] K. Amadori, M. Tarkian, J. Olvander, and P. Krus, *Flexible and robust CAD models for design automation,* Advanced Engineering Informatics **26**, 180 (2012).

[63] C. Ledermann, P. Ermanni, and R. Kelm, *Dynamic CAD objects for structural optimization in preliminary aircraft design,* Aerospace Science and Technology **10**, 601 (2006).

[64] K. Tan, E. Khor, and T. Lee, *Multiobjective evolutionary algorithms and applications*, edited by X. Wu and L. Jain (Sp, 2005).

[65] M. Gen and R. Cheng, *Genetic algorithms and engineering design* (John Wiley & Sons, Inc., 1996).

[66] T. Back, *Evolutionary algorithms in theory and practice* (Oxford University Press, 1996).

[67] M. Gen and R. Cheng, *A survey of penalty techniques in genetic algorithms,* in *Proceedings of IEEE International Conference on Evolutionary Computation* (1996) pp. 804–809.

[68] Z. Michalewicz, *A survey of constraint handling techniques in evolutionary computation methods,* in *Proceedings of the Fourth Annual Conference On Evolutionary Programming,* edited by J. R. McDonnell, R. G. Reynolds, and D. B. Fogel (McDonnell, 1995) pp. 135–155.

[69] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning* (Addison-Wesley Professional, 1989).

[70] L. Krakers, M. V. Tooren, A. Beukers, and P. L. G. La Rocca, *A design & engineering engine to investigate acoustics in preliminary fuselage design,* in *9th AIAA/CEAS Aeroacoustics Conference and Exhibit* (Hilton Head, South Carolina, 2003) pp. 1–9.

[71] G. L. Rocca, L. Krakers, and M. van Tooren, *Development of an ICAD generative model for blended wing body,* in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* (2002).

[72] E. Schut, M. van Tooren, and J. Berends, *Feasilization of a structural wing design problem,* in *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials* (2008).

[73] A. van der Laan and M. V. Tooren, *Parametric modeling of movables for structural analysis,* in *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference* (2004).

[74] H. A. Baluch, M. van Tooren, and E. Schut, *Design tradeoffs for fiber composite fuselages under dynamic loads using structural optimization,* in *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials* (2008).

[75] GENWORKS, *Genworks international,* `http://www.genworks.com` (2013).

[76] K. S. N. Ripon, S. Kwong, and K. Man, *A real-coding jumping gene genetic algorithm (RJGGA) for multiobjective optimization,* Information Sciences **177**, 632 (2007).

[77] I. G. Damousis, A. G. Bakirtzis, and P. S. Dokopoulos, *A solution to the unit-commitment problem using integer-coded genetic algorithm,* IEEE Transactions on Power Systems **19**, 1165 (2004).

[78] A. Rogers and A. Prugel-Bennett, *Genetic drift in genetic algorithm selection schemes,* IEEE Transactions on Evolutionary Computation **3**, 298 (1999).

[79] J. D. Schaffer, *Multiple objective optimization with vector evaluated genetic algorithms,* in *Proceedings of the 1st International Conference on Genetic Algorithms* (1985) pp. 93–100.

[80] H. Ishibuchi and T. Murata, *Multi-objective genetic local search algorithm,* in *Proceedings of IEEE International Conference on Evolutionary Computation* (1996) pp. 119–124.

[81] M. Gen and R. Cheng, *Genetic algorithms and engineering optimization,* (John Wiley Sons Inc., 2000).

[82] G. Syswerda, *Handbook of genetic algorithms,* (Van Nostrand Reinhold, 1991) Chap. Schedule optimization using genetic algorithms.

[83] D. Thierens, *Adaptive mutation rate control schemes in genetic algorithms,* in *Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 1 (2002) pp. 980–985.

[84] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence* (Wiley-IEEE Press, 2005).

[85] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems* (Addison-Wesley Longman Publishing Co., Inc., 2001).

[86] J. T. Richardson, M. R. Palmer, G. E. Liepins, and M. Hilliard, *Some guidelines for genetic algorithms with penalty functions,* in *Proceedings of the third international conference on Genetic algorithms* (1989) pp. 191–197.

[87] Z. Michalewicz and M. Schoenauery, *Evolutionary algorithms for constrained parameter optimization problems,* Evolutionary Computation **1**, 1 (1996).

[88] D. Orvosh and L. Davis, *Using a genetic algorithm to optimize problems with feasibility constraints.* in *IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on Evolutionary Computation*, Vol. 2 (1994) pp. 548–553.

[89] J. E. Baker, *Adaptive selection methods for genetic algorithms,* in *Proceedings of the 1st International Conference on Genetic Algorithms*, Vol. 1 (1985) pp. 101–111.

[90] J. A. Snyman, *Practical mathematical optimization an introduction to basic optimization theory and classical and new gradient-based algorithms* (Springer US, 2005).

[91] S. J. Louis and G. J. E. Rawlins, *Foundations of genetic algorithms 2,* (Morgan Kaufmann, 1992) Chap. Predicting convergence time for genetic algorithms, pp. 141–161.

[92] F. Tian and M. Voskuijl, *Knowledge based engineering to support electric and electronic system design and automatic control software development,*

in *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC),* Vol. 7A4 (IEEE, East Syracuse, NY, 2013) pp. 1–9.

[93] C. Ungil, *Matlab interface / mat-file reader,* `http://www.ungil.com/lisp/matlab.html/` (2013).

[94] *Industrial automation systems and integration - product data representation and exchange - part 21: implementation methods: clear text encoding of the exchange structure,* (2010).

[95] R. W. Prouty, *Helicopter performance, stability, and control* (Krieger Pub Co, 1986).

[96] Y. Liu, *Automotive design* (Tsinghua University Press, Beijing, 2001).

[97] S. Dominka, E. Broecker, and F. Schiller, *Automated execution of simulation studies demonstrated via a simulation of a car,* in *Proceedings of the 2008 Winter Simulation Conference* (2008) pp. 2916–2924.

[98] Y. Huang, M. D. Seck, and A. Verbraeck, *From data to simulation models: component-based model generation with a data-driven approach,* in *Proceedings of the 2011 Winter Simulation Conference* (2011) pp. 3719–3729.

[99] J. Liscouet, M. Budinger, J.-C. Mare, and S. Orieux, *Modelling approach for the simulation-based preliminary design of power transmissions,* Mechanism and Machine Theory **46**, 276 (2011).

[100] Y. He and J. McPhee, *A design methodology for mechatronic vehicles: application of multidisciplinary optimization, multibody dynamics and genetic algorithms,* Vehicle System Dynamics **43**, 697 (2005).

[101] A. C. Pil and H. H. Asada, *Integrated structure/control design of mechatronic systems using a recursive experimental optimization method,* IEEE/ASME Transactions on Mechatronic **1**, 191 (1996).

[102] H. M. Paynter, *Analysis and design of engineering systems* (MIT Press, Cambridge, MA, USA, 1961).

[103] W. Borutzky, *Bond graph modelling and simulation of multidisciplinary systems - an introduction,* Simulation Modelling Practice and Theory **17**, 3 (2009).

[104] D. Margolis and T. Shim, *A bond graph model incorporating sensors, actuators, and vehicle dynamics for developing controllers for vehicle safety,* Journal of the Franklin Institute **338**, 21 (2001).

[105] J. Fisher, *Model-based systems engineering: a new paradigm,* INCOSE Insight **1**, 3 (1998).

[106] C. J. Paredis, H. B. Brown, and P. K. Khosla, *A rapidly deployable manipulator system,* Robotics and Autonomous Systems **21**, 289 (1997).

[107] G. Ferretti, G. Magnani, and P. Rocco, *Virtual prototyping of mechatronic systems,* Annual Reviews in Control **28**, 193 (2004).

[108] P. Piela, T. Epperly, K. Westerberg, and A. Westerberg, *ASCEND: an object-oriented computer environment for modeling and analysis: the modeling language,* Computers & Chemical Engineering **15**, 53 (1991).

[109] S. E. Mattsson and H. Elmqvist, *An overview of the modeling language Modelica,* in *Eurosim' 98 Simulation Congress* (1998) pp. 14–15.

[110] Dynasim, *Dymola dynamic modeling laboratory user's manual,* Dynasim AB, Research Park Ideon, SE-223 70 Lund, Sweden, 5th ed. (2004).

[111] P. Fritzson, P. Aronsson, P. Bunus, V. Engelson, L. Saldamli, H. Johansson, and A. Karstrom, *The open source Modelica project,* in *Proceedings of the 2nd International Modelica Conference* (Deutsches Zentrum fur Luft- und Raumfahrt e.V. (DLR), Oberpfaffenhofen, Germany, 2002) pp. 18–19.

[112] MathWorks, *Simscape model and simulate multidomain physical systems,* `http://nl.mathworks.com/products/simscape/` (2014).

[113] G. Rocca and M. Tooren, *Enabling distributed multi-disciplinary design of complex products: a knowledge based engineering approach,* Design Research **5**, 333 (2007).

[114] A. Berard and A.Rizzi, *CADac: a new geometry construction tool for aerospace vehicle pre-design and conceptual design,* in *26th AIAA Applied Aerodynamics Conference* (2008) pp. 1–10.

[115] A. Rizzi, P. Eliasson, T. Goetzendorf-Grabowski, M. Zhang, T. S. Richardson, and J. B. Vos, *Design of a canard configured transcruiser using CEASIOM,* Progress in Aerospace Sciences **47**, 695 (2011).

[116] B. Mialon, A. Khrabrov, S. B. Khelil, A. Huebner, A. D. Ronch, K. Badcock, L. Cavagna, P. Eliasson, M. Zhang, J.-C. Jouhaud, G. Roge, S. Hitzel, S. Ricci, and M. Lahuta, *Validation of numerical prediction of dynamic derivatives: the DLR-F12 and the transcruiser test cases,* Progress in Aerospace Sciences **47**, 674 (2011).

[117] M. Barth and A. Fay, *Automated generation of simulation models for control code tests,* Control Engineering Practice **21**, 218 (2013).

[118] M. Voskuijl, J. de Klerk, and D. van Ginneken, *Flight mechanics modeling of the prandtl plane for conceptual and preliminary design,* in *Variational Analysis and Aerospace Engineering: Mathematical Challenges for Aerospace Design*, edited by G. Buttazzo and A. Frediani (Springer, London, 2012) pp. 435–462.

[119] N. A. Sahani and J. F. Horn, *Adaptive model inversion control of a helicopter with structural load limiting,* Journal of Guidance, Control, and Dynamics **29**, 411 (2006).

[120] D. Patt, L. Liu, and P. P. Friedmann, *Simultaneous vibration and noise reduction in rotorcraft using aeroelastic simulation, American Helicopter Society 60th Annual Forum,* American Helicopter Society 60th Annual Forum **51**, 127 (2006).

[121] SMLib, *SMLIB a function stock library for sourcemod,* `https://www.sourcemodplugins.org/smlib/` (2015).

[122] S. Devasia, D. Chen, and B. Paden, *Nonlinear inversion-based output tracking,* IEEE Transactions on Automatic Control **41**, 930 (1996).

[123] G. Looye and H.-D. Joos, *Design of robust dynamic inversion control laws using multi-objective optimization,* in *AIAA Guidance, Navigation, and Control Conference and Exhibit* (2001).

[124] M. D. Tandale and J. Valasek, *Fault-tolerant structured adaptive model inversion control,* Journal of Guidance,Control, and Dynamics **29**, 635 (2006).

[125] J.-H. Ryu, C.-S. Park, M.-J. Tahk, J.-H. Ryu, C.-S. Park, and M.-J. Tahk, *Plant inversion control of tail-controlled missiles,* in *AIAA Guidance, Navigation, and Control Conference* (1997) pp. 1691–1696.

[126] J. Kim and J. Jang, *Nonlinear model inversion control for bank-to-turn missile,* in *AIAA Guidance, Navigation, and Control Conference* (1995) pp. 1308–1315.

[127] R. T. Rysdyk and A. J. Calise, *Adaptive model inversion flight control for tilt-rotor aircraft,* Journal of Guidance, Control, and Dynamics **22**, 402 (1999).

[128] A. Rahideh, H. MShaheed, and A. H. Bajodah, *Adaptive non-linear model inversion control of a twin rotor multi-input multi-output system using aircraft intelligence,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **22**, 343 (2007).

[129] J. A. Petersen and M. Bodson, *Interior-point algorithms for control allocation,* Journal of Guidance,Control, and Dynamics **28**, 471 (2005).

[130] K. Bordignon, *Constrained control allocation for systems with redundant control effectors*, Ph.D. thesis, Virginia Polytechnic Institute and State University (1996).

[131] G. Roberts and R. Sutton, *Advances in unmanned marine vehicles* (The Institution of Engineering and Technology, London, 2006).

[132] M. Bodson, *Evaluation of optimization methods for control allocation,* Journal of Guidance, Control, and Dynamics **25**, 703 (2002).

[133] S. Bouabdallah and R. Siegwart, *Full control of a quadrotor,* in *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (San Diego, CA, USA, 2007) pp. 153–158.

[134] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation* (CRC Press, 1994).

[135] B. Surgenor and V. N.D, *Continuous sliding mode control of a pneumatic actuator,* Journal of Dynamic Systems, Measurement, and Control **119**, 578 (1997).

[136] Z. Zhu, *The display method and experimental research of assistant manual control*, Master's thesis, School of Automation Science and Electrical Engineering, Beihang University (2008).

[137] Anon, *Aeronautical design standard performance specification, handling qualities requirements for military rotorcraft,* (2000).

[138] H. Ebbinghaus, *Curve of forgetting,* `http://uwaterloo.ca/counselling-services/curve-forgetting` (2014).

[139] F. Budinsky, M. Finnie, J. Vlissides, and P. Yu, *Automatic code generation from design patterns,* IBM System Journal **35**, 151 (1996).

[140] D. Mery and N. K. Singh, *Automatic code generation from event-b models,* in *2nd International Symposium on Information and Communication* (2011) pp. 179–188.

[141] O. Pastor, J. Gomez, E. Insfr, and V. Pelechano, *The OO-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming,* Information Systems **26**, 507 (2001).

[142] M. M. R. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri, *A framework for modeling, simulation and automatic code generation of sensor network applications,* in *IEEE SECON 2008* (2008) pp. 515–522.

[143] A. Javed, P. Strooper, and G. Watson, *Automated generation of test cases using model-driven architecture,* in *Second International Workshop on Automation of Software Test* (2007).

[144] W. H. Savage, W. F-Cowan, R. J. Geiger, and G. D. Leman, *Automated software code references generation from a metadata-based repository,* (2003).

[145] D. P. Henninger, R. H. Jensen, and C. T. Keene, *Method and apparatus for automatic generation of object oriented code for mapping relational data to objects,* (1996).

[146] K. S. Perycz, A. Golichowski, B. T. Iwanojko, A. Kaminski, J. Kogut, and M. O. Przekop, *Method and system for software modularization and automatic code generation for embedded systems,* (2006).

[147] W. Sadlq, *System and method for automated code generation using language neutral software code,* (2007).

[148] J.Schaufele and T.Zurawka, *Automotive software engineering* (SAE International, Warrendale, 2005).

[149] A. Hunt and D. Thomas, *The pragmatic programmer: from journeyman to master* (Addison-Wesley Professional, 1999).

[150] U.Kiencke and L.Nielsen, *Automotive control systems* (Springer-Verlag Berlin Heidelberg, Berlin, 2005).

[151] J. Larminie and J. Lowry, *Electric vehicle technology explained* (John Wiley Sons Inc., 2003).

[152] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, *Precision flight control for a multi-vehicle quadrotor helicopter testbed,* Control Engineering Practice **19**, 1023 (2011).

[153] ANSYS, *Multiphysics solution simulation for the real world,* `http://www.ansys.com/Products/Simulation+Technology/Multiphysics` (2015).

[154] COMSOL, *COMSOL multiphysics the platform for physics-based modeling and simulation,* `http://www.cn.comsol.com/comsol-multiphysics` (2015).

[155] ADINA, *Multiphysics capabilities of ADINA,* `http://www.adina.com/multiphysics.shtml` (2015).

[156] FEATool, *FEATool - finite element analysis made easy,* `http://www.precisesimulation.com/featool/` (2015).

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerest gratitude to my supervisor Dr. Mark Voskuijl. He has a clear and deep understanding of flight dynamics and control system development in the context of aircraft design. I could always get useful information from him when I encountered difficulties in my research. Besides providing academic guidance, he is also a very approachable person. I respect him as a teacher and friend. It is hard to image how to complete this dissertation and the associated scientific articles without his guidance.

I am most grateful to my promoter Prof. dr. Leo Veldhuis. Instead of focusing on specific technical questions, he suggested me to form a systematic description of the research questions, objectives, and finally the novelty and significance of the work. Based on his comments I had many fresh new ideas. It also allowed me to think about the relevance of my research and whether it is understandable for people who are not experts in the field or even without an academic background. Finally, he helped me to improve my technical writing skills needed for the writing of this dissertation.

My gratitude also goes to Dr. Gianfranco La Rocca who offered me a postdoctoral position for one year. I really enjoyed the project and wish to continue working on this topic in future.

I would like to thank Prof. dr. Michael Van Tooren and Prof. dr. Tetsuo Tomiyama for their insightful comments and encouragement. Based on their suggestions, I rephrased the title and many descriptions in the first several chapters, resulting in more appropriate statements.

My sincere thanks also goes to Prof. dr. Zhixiong Lu. He is very strict on research but kind to students. He provides funding for his students to attend annual automotive exhibitions in China. He also recommended me to study abroad and provided a lot of help and support during my masters studies.

Special thanks go to my colleagues, Zaoxu Zhu, Peijun Wu, Li Mo, Feijia Yin and Peijian Lv. We created many happy moments to have a colorful and wonderful PhD life colorful. I am also grateful to my roommate, Lei Cheng, who is an intellectual and quiet girl. I could not have overcome all the difficulties without the encouragement from all of you.

I save the words of deepest gratitude to my parents, Zengshun Tian and Cuiping Cui. I love them more than anybody else in this world. Although they are only regular workers in a state-owned iron and steel enterprise, my parents offer me a sweet home as best as then can. I learned to be strong, courageous and responsible from my father and kind to people from my mother. Every time when I turn my back towards them and start my trip to somewhere far away from home, I cannot stop the falling tears and start to missing them. I will cherish every moment with them for the rest of my life.

CURRICULUM VITAE

Fengnian TIAN

17-10-1985	Born in Nei Mongol, China.

EDUCATION

SEPT. 2015	Ph.D. **Delft University of Technology**, The Netherlands Title: *"An Integrated Knowledge Based Engineering Mechatronics Modeling Approach to Support the Design of Unstable and Unmanned Aircraft"* Promoter: Prof.dr.ir. L.L.M. Veldhuis
DEC. 2009	Master's degree in Vehicle Engineering **Nanjing Agricultural University**, China Title: *"Research and Realization of Control Algorithm for Electronic Hydraulic Power Steering System"* Supervisor: Prof.dr.ir. Z.X. Lu
JUNE 2007	Bachelor's degree in Mechanical Engineering **Xi'an Technological University**, China Specialization in Machinery Design, Manufacturing and Automation

AWARDS

Dec. 2012	Lisp in Summer Projects, USA **Certificate of Awesome**
July 2006	2th Mechanical Innovation Design Competition for National College Students, China(North West Area in China) **Silver Medal**

List of Publications

Journals

[1] Fengnian Tian and Mark Voskuijl. Mechatronic design and optimization using knowledge based engineering applied to an inherently unstable and unmanned aerial vehicle. *IEEE/ASME Transactions on Mechatronics*, 2015. DOI: 10.1109/TMECH.2015.2441832.

[2] Fengnian Tian and Mark Voskuijl. Automated generation of multiphysics simulation models to support multidisciplinary design optimization. *Advanced Engineering Informatics*, 2015. DOI: 10.1016/j.aei.2015.07.004.

[3] Karel Kural, Mark Voskuijl, Tian Fengnian, and Joop Pauwelussen. Determination of representative loading conditions for effective semitrailer design. *Transport*, 29(4):363–375, 2014. DOI: 10.3846/16484142.2014.982174.

Conference Proceedings

[1] Fengnian Tian and Mark Voskuijl. Knowledge based engineering to support automotive conceptual design and automatic control software development. In *Proceedings of the FISITA 2012 World Automotive Congress Lecture Notes in Electrical Engineering*, volume 194, pages 393–405, 2012. DOI: 10.1109/DASC.2013.6712633.

[2] Fengnian Tian and Mark Voskuijl. Knowledge based engineering to support electric and electronic system design and automatic control software development. In *32nd Digital Avionics Systems Conference*, volume 7A4, pages 1–9, October 2013. DOI: 10.1007/978-3-642-33829-8_37.

ISBN/EAN: 978-94-6186-528-1

Cover design by Chunzhe LI