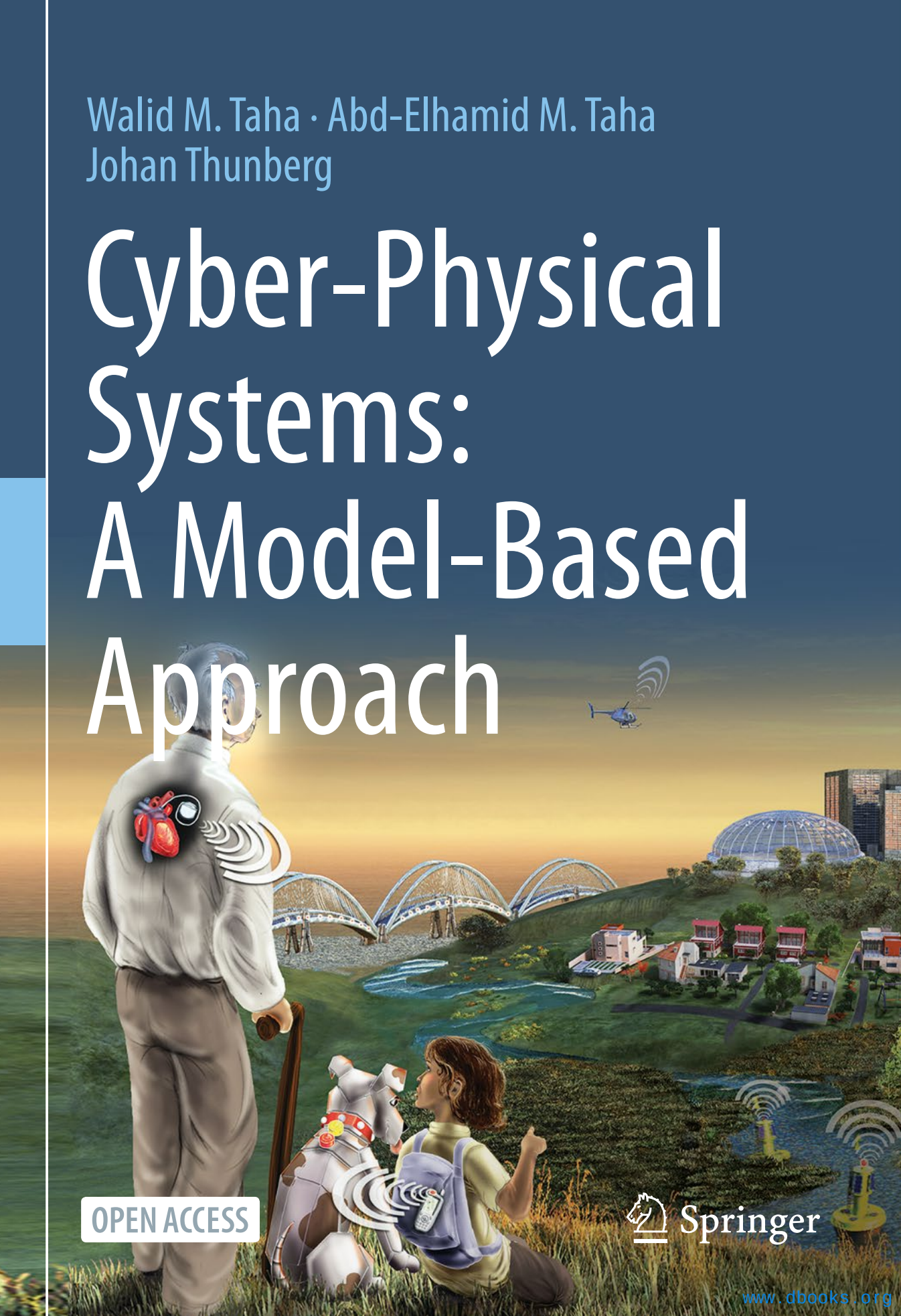
Walid M. Taha · Abd-Elhamid M. Taha
Johan Thunberg

Cyber-Physical Systems: A Model-Based Approach



OPEN ACCESS

Springer

www.dbooks.org

Cyber-Physical Systems: A Model-Based Approach

Walid M. Taha • Abd-Elhamid M. Taha
Johan Thunberg

Cyber-Physical Systems: A Model-Based Approach

Springer

www.dbooks.org

Walid M. Taha
School of Information Technology
Halmstad University
Halmstad
Hallands Län, Sweden

Abd-Elhamid M. Taha
Alfaisal University
Riyadh, Saudi Arabia

Johan Thunberg
Halmstad University
Halmstad
Hallands Län, Sweden

ISBN 978-3-030-36070-2 ISBN 978-3-030-36071-9 (eBook)
https://doi.org/10.1007/978-3-030-36071-9

© The Editor(s) (if applicable) and The Author(s) 2021. This book is an open access publication.
Open Access This book is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.
The images or other third party material in this book are included in the book's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the book's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Cover Image Credit: © N.R. Fuller, National Science Foundation

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To the future

www.dbooks.org

Preface

This book is aimed at anyone who would like to be an inventor or an innovator of cyber-physical systems, that is, machines, manufactured products, methods, or combinations of such things. The goal of such invention or innovation is to fill concrete needs in society. Modern examples of such creations include autonomous and electric vehicles, quadcopters, smartphones, and robots. More than ever before, it is useful to describe such systems as *cyber-physical* because they combine aspects that have previously been confined within the realm of relatively separate disciplines, and in particular, within computer science and within physical sciences. We believe that, increasingly, a more holistic and unified view of such systems is needed. This book achieves this goal by taking a model-based approach, which allows the reader to develop her mathematical modeling skills in a manner that allows her to combine her knowledge from a diverse range of technical disciplines into the same model. At the same time, we are guided by the fact that different types of activities go into the process of creating and developing such systems, including three that this book is concerned with: learning, teamwork, and design. We start by considering these three points in more detail.

Learning

Learning is a lifelong activity that has become more important than ever in today's knowledge economy. Today, there is much more to learn. The very computing and communication technologies that inspire this book are themselves accelerating science and technology, and the pace with which we accumulate new knowledge. To manage this changing learning landscape, you need to work smart. This book aims to help you do that by starting from a high-level, systems perspective and working down into key, representative subtopics. This way you are able to focus on key concepts that are broadly applicable to almost all modern inventions. An important tool that this book provides to help you with learning in the long term is

the model-based approach, and especially the hybrid-systems modeling formalism, which is introduced early in the book and used throughout to provide practice.

This book is also intended to encourage you to ask questions. To set an example, the book itself motivates many topics and ideas by asking questions. Asking questions is one of the most important skills for lifelong learning—it is an activity that involves taking stock of what we know and identifying the gaps that we need to fill to achieve a certain goal. The book is not about what you *should* do with these concepts, but rather, what you *could* do. It is up to you and your understanding of the world and its needs to put these concepts to work. We challenge *you* to show us how new systems can be designed.

Teamwork

Teamwork and collaboration are essential because many modern inventions require bringing together ideas from a wide range of disciplines, and to bring in knowledge from a vast number of areas. As a skill, teamwork is at the top of what employers tell us they need. Of course, teamwork and collaboration can be a tricky subject to learn about from a book—we only experience collaboration when we work with others. This book will help you do that in three ways. First, it introduces key concepts from several areas that are often implicated in new inventions, including those that are related to computing and some related to more physical science and engineering disciplines. To talk about these concepts, some basic vocabulary is introduced. Both the concepts and the vocabulary help us appreciate the type of expertise that experts in different subjects have, and to build the beginnings of the vocabulary needed to work with these experts.

Second, the book gently introduces the reader to three powerful mathematical notions that formalize the similarities between a wide range of seemingly different concepts. Seeing the similarities allows us to compress our knowledge and to be able to think quickly of alternatives. The focus is less on the mathematical origin and nature of these concepts and more on how to use them in modeling and simulation, which is where their effects are more animated, and where you will need them the most. These concepts can be seen as an example of a core mathematical formalism that give us tremendous expressivity to express ideas and to animate them through simulations. These concepts are:

1. **Conditional laws**: These are laws that only apply under certain conditions. Examples are laws such as “the temperature is higher than 25°” and “the car started moving.”
2. **Discrete change laws**: These are laws that specify a discrete change in one or more quantities. Examples are laws such as “set the direction to 45° clockwise” or “turn the tank valve ON.” Note here that the quantities do not need to be discrete, but rather, it is the change that happens in a discrete instance in time.

www.dbooks.org

3. **Rate change laws**: These are laws that relate to the rate of change of one or more quantities. Examples are laws such as "the speed of the vehicle is 25 miles per hour" or "the rate of change of speed is equal to -9.8 meters per second squared."

These concepts will be used to model all the different types of phenomena covered in the book. The point of this minimalism is to make it easier for you to see the similarity between concepts that arise in different settings. It also means that you only need to learn a small number of concepts and you will have a lot of opportunities to practice them well. When you have mastered them and their use in modeling, they will become a powerful communication tool for you both when collaborating with others and when you are working on inventions of your own. Incidentally, if you are surprised that we will be doing so much with these three seemingly simple concepts, you are in good company. They are only intuitive and simple individually. When combined, their expressive power and usefulness multiply.

Third, the first part of the book includes study problems, labs, and a project that are intended to be carried out in collaboration with at least one partner. Study problems help you check your understanding of the material covered in the chapter, labs offer more hands on experience, and the project helps you connect the learnings from different chapters together. In a classroom setting, the instructor will typically organize project teams. If you are reading this book independently, we encourage you to work on the labs, study problems, and project with a partner.

Design

Design is a process that involves creativity and experimentation, two things which are closely intertwined—trying out things is an essential part of a process of developing something new and establishing that it achieves the intended purpose. To help you experiment and explore ideas efficiently, this book makes use of a modeling and simulation environment called Acumen. Acumen was built specifically around the three key mathematical notions mentioned above, and to make it easier for both learners and researchers to study models that use these concepts. It has several features that can support you in developing your core creativity skills:

- **Specificity**: Acumen is a modeling language and environment aimed at making it easier to experiment and learn about the three key mathematical notions and the type of systems they can model. It is not, for example, a programming or a scripting language. It is not a real-time control language. The current implementation is also not built for efficiency. Rather, the focus is on providing an integrated development environment and on making sure that simulations provide results that have a simple explanation and a simple relation to the mathematical meaning of the three basic concepts.
- **Time**: In Acumen, all variables are functions of time. This allows us to reflect this fundamental aspect of the world we live in: its dynamism. This notion is central to two of the three concepts mentioned above. Interestingly, the two notions are often

seen to imply different notions of time, one discrete and the other continuous. Acumen has a notion of time that unifies both.

- **Visualization**: In Acumen, all variables are automatically plotted with respect to time. This helps you visualize the dynamic nature of all variables. Plots have no axis grids to encourage you to focus on the qualitative aspects first. If you are interested in more information, you can point to the curve at a certain time and the system will display more detailed information. Finally, Acumen provides a mechanism to simplify the display of 3D scenes and animation to help you visualize the behavior of more complex systems.
- **Openness**: All models and sub-models used in Acumen are expressed in the source language, and can therefore be easily inspected by the reader. The tool itself is open source and freely available online at http://acumen-language.org and the manual is contained as an appendix to this book. In this book, it is used to provide study problems that involve continuous dynamics, hybrid systems, automata, and discretized and quantized systems.

Modeling is a critical skill for the upcoming generations of inventors, engineers, and scientists, as it allows us to clearly formalize, reflect on, and communicate ideas. For this reason, modeling is a central theme and tool in this book.

Using This book

This book is designed as the textbook for an introductory graduate or undergraduate course to the subject. It can also be used independently. When used for a course, it is typical that Part I is covered sequentially and then one or two topics from Part II are covered. Part I (Core Concepts) contains the material expected to be covered by most university-level courses on the subject and is intended to be covered in sequence. Part II (Selected Topics) contains additional topics that have limited interdependence among them. Thus, a course can include any one or more of them once the materials in Part I (at least up to Chapter 5) have been covered.

When this book is used as the textbook material for a course that takes up 50% of a student's work week, one chapter can be covered each week. This is the case, for example, in the Swedish higher education quarter system, where students take two courses per quarter. In a US quarter system where one course is roughly a 30% of a student's work week, or in a UK semester system where it is closer to 20% of a student's work week, the pace should be adjusted accordingly.

At the end of each chapter in Part I, there are three different types of tasks; they are referred to as Study Problems, Labs, and Projects and are numbered in accordance with the chapters. Study problems are suitable as classwork or home assignments done by students individually or in groups and are submitted to the teacher weekly. These are, in comparison with the labs and projects, of a more theoretical nature and train the students to understand the boundaries of what simulation can deliver. Furthermore, the theory helps us appreciate how simulations can answer questions not possible to answer by mathematical derivations.

www.dbooks.org

The labs are tasks done in Acumen; typically in a classroom or in lab environment under the supervision of a lab assistant. There is no recommendation to assess (grade) the labs. They can be seen as an opportunity for students to become familiar with Acumen and the specific concepts discussed in the chapter that the lab illustrates.

Each chapter's project is a sub-project of a larger project that runs as a common thread throughout the Part I. The end goal is to develop a model of a table tennis (ping pong) playing robot. To succeed, we break down the task into sub-goals manifested through project's activities. The final model is based on incremental additions made in the respective chapter's project activity. Then the content of those ranges from modeling of mechanical and physical phenomena, through hybrid phenomena such as bouncing balls, modeling of sensors and actuators subject to discretization and quantization, to control theoretic aspects such as the control of a robotic arm and coordinate representations thereof. As a proposal for a final project, the different models for the ping pong playing robots can compete against each other one by one in a round-robin format in the simulation environment Acumen.

The three different tasks, namely, study problems, labs and projects, work in symbiosis where study problems provide a theoretical understanding of the material, the project provides a practical understanding of modeling and simulation, and the labs act as a bridge between these two parts.

Summary of Content

Chapter 1 introduces the field of Cyber-Physical Systems (CPS), places it in the broad context, and explains the importance of this interdisciplinary subject in today's connected society. Chapter 2 addresses modeling of mechanical and electric systems. This includes conservation laws, statics, and dynamics. We model such systems mathematically using linear equations and Ordinary Differential Equations. The physical models addressed capture phenomena that evolve in continuous space and time. In Chapter 3, the key question raised is how we model phenomena that may contain not only continuous part but also discrete ones. The answer we provide is to model these phenomena as hybrid systems. We illustrate this by modeling a bouncing ball as a hybrid automaton and furthermore provide examples of physical systems with computational parts that are both discrete and continuous. In Chapter 4, we introduce basic concepts about control theory. This includes static control and dynamic control including Proportional Integral Derivative (PID) control. Chapter 5 can be seen as a continuation of Chapter 4, where we consider the effects of digital controllers, actuators, and sensors. Such devices give rise to effects such as quantization and discretization. The continuous evolution of a CPS device in the physical world subject to control and sensory discretization and quantization effects can be modeled as a hybrid system. Chapter 6 addresses coordinate transformations, which is key to control are essential for the design of a CPS. As examples we consider robotic arm manipulation and conversion between Euclidean and Polar coordinates. Chapter 7 sheds light on challenges arising in multi-agent systems where agents may

have different, possibly conflicting, objectives. It provides understanding as well as bridges the gap to systems involving multiple devices that, for example, compete or collaborate for resources. Chapter 8 introduces Communication from a point of view that is suitable for the needs of the CPS context, and Chapter 9 does the same for basics of Sensing and Actuation. Appendix A includes a slightly revised edition of the user manual for the Acumen language, as used in this book.

Expected Background

Our goal is to make the book as accessible as possible and to inspire the reader to explore further in a wide range of technical topics far beyond those covered within the book itself. That said, to make the best use of the book, it will greatly help that the reader is familiar with—and not necessarily mastery of—the following:

1. Arithmetic, basic algebra, polynomials, geometry, and trigonometry.
2. Linear algebra and calculus. Again, here it is only familiarity and not mastery that is required.
3. Computer basics. Students should be familiar with the basics of how computers are built and how they work. No programming experience is required.

It is our intent that the book provides a gentle environment to further develop the student's familiarity with these topics. Thus, to the extent that it is possible, the book is self-contained for any university freshman in a Science, Technology, Engineering, or Mathematics (STEM) program.

After This Book

Invention, innovation, computing, physical systems, modeling, and simulation are all vast topics that are impossible to cover in any single textbook. They are also all evolving subjects that are a living part of our world. Even the overarching notion of Cyber-Physical Systems (CPS) that you will learn about in this book is also far too broad for one book. The goal of this book is to give you a sense of what is out there, to help you get started, and—if we succeed—to inspire you. To do that, we must repeatedly press the brakes in each chapter to avoid getting too deep and technical, which is the role of the books and research papers that you will choose to read when you decide that you wish to learn more about these specific topics. Our philosophy in this respect seems to be well-aligned with the following thought articulated by Benjamin Bloom in *Evaluation to Improve learning*, 1981:

> In each subject field there are some basic ideas which summarize much of what scholars have learned over the long history of the field. These ideas give meaning to much that has been learned, and they provide the basic ideas for dealing with many new problems as they are encountered by people who have learned what the field has to offer. We believe that it is

www.dbooks.org

> a primary obligation of the scholars as well as teachers of the subject to search constantly for these abstractions, to find ways of helping students learn them, and especially to help students learn how to use them in a great variety of problem situations. To learn such principles and generalizations adequately is to become a very different human being. Through them one comes to appreciate the beauty and orderliness of the universe. Through them one learns to appreciate the great power of the human mind. To learn to use such principles is to possess a powerful way of dealing with the world. (p. 235)

To pursue this ideal, we have limited the scope of the book in several respects, which are addressed by more specialized textbooks and courses that the reader can easily find through an online search or by asking advisors in a university setting. Thus, a natural step after this book is to pursue more specialization. Understanding how a broad range of areas of science and engineering connect is a different matter from understanding the specific areas or their specialties. The former type of knowledge is called breadth, and that is what this book addresses thoroughly. To do this, we have to leave out a lot of material that is covered in programs that focus on specific subjects, including most aspects of modeling electrical circuits (linear systems, electronics, power electronics), physical systems (statics, dynamics, basic physics, classical mechanics), computer systems (digital logic, computer architecture), communications (communication theory, information theory, networks, wireless networks, real-time networks), control (linear systems, digital control, nonlinear control). Specific directions that are natural to seek deeper specialization in this area are:

1. **Embedded Systems**. A deeper understanding of specific methodologies and technologies is often needed to build actual physical systems. For example, we do not address specific architectures, microprocessors, or popular embedded technologies such as the Arduino or the Raspberry Pi. We also do not talk about common values for resistors capacitors or other devices, nor do we talk about device numbers (such as popular chips delivering transistors or operational amplifiers). A notable textbook with a modern approach to design methodology is:

   > Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*. MIT Press, 2011.
   > Available online at http://LeeSeshia.org.

   Another focusing specifically on managing complexity in design is:

   > Hermann Kopetz. *Simplicity is Complex - Foundations of Cyber-Physical System Design*. Springer, 2019.

A now classic textbook with an engineering focus is:

> Peter Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Second Edition. Springer, 2011.

2. **Security, Safety, Human Factors, Market Analysis, Law**. This book focuses on topics that we expect to be inspiring and engaging to newcomers to this area. At the same time, it is important to recognize that there are many different areas of expertise that apply to invention and product design that cannot possibly be covered in one textbook. Here we have listed a few, and there are probably others. This is another example of how much knowledge there is today, and of why teamwork and collaboration are so important.
3. **Formal Verification**. Our focus is on modeling and simulation as a basic method for exploring new designs and learning core building concepts. We believe that this is very important for beginning learners and for generalists. When building high assurance systems, it can be invaluable to complement traditional design methods with formal verification. This requires reasoning formally (and, when possible, mechanically through the use of a computer) about the properties of the systems that we design. Two notable textbooks covering this area are:

   > André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, 2018. Lecture videos available online at http://video.lfcps.org/.

   and

   > Rajeev Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.

4. **Mathematical Foundations**. Our focus is on mathematics as a language for expressing ideas clearly, for analyzing ideas using mechanized tools such as computer simulation, and for communication between individuals. We put little emphasis on proof in general and on foundations (differential equations, calculus, linear systems, topology, real analysis).

Thus, the purpose of this book is not to cover all these areas, but rather to give you enough knowledge to appreciate the nature and significance of these different areas, and to enable you to peruse the ones you choose to learn more about.

Our Experience Teaching with the Book

The lecture notes that served as the basis for this book have been used as the basis for courses taught at Halmstad University for several years by Walid Taha and by Johan Thunberg, at Rice University by Robert "Corky" Cartwright and by Mike Fagan, and at Alfaisal University by Abd-Elhamid M. Taha. At Halmstad University, the material was developed in the form of lecture notes starting in 2012. These lecture notes were made available online at `bit.ly/LNCS-`$yyyy$ where $yyyy$ is any year between 2012 and 2018, inclusive.

www.dbooks.org

Experience with teaching this course at Halmstad University was reported in the following publications:

> Walid Taha, Robert Cartwright, Roland Philippsen, and Yingfu Zeng. "A first course on cyber physical systems." In *Workshop on Cyber-Physical Systems Education (CPS-Ed)*. 2013.
>
> Walid Taha, Yingfu Zeng, Adam Duracz, Xu Fei, Kevin Atkinson, Paul Brauner, Robert Cartwright, and Roland Philippsen. "Developing a first course on cyber-physical systems." *ACM SIGBED Review* 14, no. 1 (2017): 44–52.
>
> Walid Taha, Lars-Göran Hedstrom, Fei Xu, Adam Duracz, Ferenc A. Bartha, Yingfu Zeng, Jennifer David, and Gaurav Gunjan. "Flipping a first course on cyber-physical systems: an experience report." In *Proceedings of the 2016 Workshop on Embedded and Cyber-Physical Systems Education*, p. 8. ACM, 2016.

This course focused on Part I (Core Concepts) and typically covered material from one chapter from Part II (Selected Topics). The course was initially an elective (optional) course for senior undergraduates and Masters students and was then later converted into a required introductory course for Masters students. The material was also used as a basis for an introductory doctoral student course. At Rice University, it was offered as an elective undergraduate course. At Alfaisal University, it was offered as an elective undergraduate course.

Let Us Hear from You!

We would be very interested to hear from both students and teachers about their experience with this book, both on positive experiences and on areas where the book or Acumen can be improved. To contact us, please write us at cps–book@effective-modeling.org.

Seattle, Washington — Walid M. Taha
Riyadh, Saudi Arabia — Abd-Elhamid M. Taha
Halmstad, Sweden — Johan Thunberg

Acknowledgments

There are several people without whom this book would have not been possible.

The area of cyber-physical systems, which is the subject of the book, would have not materialized if it were not for the efforts of many individuals that believed strongly in the need for it. We are indebted to the whole community, especially the core group of visionaries that include Helen Gill, Edward A. Lee, and Janos Sztipanovits, who played a key role both in shaping the community and in supporting the efforts that resulted in this book.

In recognizing the importance of cyber-physical systems and of modeling formalisms, we are greatly influenced by Paul Hudak and his work on both Functional Reactive Programming and Domain-Specific Languages. We see Acumen as a continuation of this work. Paul's interests spanned programming languages and building RoboCup robots and gave us confidence that a project such as this book would be possible.

Acumen itself would have never been possible without the enthusiasm, excitement, and hard work of its core team of developers, including Adam Duracz, Paul Brauner, Jan Duracz, Kevin Atkinson, Yingfu Zeng, and Xu Fei. The design of Acumen was also influenced by discussions with several other colleagues, especially Aaron Ames, Robert Cartwright, Michal Konečný, Eugenio Moggi, Marcia O'Malley, Roland Philippsen, and André Platzer.

We thank the students and colleagues that took a course based on early versions of this book with Walid at Halmstad University in Spring of 2012: Tantai Along, Maytheewat Aramrattana, Amirfarzad Azidhak, Chen Da, Carlos de Cea Domínguez, Adam Duracz, Carlos Fuentes, Pablo Herrero García, Veronica Gaspes, Nicolina Månsson, Diego Leonardo Urban, Viktor Vasilev, Rui Wang, Fan Yuantao, Yingfu Zeng, and Hequn Zhang. Their interest and enthusiasm provided critical support for continued development of the lecture notes into the 2013 edition as well as several editions that continue now in the form of this book. On the way, numerous other students have contributed to its development.

The book would have not been possible if it were not for support from the National Science Foundation, which funded a project that was initially managed by Helen Gill

www.dbooks.org

and later by Ralph Wachter and David Coreman. The project description included a course outline that very closely matches the contents of this textbook.

The book would have also not been possible without the support of the Head of Education at the School of Information Technology, Halmstad University, Jörgen Carlsson, and the program coordinator for the Embedded and Intelligent Systems (EIS) Master's program at Halmstad University, Stefan Byttner, both of whom encouraged the introduction of the course and provided the academic environment that enabled its development to full fruition.

The chapter on sensing and actuation benefited greatly from discussions and feedback from Per-Erik Andreasson, Emil Nilsson, and Ross Friel.

Earlier versions of several chapters benefited from editing by Mark Stephens. Staffan Skobgy kindly read a draft of the book and provided us with valuable feedback. Several colleagues gave us valuable suggestions, including Maytheewat Aramrattana, Jörgen Carlsson, John Garvin, Mohammad Reza Mousavi, Perdita Stevens, Sotiris Tzamaras, and Kazunori Ueda.

The Acumen manual was at various points edited by Mark Stephens. Valuable input relating to the core language and its intricacies was provided by Kevin Atkinson, Adam Duracz, Veronica Gaspes, Viktor Vasilev, Fei Xu, and Yingfu Zeng. Roland Philippsen and Jawad Masood suggested several suggestions from the point of view of actual users with expertise in analytical dynamics.

Support for the development of Acumen and this textbook was provided by the US National Science Foundation (NSF) Cyber-Physical Systems (CPS) project #1136099, by the Swedish Knowledge Foundation (KK), by the ELLIIT Strategic Network, and by Halmstad University. The development of Acumen was also supported by several corporate and institutional sponsors, including National Instruments, Schlumberger, Volvo Technology Group, and SP (now RISE).

We are enormously grateful to Roslyn Lindquist for developing the beautiful illustrations for this book and for gracefully putting up with our last minute requests for modifications, changes, and more illustrations.

Last, but not least, we would like to acknowledge at least part of the debt we owe towards our parents, including a mother who painstakingly worked to ensure that we know the difference between how the digits for 2 and 3 are written and a father for answering endless questions. We are forever grateful.

Contents

www.dbooks.org

www.dbooks.org

www.dbooks.org

Part I
Core Concepts

Chapter 1
What is a Cyber-Physical System?

Our starting point is to reflect on our world today and to consider examples and characteristics of what has come to be known as Cyber-Physical Systems (CPSs). We then look at the innovation process and the associated workforce challenge. Next, we explain how the field of CPS brings together several previously distinct fields such as Embedded Systems, Control Theory, and Mechatronics. We conclude with an overview of what you can expect to learn from this book.

1.1 Our Planet. Our Knowledge. Our Destiny

We live in a world that is changing at a much faster pace than ever before. More than at any other time in our history, there is a pressing need for new ways of looking at science, technology, and social phenomena—ways that can help us understand our planet, ourselves, and how we can take control of our collective destiny. Our population and our consumption of the Earth's resources are soaring. Estimates put the world's population near eight billion[1] and annual per capita energy consumption at about 20 MWh/year, that is about 2 tonnes of oil equivalent per year.[2]

Fortunately, with the development and availability of powerful communication infrastructure, our awareness of the state of the world and our ability to influence it are also improving. For example, it is remarkable that the number of smartphone users on the planet has already topped three billion,[3] which is close to half of the world population. This means that our collective ability to share information and cooperate has already reached a remarkable level.

[1] United Nations' World Population Prospects 2019.

[2] The International Energy Agency (IEA)'s statistics reports about 2 Tonnes of Oil Equivalent (TOE) for Total Primary Energy Supply (TPES) per capita. This amounts to 22 Mega-Watt hours (MWh) for the conversion rate of 11 MWh per TOE.

[3] Statista reports 3.3 billion smartphone users worldwide.

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_1

www.dbooks.org

Without a doubt, our knowledge in the areas of science, technology, and social sciences has played a key role in shaping the modern world. Of particular importance is knowledge in the areas of computing and communication. These are areas that have recently witnessed exceptionally rapid development due to advances in digital computing and communication. In the history of human knowledge, these are relatively recent developments that are currently treated as separate disciplines. In research and in education, topics relating to digital computing and communication have been viewed as distinct areas of specialization.

The workforce marketplace has long prized specialized expertise. But in today's knowledge economy[4] specialization also poses new challenges: The process of innovation that leads to new breakthroughs relies critically on interdisciplinary collaboration and insight. This is especially the case for digital products that have a physical presence. While many successful Internet-based products benefit from the relative ease of innovation in "pure information technology," such ease is not currently present for systems that sense the physical world or can interact with it. The reasons for this problem are multifaceted. To develop a system with a physical presence or "embodiment" requires diverse expertise from many areas (e.g. Mechanical Engineers, Computer Scientists, Electrical Engineers, Computer Architects), and experts in these areas may not have a common language. What is more, in many cases addressing this problem requires finding commonalities in the foundations of these disciplines as well as ways to reflect them in the education of the experts in these different domains.

We are not alone in holding this view. Given the extent to which digital technologies pervade modern life, a growing community of researchers and educators believe that closer integration with many other disciplines will be needed in the future.[5] The development of numerous innovative products in the home, health, and entertainment sectors all require close collaboration between several innovators with different advanced training (such as Masters or Ph.D. degrees). The current organization of disciplines at university level appears to hamper the cross-disciplinary communication needed to realize such collaboration. The organization of scientific knowledge and its educational delivery may seem like distant problems, but they have concrete implications for the workforce and, as a result, affect our collective ability to contribute to the world we live in.

1.2 Observe. Understand. Innovate

One of the goals of this book is to help you learn skills that can make you become a better observer of the world around us as well as new products and technologies. This is one of the reasons that the book covers physical modeling. Doing so has a secondary purpose of increasing your awareness of the mathematics that can be

[4] See for example Powell and Snellman, the Knowledge Economy, 2004.

[5] See for example Stankovic, Sturges, and Eisenberg, A 21st Century Cyber-Physical Systems Education, 2017.

used to model our world, and creates opportunities for showing how this type of mathematics can provide insights into this world. Mathematics gives us a way to test our understanding of different phenomena, and as such provides us with a means to improve this understanding. Our understanding of problems that need to be addressed and components that can be used to fill these needs are prerequisites for successful innovation.

1.2.1 Cyber-Physical Systems and Hybrid Systems

While several definitions are used in the literature, an early definition is simple and intuitive enough: According to Lee and Seshia, the term Cyber-Physical Systems (CPSs) was coined by Helen Gill around 2006, at the National Science Foundation (NSF) in the U.S., to refer to the integration of computation with physical processes.[6] Often, it also has a communicating or networked aspect.

If we consider this description from the point of view of the mathematics needed to model them, we can begin to see some technical requirements: Computational components give rise to a need for discrete modeling; physical components give rise to a need for continuous modeling; and a communications/networking aspect gives rise to a need for probabilistic and possibly also a game theoretic modeling. Many fields of mathematical modeling will use either continuous or discrete mathematics, and not mix them. And, in fact, combining these two can lead to some fundamental technical problems. Adding probabilities on top requires an additional level of care. But at a more practical level, simply expressing models that combine both discrete and continuous components requires a modeling formalism that can express both. That is precisely what hybrid (continuous/discrete) systems provide. In this book we will make extensive use of continuous systems, discrete systems, and hybrid systems, for modeling CPSs, and will make modest use of probabilities in introduction to basic concepts in the treatment of game theory, communication, and sensing and actuation.

1.2.2 Examples

Many examples of CPSs surround us in everyday life. In the home we have cleaning robots, smart lighting systems, and smart heating, ventilation, and air-conditioning (or HVAC) systems.

For transportation we have cars, planes, motorized scooters, Segways, and electric bicycles. Existing systems like these are representative of the areas where we can expect to see significant innovation and development in the future. For instance,

[6] Page xii of Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber–Physical Systems Approach, http://LeeSeshia.org, ISBN 978-0-557-70857-4, 2011.

www.dbooks.org

while cars have been around for almost 350 years,[7] new features like Lane Departure Warning Systems (LDWS) are now available in vehicle product lines.

Medical solutions include pacemakers, insulin pumps, personal assistance robots, and smart prosthetics. Many of these technologies did not exist until recently, and have the potential both to save lives and to significantly improve health and well-being. Wearable fitness and health-monitoring systems promise to have a hugely positive impact on users, whether or not they are healthy or have a physical or a cognitive disability. Health monitoring systems are just one example of the whole area of sensor networks, which includes those made of tiny sensors used to observe large land, marine, or aerial spaces.

Finally, examples from the energy sector include windmills, smart grids, and various energy harvesting technologies. In fact, it is no exaggeration to think of our entire planet as a single, massive CPS.

While the skills you will acquire in this book are centered primarily around systems that you can construct, these same skills will also help you better understand existing systems.

When we survey such examples we notice that some are more futuristic or more spectacular than others. For example, the Segway may seem more critically dependent on having a "cyber" (or computational) component than a car. A car can exist and function without a computational part. But a Segway has just two wheels, and it is not at all obvious whether it can even exist or function without the computational component that keeps it upright.

Mechanically, the Segway is an unstable system that we can prove mathematically ought to fall if the computational component that keeps it upright is switched off. The Segway uses a *real-time* control system that runs on a dedicated, embedded computer. Whereas, traditionally, many systems have been designed to be stable in the absence of active control, the Segway and many generations of jet fighters (such as the Saab JAS 39 Gripen) have designs whose stability depends critically on active control.[8] The idea in all of these cases is that pursuing this path leads to more efficient designs that can realize functionality that would be impossible without the active control. For a variety of technical reasons, the control itself would not be possible without a computational component. One of the goals of this book is to help you understand what makes a system more challenging to realize—this is often exactly the same thing that makes it seem futuristic and spectacular.[9]

While powerful airborne vehicles can be very impressive, in the grand scheme of things their applications are relatively limited, and their impact on daily life can be minimal. In contrast, smart home technologies may have a bigger and more direct impact. For example, significant energy is expended in heating and cooling buildings, washing and drying clothes, and transporting people and commodities to

[7] History of The Automobile provides an example that could be as old as 1672.

[8] Aircraft functions of this type are sometimes referred to as supermaneuverability.

[9] Clarke's third law states that "Any sufficiently advanced technology is indistinguishable from magic." In a sense, we are saying here that what appears magical is often also what is at the edge of our knowledge, and is therefore where we need to push further to increase our understanding of the world.

and from homes. This means that the optimization of HVAC systems can have a significant impact on global energy consumption. Similarly, computation can enable sophisticated hydroponic gardening right in the home to provide us with a local supply of fresh nutrition. Combining the two may also enable more advanced management of various parameters of comfort in the home (air moisture levels, CO2 levels) and improve health and living conditions.

1.2.3 Computational vs. Physical Systems

It is common when we first hear the definition of Cyber-Physical Systems to assume the computational and physical subsystems are distinct. Often this will be the case, but not always. The key point is that when we use these designations we are making an abstraction. Every physical system that we can think of, by definition, will have physical components. At the same time, computation is an abstract notion that we can identify when we recognize the presence of a mathematical function or relation. Today, we often assume that computation is performed digitally. But this is not always the case: Analog computers have long existed, and quantum computers are already being built. Even when we limit ourselves to digital computation, the distinction is still not clear: A modern microprocessor has aspects that simultaneously touch upon essentially all computational and physical aspects discussed in this book. Especially to microprocessor designers, it is simultaneously physical and computational in a very real way, and both its physical and computational characteristics affect each other directly.

1.2.4 Biological and Intelligent Systems

While the focus of much CPS research and education is on systems that we can construct and develop into products, it is also instructive to reflect on one class of systems that has many characteristics of CPSs—namely, living creatures including ourselves. While we often view living creatures as purely biological systems, living systems clearly have physical manifestations. These manifestations simultaneously exhibit a range of physical phenomena, including mechanical, chemical, electromagnetic, and optical. At the same time, they often seem perfectly capable of computation and communicate on a regular basis. Living systems can be a great source of inspiration for the design of new CPSs and, similarly, advances in CPSs could provide us with better tools to improve understanding of life and ourselves. For example, living systems are the inspiration for the field of Artificial Intelligence, which aims to develop computational methods for solving problems that are important in the real world, but for which we may not even have a clear notion of what an acceptable solution should be.

www.dbooks.org

No part of this book is dedicated to such systems, but the principles covered are still applicable.

1.3 Developing New Products

To collaborate on developing new products, it is useful to have a shared concept of product development. Different organizations, and even different individuals, have different approaches. For this reason, it is important to consider features common to all processes, and to use them as a starting point. We can at minimum distinguish four artifacts in the process of product development: idea, model, prototype, and product. This is by no means an exhaustive list, but is sufficient to allow us to describe key aspects of any such process.

An **idea** is a mental notion of an object, function, or design. Ideas are the colorful realm of inspiration. Good ideas are usually created in an environment where we have a good understanding of the problem that needs to be solved, the context in which the solution will be applied, and the space of feasible solutions. Generating good ideas, therefore, requires awareness of the real needs as well as what science, technology, and society makes possible.

A **model** is a formal description. In this context, the word formal means having form, such as syntactic or geometric manifestation. Thus, a textual or graphical description can be seen as formal. Mathematical descriptions are also excellent examples of formal models. Moving from an idea to a model makes it possible to apply conceptual and computational tools to analyze the new function or design. Obviously, mathematical descriptions have the advantage of being amenable to mathematical analysis and reasoning, but other formal models can also enjoy similar properties. Models written in plain language and including possibly a few drawings, as is common in patents, can be more accessible to a general audience than mathematical models. Models can also be realized as computational codes, which can be used for efficient early testing and design space exploration.

A **prototype** is a physical, operational instance of the model. This enables more early testing, which allows qualitatively different validation than what can be achieved analytically or computationally. Prototypes can be used to evaluate the safety as well as the response of users to first-hand experience with the product concept. Today, additive manufacturing technologies such as 3D printing are proving to be a powerful tool for enabling the rapid construction of prototypes with minimal delays and costs.

A **product** is a manufactured commodity that can be sold commercially to end users. While the public often identifies new technology with new products, building products involves much more than technical innovation. Creating products involves systematic analysis of the market, finding financing, recruitment, management, production planning, logistics, marketing, sales, billing, customer support, and other business operational activities. Even though this book does not cover these aspects, it is important to be aware that these are significant parts of the effort to deliver a finished product.

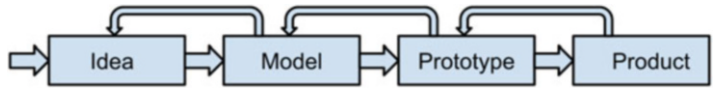


Fig. 1.1 A typical process for developing a new product

Figure 1.1 depicts the typical relation between such artifacts, and the basic iterative cycle for moving from one to the next. As a rule of thumb, it is often the case that moving from one stage of development to the next involves at least one order of magnitude (that is, at least ten times) more effort than the previous stage. This compounding of effort in moving from one stage to the next, combined with the fact that the process often involves significant iteration and refinement, means that maximizing the quality of the intermediate product *before* we move to the next stage can significantly reduce the final cost. The high cost of iterating back from a late stage in this pipeline is a big motivation for modeling and simulation. In the worst case, such high costs can include production defects such as Boeing's problems with the 737 Max model, Toyota's brake system problem, and the Intel Pentium bug.

These examples illustrate that the better equipped we are to produce better ideas, models, and prototypes, the more successful we can be as innovators. This observation highlights the importance of techniques such as virtual prototyping (using rigorous modeling and computational simulation), testing (of both computational and physical components), formal verification (of discrete, continuous, and hybrid systems), and model-based production and manufacturing.

1.4 Is the Field of Cyber-Physical Systems New?

While the term cyber-physical systems is relatively new, we should consider whether the field or *idea* behind this way of studying systems also new. It is important that inventors, innovators, leaders in a discipline reflect on and understand the nature and the reasons for the existence of the field. When we do this for CPS, we must consider a wide range of related disciplines and reflect on how they are connected. In the following we consider several relevant concepts and discuss their relationships to CPS.

A **hybrid system** is a mathematical model that features both continuous and discrete behaviors (related areas are **switching systems** and **impulsive differential equations**). While many CPSs can be modeled mathematically as hybrid systems, it is important that we distinguish between the concept of the actual, physical system and its mathematical models and the techniques used to study such.

The mathematical models are there to capture observed behavior, but as new observations occur that cannot be explained by the model it has to be modified or even replaced. It is a common mistake to think of models of physical systems as continuous and models of computational systems as discrete. Models of physical systems can be continuous, discrete, or a combination of both. For example, Bohr's model introduced

www.dbooks.org

in 1913 put forth the notion that electrons can only exist in discretely different orbits around the nucleus of an atom, planting the seed for quantum mechanical models. In this book, we study hybrid models of a simple bouncing ball (flying is a continuous behavior, bouncing is a discrete event). Furthermore, at the quantum level, many important phenomena cannot simply be viewed as continuous or discrete systems; rather, as probabilistic systems. Computational systems cannot always be viewed as purely discrete systems. Digital computers are generally implemented as continuous electronic circuits designed to operate reliably only when viewed as discrete systems. Also, there are systems known as analog computers that are continuous systems. Some examples of such systems can be realized perfectly using digital computers. Finally, quantum computing is an active research area that relies on probabilistic models. CPSs are real-world objects, whereas hybrid systems are a mathematical abstraction.

An **embedded system** is a computational system embedded in a physical system. Any CPS contains an embedded system. The main distinction is that the term "embedded system" reflects a primary focus on the computational component (that is *embedded* in a larger, physical system). Traditionally, research on embedded systems focused on problems such as formal verification of discrete systems (automata), hardware design, minimization of energy consumption and production cost, as well as embedded software development. The CPS view emphasizes the importance of taking into account the physical context of the computational system which is often necessary to design, test, and verify the functionality that we are developing.

A **real-time system** is one which must respond to external changes within certain timing constraints. Many, but not all, real-time systems are embedded systems. For example, an automated trading agent would not normally be viewed as an embedded system, even though it must operate under strict timing constraints if it is to function usefully in response to rapidly changing market conditions. Traditionally, research in real-time systems has focused on scheduling or real-time tasks in systems that have periodic or aperiodic request patterns, multiple (interchangeable) computational resources, tasks of varying priorities, and real-time communication. Naturally, research in this area often focuses on worst case run time requirements. A CPS may or may not be a real-time system: The control system in a car has real-time constraints, but the sound system does not necessarily have such.

Reliability is the ability of a system to continue to perform its function despite the failure of some of its components. Reliability can be achieved in several ways, starting from building components from stronger materials to adding redundancy and error-checking to detect and try to compensate for errors and failures. In many domains, including computational systems, probabilistic methods have been used effectively to increase reliability, while keeping costs manageable.[10] Probabilistic methods, however, are only one tool for designing and constructing reliable systems. Mastery of other more fundamental concepts in systems design allows us to become more effective users of probabilistic methods. Different CPSs have differing reliability demands, and reliability need not feature prominently in the develop-

[10] For some examples of how probabilities are used as a model of reliability, refer to Johan Rhodin's Wolfram Technology Conference talk.

ment of all individual CPSs. However, in general, as connectivity between different CPSs increases, there is also an increasing need to consider system-wide reliability implications for each type of individual cyber-physical subsystems.

Dependability is a more holistic notion that can encompass several related attributes, such as availability, reliability, durability, safety, security, integrity, and maintainability. In the context of the interdisciplinary field of **systems engineering**, it is viewed as a measure of these combined attributes. Systems engineering is often viewed as a field of both engineering and engineering management, reflecting its unique point of view between what is traditionally engineering and traditionally management. We see both systems engineering and dependability as of great importance for inventors and innovators, and that the content covered in this book will give the reader a solid foundation to pursue further studies in these disciplines.

A **multi-agent system** is a mathematical model consisting of interactive objects.[11] It is often associated with the mathematical discipline of **game theory**. Multi-agent systems and game theory provide useful techniques for modeling and reasoning about concepts such as *belief*, *knowledge*, *intent*, *competitiveness,* and *cooperation*. Notions of intelligence are often considered in relation to multi-agent systems and mathematical models of both discrete and hybrid (continuous/discrete) games exist. In contrast to the study of hybrid systems, the multi-agent systems and game theory focus on the behavior of collections of agents rather than individual agents.

Finally, CPS is closely related to the disciplines of **mechatronics**, **control theory**, **robotics**, and the **Internet of Things (IoT)**. A common feature of these disciplines is that they are highly interdisciplinary. CPS can be viewed as an attempt to take an even more all-encompassing approach than the first three disciplines, and being quite comparable to the last one (IoT). Texts on mechatronics may not necessarily dedicate a large part to communications and networking, or to hybrid systems foundations. Textbooks in control theory covering issues such as hybrid systems are still considered relatively advanced and specialized. Robots[12] are obviously great examples of CPSs, illustrating many of the challenges involved in designing innovative CPS products. We will use a ping pong (table tennis) playing robot as a running case study in a project that we will develop incrementally in different chapters. Both CPS and IoT take the view that the world is becoming highly connected and computational, and it is possible that the two approaches will converge. Historically, CPS is seen by some as having emerged from the Control Theory community, whereas IoT as having emerged from the Communications community.

[11] The definition we use is inspired in part by the one used in the article Multi-agent Systems. More information on the subject can be found in the online text of Shoham and Leyton-Brown.

[12] The definition we use is more specific than the one used in the article Robot. We exclude usage of the word when referring to purely computational ("virtual") systems, which we view as metaphorical use of the term.

www.dbooks.org

1.5 What You Will Learn from This Book, and How

The specific goals of this book include:

- Helping you appreciate the value of several distinct disciplines to being an effective innovator. The disciplines we will consider include physical modeling, control, hybrid systems, computational modeling, and game theory.
- Providing you with experience in model-based design. This experience will help you be comfortable with the differences between actual physical systems and phenomena on the one hand and mathematical models on the other. It will also help you appreciate the importance of virtual prototyping for rapid product development and rapid accumulation of knowledge about a domain or a product.
- Giving you a chance to review and sharpen your mathematical skills, including mathematical modeling, differentiation and integration, and solving simple algebraic and differential equations.

The book involves a simulation-based project. Simulation has many valuable uses, including:

- Providing a well-motivated opportunity to be exposed to mathematical modeling.
- Avoiding the need for the existence of analytical solutions, that is, solutions in the form of a formula that we can calculate with, because they usually exist only for a smaller class of problems than what we can simulate.
- Enabling many more virtual experiments to be run than would be possible with physical ones. Physical testing can be prohibitive for reasons including cost, safety, and controllability.
- Enabling easier measurement and evaluation than may be possible with physical experiments.
- Providing an opportunity to learn an important skill in CPS design, namely, systematic experimentation.
- Increasing the chances of producing a successful CPS design.
- Producing many useful visualizations.
- Facilitating the creation of animations and computer games.

As mentioned earlier, the project will focus on studying a robotics problem, namely, how to design a robot that can play ping pong. There are several reasons why robotics is a useful example of a CPS domain, including:

- It involves intimate coupling between cyber and physical components.
- Even simple, rigid-body modeling of 3-D dynamics requires the use of hybrid, non-linear Ordinary Differential Equations (ODEs).

Designing robots gives rise to:

- Significant embedded and real-time computation requirements.
- A need to consider issues of communication and belief.

Through these experiences, we can develop a sense of how designing systems becomes more challenging as certain characteristics/parameters of the system increase, such as:

- Model complexity resulting from:
  - Increasing the degrees of freedom (in models of physical systems).
  - Increasing the size of state space (in models of computational systems).
  - Reducing what can be sensed or actuated (in the control system view).
  - Reducing the dependability of the components (across all aspects).
- Simple equations vs. time dependent equations, such as in going from:
  - Linear to non-linear Ordinary Differential Equations (ODE)s.
  - ODEs to Partial Differential Equations (PDE)s.
  - ODEs to Integral/Differential Equations (IDE)s.
- Models of computation, such as in going from
  - Boolean circuits to automata to Turing machines.
  - Systems that are either discrete only or continuous only to systems that involve both types of behavior (hybrid systems).
- Uncertainty about model parameters, structure, dimensionality, and determinism.

Complex systems are challenging for engineers developing state-of-the-art tools, as well as for researchers conducting basic research. Developing a sense of what today's analytical and computational tools can handle will enable you to be a more effective innovator by focusing on designs that are feasible to analyze and design. It will also enable you to be a more effective researcher by understanding where the frontiers of knowledge lie, which is knowledge that is prerequisite to making fundamental research advances.

Congratulations on completing this introduction! Before we continue and summarize the chapter, we suggest readers that are using this book as course material for a course on CPSs to consult the Acumen manual, see Appendix A. Acumen will be used as the simulation and modeling environment in the project.

1.6 A Writing Tip

We have noticed over the last 10 years significant confusion about how to use the abbreviation CPS as a short hand for the term Cyber-Physical Systems in the plural case in particular. This point is worth raising because abbreviation practice makes reading easier and can help avoid confusion. The main tricky bit seems to be that the abbreviation CPS is used both as the name of an area that studies a type of system and a plural for a group of systems. In particular, if we want to abbreviate "The area of Cyber-Physical Systems is new" we would replace the term by CPS. In contrast, if we say "Both cars and robots are Cyber-Physical Systems" we would replace the

www.dbooks.org

term by CPSs. When in doubt, think of the term Operating Systems (OS). If we want to abbreviate "The area of Operating Systems is new" we would replace the term by OS. In contrast, if we say "Both Linux and BSD are Operating Systems" we would replace the term by OSs.

1.7 Chapter Highlights

1. Cyber-Physical Systems: Today and Tomorrow.
   (a) Examples: autonomous vehicles, the smart grid, smart homes, smart cities.
   (b) A smart planet.
   (c) The challenges for mankind: innovation, safety, privacy, security, regulation, and ethics.
2. Is CPS new?
   (a) Relation to other fields.
   (b) Workforce and educational challenge.
3. The innovation process.
   (a) Distinct and easily recognizable stages: Idea, Model, Prototype, Product.
   (b) Order of magnitude increase in cost with each transition between stages.
   (c) The highly iterative nature of the process.
   (d) The cost of failure. How "late" discovery of flaws can be exponentially costly.
   (e) The role of virtual prototyping and verification.
4. Why this book?
   (a) The goal is to introduce you to the field, so that you know where to look for knowledge when you need it.
   (b) Emphasis on modeling and simulation, which help you understand the math, accelerates your ability to experiment and innovate, and is fun!
5. What you will learn from this book
   (a) Recognizing the sources of technical complexity through learning about hybrid systems, control, communication, and game theory. This includes understanding the nature of the system dynamics (linear, non-linear, ODEs, etc.) and the size of the systems studied (in the discrete and the continuous domains).
   (b) Experience through an in-depth case study (the project)

1.8 Study Problems

1. Explain, in your own words, how you imagine mathematics can help you become a better innovator.
2. Consider a product that is expected to appear on the market in the near future. For this product, describe the issues that should be addressed at the idea, model, prototype, and product stages.
3. Explain, in your own words, and based on your own experience, all the possible challenges that one may encounter if one wants to design and build a robot that can play a game of ping pong with a human. Be as specific as you can about the challenges that you identify. You may search online for results related to this problem, and include ones that caught your attention. Limit your total answer to 600 words and use only one page.
4. Repeat Problem 3 for any other CPS of your choice. Discuss with a colleague.
5. Give an example of a problem that is important to each of these research areas. Make sure that in each case this problem is central to the area: Embedded systems, Real-time systems, Reliability, Mechatronics, Control Theory, Multi-agent systems, and Internet of Things.

1.9 Lab: Warm Up Exercises

The purpose of the lab activities is to bridge between the theory discussed in the chapter and the more experimental activities of the project. The labs and the project will use Acumen. Acumen is an open source modeling and simulation environment specifically aimed at CPS design. The Appendix found at the end of this book contains a manual for Acumen. The distribution contains a set of examples that will be used in the Labs.

The purpose of this first lab is to connect to practical experience with modeling and simulation. The activities of the lab are to:

- Get Acumen set up on your computer. To do that
  - Download the 2016/8/30 Acumen distribution
  - Uncompress it or unzip it
  - Run the Jar file that you will find in the uncompressed folder
  - If this does not work right away, then go to Java.com, download and install Java 8, and restart your machine, run the Jar file again.
- Work through the first set of examples in Acumen. The purpose of these exercises is to practice
  - Geometry (in 3D, like our real world, and what we need to understand in order to talk about robot arms)
  - Dynamics, the basic differential equations needed for things like $f = ma$

www.dbooks.org

- Animation, which is the simple combination of the above two things.
- Working through these problems during the lab will help familiarize you with Acumen concepts and syntax, and will help you learn how to deal with simple error messages (mostly relating to syntax)

The examples are in three folders in the `01_Introduction` examples folder. The first sequence (00–09) are examples of static 3D forms; the second sequence (10–19) are examples of dynamic behaviors and their plots; the third sequence of examples (20–29) are dynamic 3D forms.
Go through all the examples in the three sequences, one by one. For each example, read the text model, read the model itself, write down how you expect the model to behave, run the model, and then compare your expectations to what you saw when you ran the model. Next, make small changes to the model to test your understanding of how it work. You will notice that each example contains questions and challenges to help you make sure that you fully understand each model. Going through all three sequences is a really good way to brush up on the mathematical concepts that will be used in the coming few chapters.
Feel free to play and experiment and come up with whatever shapes you would like! The material introduced in these examples is enough to let you create a lot of really interesting 3D shapes.

The third sequence of example will be particularly helpful preparation for the coming chapters, as it introduces several basic concepts that illustrate how differential equations are used to model dynamics. Here is a brief summary of these concepts: Simple dense-time models are introduced with the equation $x' = 1$. Definite integration of constants and polynomials arises naturally in discussing such equations, but such integrations are needed only when we calculate solutions by hand. A simulation environment that solves these differential equations (like Acumen) can also calculate numerical solutions automatically. Higher derivatives are illustrated by the equation $x'' = 1$. Exponential functions are introduced by the equations $x' = x$ and $x' = -x$. Complex exponential functions (trigonometric functions) in the solution forms are introduced by the equation $x'' = -x$. These systems also exemplify linear differential equations. Physical systems that can be modeled by these types of equations (falling ball, electric flows, AC current, etc.) are briefly touched upon. The pendulum equation $x'' = -\sin(x)$ is given as an example of the non-linearity that arises naturally in mechanical models.

1.10 Project

The goal of the project is to provide you with a concrete case study for applying what you learn in each chapter and to provide you with an opportunity for first-hand experience of the challenges and rewards of designing CPSs. To achieve this goal, the project will focus on an example of a CPS design task—namely, building a machine that can play ping pong. While building a robot can often require significant time, space, and financial resources, we can use model-based design and simula-

tion techniques to produce a comprehensive blueprint and significantly sharpen our knowledge and skills with much less cost.

The first project activity will give you a sense of the overall way you will work with models in the project. The Cannon Beach game is intended to help you apply your knowledge of physics, control, differential equations, and other areas to solve a challenging design problem. Your knowledge will be deeper in some areas than in others. This variability in expertise is normal, and is typical in real-world situations. It is an important engineering skill to be able to find ways to solve difficult problems with the experience you already have, and to continue to develop your skills while working on new challenges.

The picture above shows what the game visualization looks like. From left to right you have a pile of cannonballs, the cannon, and the target. When the game is being played, the target (bullseye) appears at different locations. When the cannonball is fired, it travels with a given speed and the angle where the cannon is pointing. The angle is measured from the ground to the cannon. The bullet is subject to the effect of gravity and air resistance. You probably already know a lot about the effects of gravity for this problem, but not so much about air resistance. This problem is a chance for you to learn a bit about how air resistance is modeled, and how it affects solving problems.

Your task is to design a player (or "controller") that sets the direction of the cannon so that you hit the target. The more accurately you hit the target, the more points you get.

This controller takes the target position as input, and must compute a value for the angle. You can assume for now the velocity is given (see model) and leave it as it is. In terms of specific technical details, you can assume that gravity $g = 10\,\mathrm{m/s^2}$ and air resistance coefficient $k = 0.01\,\mathrm{m/s^2}$. The target can appear at any point between positions -6 and 6. You will need to read the model of the game to learn the full details.

What you need to submit for this assignment is a modified version of the following template. In this template, the model parameters *position*, *target*, *velocity* represent the cannon's position, the target's position, and the bullet's velocity, respectively. The angle output should between 0 and π.

www.dbooks.org

```
model ResponseExample(position,target,velocity) =
initially
  angle = 0, distance = 0
always
  distance = target - position,
  angle = pi/4
```

1.11 To Probe Further

At the end of each chapter you will find a collection of pointers for further exploration. Some will be more technical, others will be lighthearted. The idea is to help you explore further beyond the material presented in the chapter.

- Background on CPS, mechatronics, innovation, and robotics
- Lawrence Lessig's lecture on Threats to a Freedom to Innovate
- A US News article on the rise of open education
- Hans Rosling's lecture on the world population: Religion and Babies
- Legal concerns about Space Oddity
  - Chris Hadfield's website
  - Economist article about Copyright in Space
- Economist article on Open-Source Medical Devices
- Wired article on 3D printing, a technology that can revolutionize manufacturing.
- New York Times article on Why Innovators Get Better with Age
- A proposed mock Coursera course about Coursera, the leading MOOC provider
- A mathematician's lament
- A humorous video illustrating problems this course cannot help solve
- The European Summer School on CPS
- Technical videos from the Halmstad Colloquium

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Chapter 2
Modeling Physical Systems

How can we use math to predict the behavior of physical systems? In this chapter we cover principles for modeling physical systems; differential equations, with a focus on Ordinary Differential Equations (ODEs); systems of equations; vector calculus; one-, two- and three-dimensional mechanical systems (statics and dynamics); and resistive and linear electric circuits.

2.1 Reconnecting with the Physical World

In Chapter 1 we explained how CPSs are ubiquitous in modern society. The systems are closely coupled with the physical world. To understand these systems and to build the skills required to develop new innovations we need to have some experience with modeling physical systems. Often, it is the physical nature of the problem that will either drive the solution or affect the extent to which a new product solves the given problem.

The physical phenomena that are useful to model depend on the problem. For different problems, we may wish to model physical, chemical, biological, economic, or even social phenomena. While working on our solution, it can be very helpful to consult specialized textbooks or research papers about modeling any particular phenomena that seem to have a significant effect on the behavior of our system.

Physics has a particularly important role in the design of CPSs. It encompasses mechanics, electromagnetics, optics, and thermodynamics, all of which are often present even in the realization of the cyber (computational) components of a CPS. Fortunately, understanding the basic principles for such systems can help us both in analyzing real-world problems and in developing the "mathematical muscles" that can help us to learn more about modeling on our own when required.

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_2

www.dbooks.org

2.2 Conservation Laws

A theme in physical modeling is the presence of conservation laws. These laws are the workhorse of the process of mathematical modeling, so, it is very useful to keep an eye out for them. Examples from mechanical systems include:

- Conservation of energy.
- Conservation of momentum (translational and rotational).
- Conservation of mass.

The main example from electrical systems is:

- Conservation of current.

Deeper principles in physics allow us to connect some of the principles mentioned. For example, the famous equation $E = mc^2$ allows to connect energy and mass. For us here, the importance of these principles stems from their utility in allowing us to model and analyze physical systems.

2.3 Elements in Mechanical Systems

In addition to conservation laws, mechanical and physical systems will have standard elements. Examples in a mechanical system include:

Mass The basic rule for a mass is $F = ma$, where F is force, m is mass, and a is acceleration. Note that this is a second order differential equation, since a is really x'' (the second derivative of x), where x is the position. Note also that this equation holds when we are in one, two, or three dimensions. In all cases, F and a are both n-dimensional vectors where n is the dimension of the space we are working in, but m always remains a scalar (single-dimensional) value.

Force (Including Gravity) Force is an element that can represent the basic mechanical interaction between two objects. It is also involved in the basic rule $F = ma$. For our purpose it can be through physical contact or the effect of gravity.

Lever This is an element that consists of a long rod balanced on a pivot, which has two forces applied to it that would individually cause it to rotate in opposite directions. The lever is in equilibrium if the torques around the pivot are equal. In the case of the example shown in Figure 2.1, this means that

$$m_1 g a = m_2 g b. \tag{2.1}$$

Interestingly, we can see gravity appears as a multiplier on both sides. This means two things. First, in the rare situation where there is no gravity, and so, $g = 0$, the equation holds for any masses and lengths. More practically, in non-zero gravity we can divide both sides of the equation by g to get

$$m_1 a = m_2 b. \tag{2.2}$$

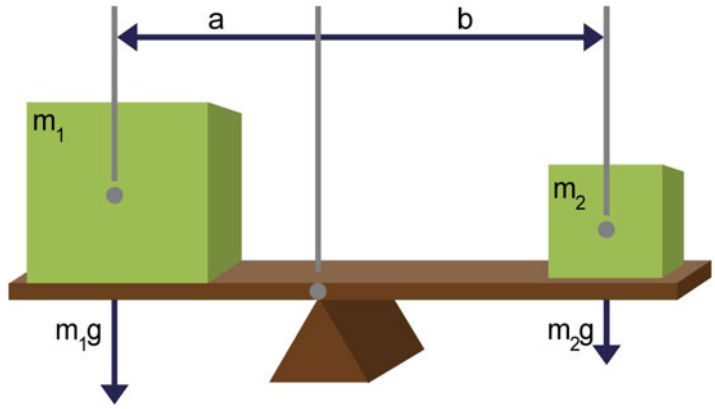


Fig. 2.1 Two masses m_1 and m_2 are placed on a rod balanced on a pivot. The distances to the pivot, a and b, can be chosen so that the system is in equilibrium and does not move

Friction is a phenomena that generates a force that opposes the direction of (potential) movement, and is usually proportional to a normal force that is normal to the direction of the movement (often a surface along which another object is moving). In the case of the example shown in Figure 2.2, the normal force would be pointing upward.

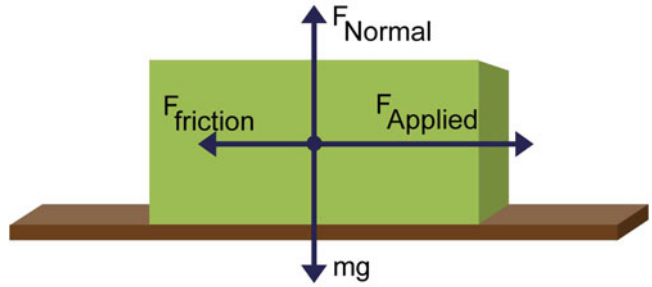


Fig. 2.2 A box is pushed with a force $F_{Applied}$ to the right. A force $F_{friction}$ is acting in the opposite direction. The gravitational force is negated by a normal force that points in the opposite direction

Spring The spring is a component that is modeled by Hooke's law, which is usually represented by the equation

$$F = -kx, \tag{2.3}$$

where F is a force, x is a displacement relative to a neutral (natural) position for the spring, and k is a (scalar) coefficient that relates these two values. Figure 2.3 illustrates this behavior by presenting three instances of the same spring in equilibrium. In the first case there is no mass attached; in the second, there is a single mass attached;

www.dbooks.org

in the third, there is a double-mass attached, leading to double the extension seen in the middle example. It is important to note again that both the F and x quantities can be n-dimensional, and that the k parameter is always a scalar.

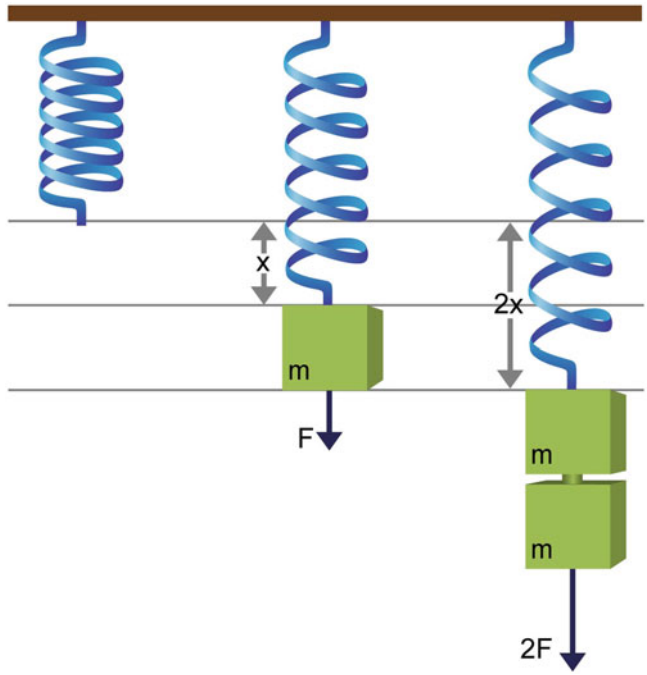


Fig. 2.3 Illustration of Hooke's law. If an object with mass $2m$ is attached to a spring, its length increases twice as compared to when an object with mass m is attached to the spring

Damper A damper is a device that creates a force against the direction of movement, and in proportion to the speed of the movement (compare this to friction mentioned above). The rule for a damper therefore has the form

$$F = -kv, \tag{2.4}$$

where F is a force, k is a constant, and v is a velocity. We use k for constants. Thus, this is not the same constant as the one used in Hooke's law, i.e., Equation (2.3). The example shown in Figure 2.4 presents a combined example involving a mass, a damper, a spring, and an external force. This example is one dimensional.

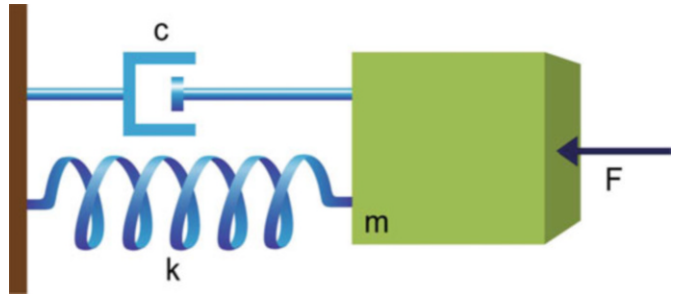


Fig. 2.4 A system with both a damper and a spring. Dynamical behaviors of such systems will attenuate over time and the system converges to a static configuration

Air Resistance This is a force created by air and applied to any object moving through it. At slow speeds (much less than the speed of sound), we can view this force as being proportional to the square of the speed. Thus, the formula can be expressed in the one-dimensional case as

$$F = -kv \cdot \text{abs}(v). \tag{2.5}$$

We use the absolute value function abs in this manner to make sure that the resulting value gives the force the right sign. Now, in the more general 3D-setting, the equation would be expressed as follows:

$$F = -kv\|v\|, \tag{2.6}$$

where $\|v\|$ is the Euclidean norm of v. The reader can verify that (2.6) reduces to (2.5) in the one-dimensional case.

Using these rules, we can model complex systems by writing the equations for all of the individual components and solving (or simulating) them as a system of equations.

A **statics** problem is one where the system components are not moving, and nothing internal to the system can cause them to move. As an example, consider the system shown in Figure 2.5. If we are told that this system is at rest, then we can write an equation that relates $m,\ g,\ k, x$, and F_k. Given any three of these parameters, we can determine the fourth. Also, given a relation between two of these parameters, we can determine a relation between the other two.

The same system can become a **dynamic** system if we are told that at least one part may be moving. For example, the mass m (at position x) may be moving, in which case it would have speed x' and acceleration x''. The possibility of a non-zero acceleration must now be factored into the equation, and it would involve $g,\ k,\ x,\ x'$, and F. We will also generally need to know the position at some given time, such as the position $x(0)$ at time 0, in order to be able to solve the resulting equation.

A more involved example is the following configuration in Figure 2.6. For this example, we can write a system of ODEs involving x_1 and for x_2, their derivatives

www.dbooks.org

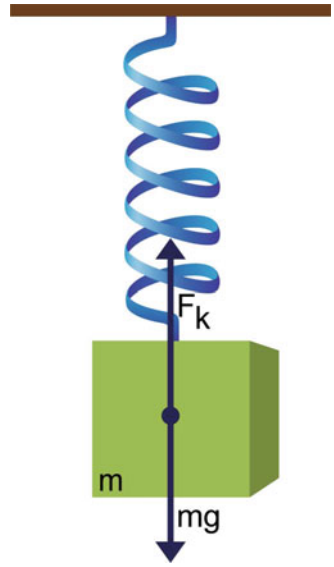


Fig. 2.5 An object with mass m is attached to a spring. The gravitational force is acting on the object but no air resistance is assumed

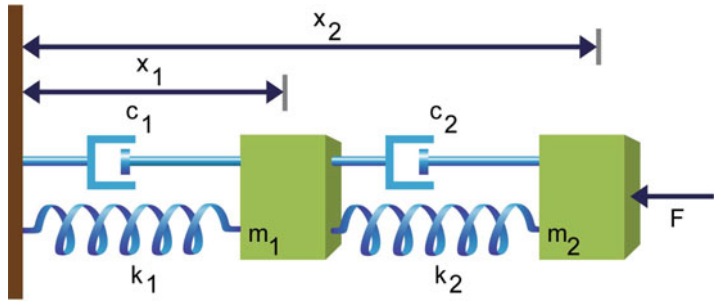


Fig. 2.6 A one-dimensional system involving two objects with masses m_1 and m_2, respectively. The objects are connected via springs and dampers

and their second order derivatives. This system of equations provides a model which we can analyze and simulate.

2.4 Working in 2D and 3D

For many physical systems, it is useful to reason about the mechanics in 3D. But occasionally the problem can help us keep things simple, and we can work in 2D or even 1D. When we need to work in 2D or 3D, the key insight required is how to factor a single force or speed into multiple forces that are relevant to the equations for

the components. In general, for such a factorization, basic trigonometric identities will be very helpful. Consider the example in Figure 2.7. In this example, the normal

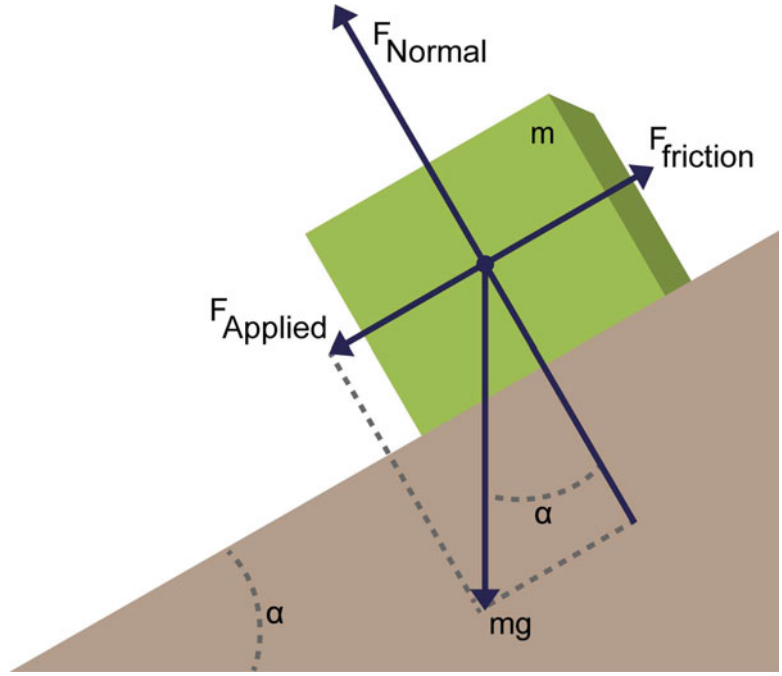


Fig. 2.7 A box with mass m is placed on a slope. A friction force given as a function of the normal force is stopping the box from sliding down. However, when the inclination, given by the angle α, is large, the friction force is not large enough to stop the box from sliding

force F_{normal} is a determined by the gravitational force mg and would have to be computed using the angle α. The value of $F_{friction}$, in turn, would be determined by the friction laws. A good source of examples of this type of analysis can be found at physicsclassroom.com.

2.5 Elements in Electrical Systems

Standard elements in electrical systems include:

Resistor This is one of the most basic elements of an electric circuit. The current I passing through this element has a direct relation to the voltage V across the resistor. This relation is known as Ohm's law, and is expressed mathematically as

$$V = IR, \tag{2.7}$$

www.dbooks.org

where R is the resistance. The diagram in Figure 2.8 illustrates schematically a situation where this relation should hold.

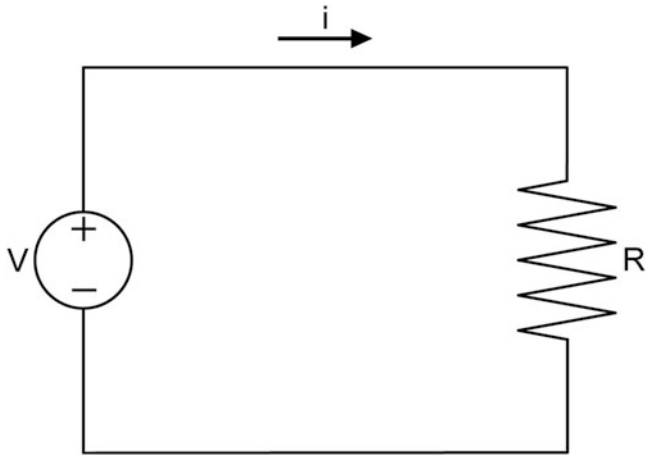


Fig. 2.8 The current I is passing through a resistor with resistance R

Capacitor This is an element where the voltage across is proportional to the integral of the current going through the element. The diagram in Figure 2.9 includes the schematic symbol for this kind of device in a circuit diagram. Alternatively, we can say that the current is proportional to the rate of change of the voltage across it, that is, $V' = dV(t)/dt$. Mathematically, this means that

$$I = CV', \tag{2.8}$$

where the ratio C is called the capacitance. The higher the capacitance, the greater the current needed to correspond to a small change in voltage.

Inductor This is an element where the rate of change in current, that is, $I' = dI(t)/dt$, is proportional to the voltage across it. That is

$$V = LI', \tag{2.9}$$

where the ratio L is called the inductance. The higher the inductance, the higher the voltage we need to get the same rate of change in current. Figure 2.10 illustrates a circuit involving an inductor.

Voltage Source A voltage source simply provides a voltage that is either fixed (a direct current or DC source) or variable, in which case it may be an alternating current (AC) source.

Current Source Like a voltage source, a current source provides a current that may be fixed or variable, depending on the type of source.

As can be seen from the remarks above, each of these elements comes with a set of equations that describes its effect on the system that it is used in. To analyze a

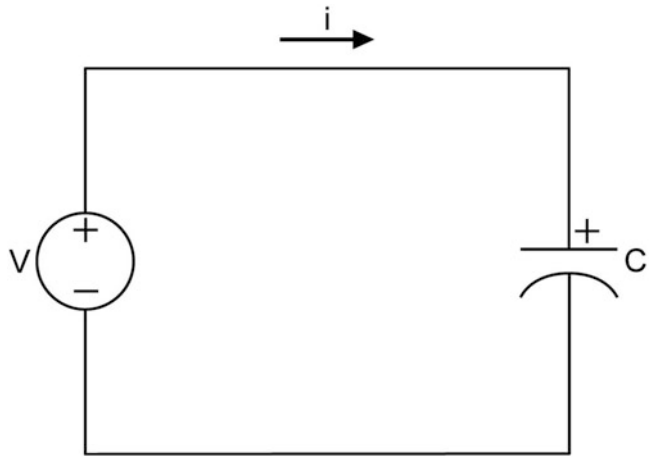


Fig. 2.9 Circuit with a capacitor

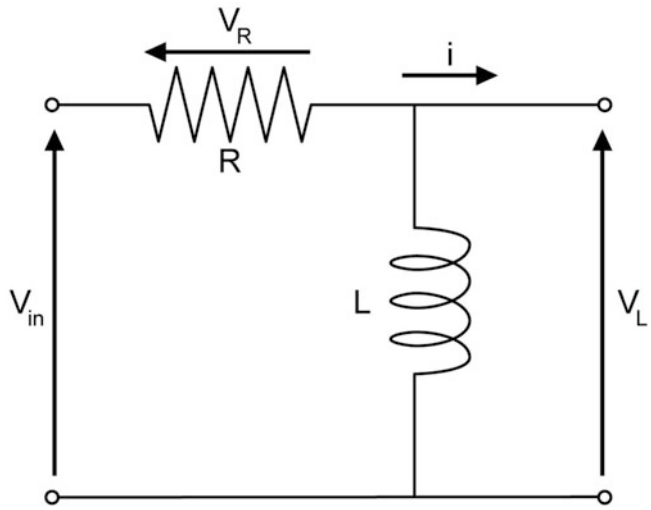


Fig. 2.10 Circuit with an inductor

given circuit, these rules are applied, along with one or more instances of the basic conservations laws for circuits, which are that:

- At any **node** (that is, region of the circuit that is connected and is not interrupted by elements), the total in-going and outgoing current must be zero.
- Around any **loop** in the circuit (that is, any path through connected components in the circuit) the total voltage differences experienced must be zero.

Several examples of this kind of analysis can be found online at this page by Erik Cheevers (see also To Probe Further below).

www.dbooks.org

2.6 The Absence or Presence of Time in a Model

In many important systems, it is possible to remove time out of the modeling process. The study of mechanical systems where time is not involved is called "statics." Electrical systems that can be analyzed without need to consider time include circuits made purely out of resistors and a constant source of voltage or current. Whether or not we need to model time depends on

1. The type of components involved in the system,
2. The state of the system, and
3. The type of input that we are applying to the system.

For example, a mechanical system made of a lever, e.g., a teeter-totter, that is in balance does not need a notion of time to be analyzed. Similarly, an electric circuit made of resistors, capacitors, and inductors, and where all currents and voltages are constant, can also be analyzed without reference to time. However, if the mechanical system is not in balance or the circuit has a voltage changing over time, then time needs to be taken into account.

2.7 Arithmetic Equations, and Linear and Non-linear Systems of Equations

When we consider problems where there is no notion of time, we generally build equations using arithmetic operations including addition, subtraction, multiplication, and division. Equations that use only addition and subtraction (or multiplication/division by a constant) are called "linear equations," and are generally relatively easy to solve by hand when the number of variables involved is small.

Example 2.1 Consider the following set of equations:
$x + y = 3$
$x - 3y = -1$
Solve this equation for x and y.

Solution 2.1 This is a linear system of equation. A basic strategy for solving such a system of equations is to convert one of the equations into a form where only one of the variables appears on the left-hand side of the equal sign. Then substitute that variable with the expression on the left-hand side in the remaining equation. The first equation can be rewritten as $x = 3 - y$ when we subtract y from both sides. When we substitute the right-hand side into the next equation we get $(3 - y) - 3y = -1$, which simplifies to $3 - 4y = -1$. By subtracting 3 from both sides we get $-4y = -4$. Dividing both sides by -4 we get $y = 1$. Substituting that into $x = 3 - y$ we get $x = 3 - 1 = 2$.

If we have equations where we multiply two different variables, or if we use functions like exponentiation, logarithms, roots, or trigonometric functions for variables or values that depend on variables, then these equations are no longer linear. Sometimes, it is possible to use a method similar to the one above to solve such equations. In general, however, it is not possible. As a result, it is often necessary to resort to iterative approximation methods to solve such equations, which can be more computationally expensive (even when done by a computer).

2.8 Where Different Numbers Come from

It is helpful to note that the kinds of operations that we use in the equations actually give rise to different classes of numbers. For example, the $\mathbb{N}$ (natural numbers), $\mathbb{Z}$ (integer numbers), $\mathbb{Q}$ (rational numbers), $\mathbb{A}$ (algebraic numbers), and $\mathbb{R}$ (real numbers) are the numbers that can express solutions to different kinds of problems. In the order they were presented, these sets are larger and larger, and incorporate more points. The set $\mathbb{N} = \{1, 2, 3, \ldots\}$ is also referred to as the counting numbers. The set $\mathbb{Z}$ contains $\mathbb{N}$ but also all numbers in $\mathbb{N}$ constructed by multiplied by -1 or 0, that is, besides $\mathbb{N}$. The rational numbers are all numbers that can be written as $\frac{a}{b}$, where $a \in \mathbb{Z}$ and $b \in \mathbb{N}$.

The algebraic numbers in the set $\mathbb{A}$ are solutions to equations like

$$x^m = n \tag{2.10}$$

and they require having inverse options like $\text{root}(m, x^m) = \text{root}(m, n)$ implies $x = \text{root}(m, n)$. The value computed by such a function is not in the set of natural numbers $\mathbb{N}$ nor in the set of rational numbers $\mathbb{Q}$, but is in a "new" set $\mathbb{A}$.

2.9 Time-Dependent and Differential Equations

Because many components have time-dependent behavior, we will often need ordinary (as opposed to partial) differential equations to describe them. Ordinary Differential Equations (ODEs) can be classified in several ways. One of the most important classifications is the distinction between linear and non-linear equations. In the case of linear ODEs, the equations are linear in the variables and the derivatives thereof. Linear ODEs usually have solutions only involving exponential or complex exponential functions. By complex exponential functions we mean exponential functions where the exponent has a complex number coefficient. According to Euler's identity, exponentials that have purely imaginary coefficients correspond to sine and cosine functions. They are commonly used in connection with electrical circuits and mechanical systems. It is very important to note that sine, cosine, and other functions occur in the *solutions* but not in the equations themselves.

www.dbooks.org

Non-linear ODEs may have solutions that do not have a closed form (analytical) representation. In such cases one usually has to rely on simulation and numerical solutions, which is commonly done for mechanical systems and electronics circuits. One could point out in this context that even though the solution is not expressible in closed form, certain aspects of it could sometimes be deduced analytically. For example one could prove that the solution will converge to a specific point as time progresses (but not exactly how without simulation). This is something that goes beyond the scope of this book.

Other types of differential equations exist apart from ODEs, such as Partial Differential Equations (PDEs) and Integral Differential Equations (IDEs), but those will also not be covered in this book. Our focus here is on ODEs with respect to time and rigid body systems, with no flexible elements.

We will always use t as the variable for time, and we differentiate with respect to time. We identify time with the real numbers $\mathbb{R}$. For derivatives with respect to time, we often see the notation dx/dt or dy/dt. Depending on the situation, we may write the same equation as:

$$dx(t)/dt = 1 \quad \text{or} \quad \frac{dx(t)}{dt} = 1. \tag{2.11}$$

Or more concisely as:

$$dx/dt = 1 \quad \text{or} \quad \frac{dx}{dt} = 1. \tag{2.12}$$

Or even more concisely as:

$$x' = 1. \tag{2.13}$$

The notation in the last equation is the same as that in Acumen. Acumen is a small modeling language and it is designed to be this way to let us learn a lot about cyber-physical systems in this book.

2.10 Prototypes of Equations (That Will Recur Throughout the Book)

The Acumen examples from Lab 1 are *prototypes* that represent important classes of subproblems that can be used to solve a bigger problem. We will see examples of these equations throughout the book.

In what follows, we discuss how these prototypes can be solved.

- $x' = 1, x(0) = x_0$.

  Solved by taking definite integral from 0 to t to both sides: : $\int_0^t x' ds = \int_0^t 1 ds$ and this implies $x(t) + k_1 = t + k_2$ which in turn implies $x(t) = t + k_2 - k_1$. Now we can use this equation and our initial assumption that $x(0) = x_0$ to deduce that $k_2 - k_1 = x(0)$. Thus $x(t) = t + x_0$.

- $x'' = 1, x'(0) = v_0, x(0) = x_0$.

  Solved by applying the same procedure as the previous equation twice. In particular this is used in Newton's law $f = ma$ where a is the second derivative of x. We assume that f is constant and consider the equation:

  $$\frac{f}{m} = x'',$$

  where $\frac{f}{m}$ is chosen to be equal to 1 for simplicity. $\int_0^t x''ds = \int_0^t 1ds$ and so $x' = t + v_0$. We can now do the same trick to solve it: $\int_0^t x'ds = \int_0^t (t + v_0)ds$, which gives us $x = t^2/2 + v_0t + x_0$. As an exercise, use this method to solve a variation of this problem where $x'' = -9.8$.

- $x' = x$.

  If we apply the same trick as in the previous examples and integrate both sides: $\int_0^t x'ds = \int_0^t xds$, that is $x(t) - x(0) = \int_0^t xds$ we cannot move forward because we need to know x to be able to integrate it. The solution, if $x(0) = 1$, is the exponential function $x(t) = e^t$, since its derivative is the function itself. The function e^t describes exponential growth, present in applications such as bank interest growth or bacteria population growth. The function e^{-t} models exponential decay, for example, radioactive isotope decay or capacity charge decay.

- $x'' = -x$.

  A solution, when $x(0) = 0$, is $\sin(t)$ (recall: $\sin'(t) = \cos(t)$ and $\cos'(t) = -\sin(t)$). Note that $\sin(t)$ is an exponential function with a complex coefficient. An example modeled by this equation is a spring mass. Please see Harmonic oscillator for a description. The equation arises from the description of the force exercised by the spring: Remember that the force created by a spring is $-kx$ where x is the distance from the equilibrium part. The equation above would arise if there is a mass $m = 1$ and a spring coefficient $k = 1$.

- $x'' = -\sin(x)$.

  $\sin(x)$ is a non-linear function, so this is an example of a non-linear ODE, where we usually resort to simulation for evaluation of system behavior. This particular equation is used to describe a pendulum pendulum.

www.dbooks.org

Key point from these examples is that $x' = 1$ and $x'' = 1$ have polynomial solutions, whereas $x' = x$ and $x'' = -x$ have (real or complex) exponential solutions.

Example 2.2 Prove that the solution of $x' = x$ is NOT a polynomial of finite order. Proof sketch: Assume that the solution is a polynomial of finite order, and the highest nominal has power n. Derive a contradiction.

Exercise 2.2 Prove that the solution to $x'' = \sin(x)$ is not an exponential function.

2.11 Remarks on the Basic Machinery for Solving Differential Equations

In many cases we know how to solve a differential equation because we know what function has that derivative. For example, consider the monomial t^{n+1}. This is a function of t that has the following derivative: $d(t^{(n+1)})/dt = (n+1)t^n$.

Example 2.3 $d(t^2)/dt = 2t$ and $d(t^{14})/dt = 14t^{13}$.

The linear property of differentiation states that if a and b are functions of time, then $d(a+b)/dt = d(a)/dt + d(b)/dt$.

Example 2.4 The properties described above help us to compute derivatives by hand. For example, we can use it to compute that $d(t^{101} + t^{52})/dt = dt^{101}/dt + dt^{52}/dt = 101t^{100} + 52t^{51}$.

Fundamental Theorem of Calculus $\int_A^B f(t)dt = F(B) - F(A)$ where dF/dt is f, (see Fundamental theorem of calculus).

Why Is the Fundamental Theorem of Calculus so Important for Us? There is a common pattern in mathematics that is worth keeping in mind as we look at solving differential equations. This pattern highlights the importance of inverse functions in helping us solve equations. For example, if we are asked to solve for x in the equation $x + m = n$, what do we do? Firstly we isolate x by applying the inverse of $+m$, which is $-m$, to both sides of the equation. This means that we convert the first equation to a new one: $x + m - m = n - m$. Then, we simplify it to get $x = n - m$, and this last equation gives us the solution to this problem. A similar thing happens with differential equations. Intuitively, the Fundamental Theorem of Calculus is important because it tells us that there is a way in which integration is essentially an inverse of differentiation. This means that it is often very useful to use integration when solving differential equations.

Note on the Exponential Function Not only does it hold that $(e^t)' = e^t$, but also $d(ke^t)/dt = kd(e^t)/dt = ke^t$.

A generalization of $x' = x$ or $x' = -x$ is $x' = kx$. This corresponds to the generalization of the exponential function to e^{kt}, for which we have $d(e^{kt})/dt = ke^{kt}$.

2.12 Chapter Highlights

1. Overview of Modeling
   (a) Basic physical phenomena
      - Physics
      - Chemical process
      - Biological processes
   (b) Characteristics of models' physical phenomena
      - Quantities are often real-valued
      - Change is often continuous
      - Usually enjoy conservation laws that are the key to modeling them!
      - The equations we get will be:
        – Either linear or non-linear
        – Either without time (statics) or involving time (dynamics)

2. Mechanical Systems (Statics)
   (a) Conservation laws: Force, Energy, Momentum
   (b) Components: Mass, gravity, surface, friction, spring, pulleys
   (c) Static Examples:
      - A single mass with gravity
      - A single mass with gravity, friction, and a lateral force
      - A pile of masses
      - A lever (teeter-totter)
   (d) Laws for different components (like springs and pulleys)
   (e) Conservation of force generalizes naturally in 1, 2, 3 dimensions

3. Electrical Systems ("Statics")
   (a) Conservation laws: Current, Voltage
   (b) Components: Resistors, capacitors, and inductances

2.13 Study Problems

1. Modify the pendulum equation $x'' = -\sin(x)$ to model air resistance on the point mass as it moves. Assume a coefficient of 1 for the term that you introduce to model air resistance. Modify the Acumen pendulum model used in the first lab (Sect. 1.9) to show the behavior resulting from this modification.
2. Consider the mechanism in Figure 2.11.

www.dbooks.org

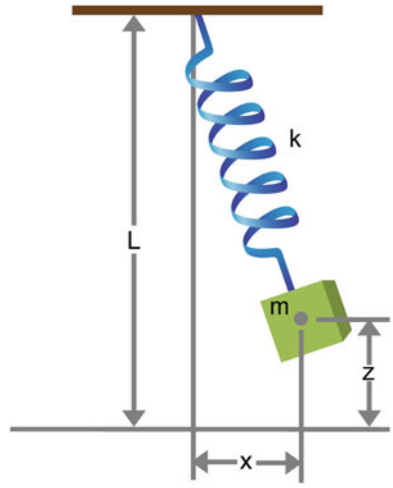


Fig. 2.11 An object with mass m is attached to a spring, which, in turn, is attached to the ceiling

Assume that all objects are sufficiently far from each other. In other words, you do not have to worry about impacts.
Assume that there is a gravitational force g pulling the mass down, and that the normal length for the spring is l_0.

(a) Assume the usual spring law, and a coefficient k. Write the equations for x'' and z'' for the system.
(b) Assume that air resistance has an effect on the object with mass m as a result of its movement. Assume a coefficient r for the effect of air resistance on this mass. Write an updated version of the equations you wrote above to reflect this effect.
(c) Instead of the usual spring force law, use the following modified law:

$$\text{force} = k(\text{length} - l_0)^3.$$

Write the equations for x' and z' for the system above.
(d) Assume that air resistance has an effect on the masses m as a result of its movement. Assume coefficient r for the effect of air resistance on this mass. Instead of the usual air resistance law, use the following modified law:

$$\text{force} = r(\text{speed})^3.$$

Write a modified version of the equations you wrote above to reflect this effect.

3. Consider the following simple system shown in Figure 2.12.

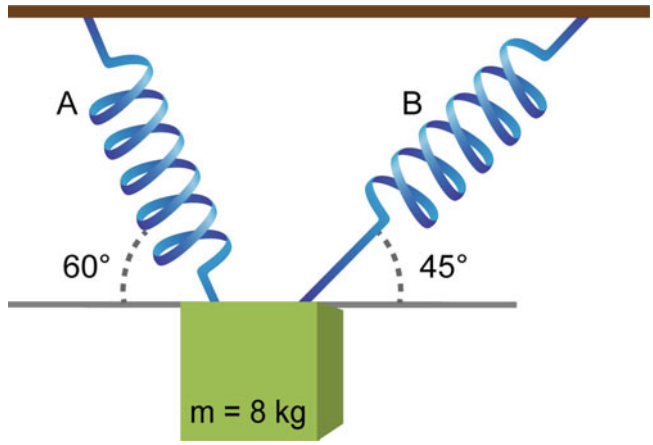


Fig. 2.12 An object with mass m is attached to two springs, which, in turn, is attached to the ceiling

The diagram depicts an 8 kg mass that is hanging from two springs. The mass is subject to the effect of gravity. Assume $g = 10\,m/s^2$. The mass has no other support than the springs (if they are removed, it falls). The springs are attached to the ceiling. The distance between the box and the ceiling is 1 m. We denote by k_A and k_B, the spring constants for spring A and spring B, respectively. The system is *static*, meaning, it is stable and there is no movement. This means that the springs are *stretched*, or extended so that they are each exerting a force.

(a) As expressions of the spring constants k_A and k_B, write the equations for the x-component and the y-component of the forces acting on the mass. Note the particular convention indicated above for axes.
(b) Determine the spring constants k_A and k_B.

4. Consider the impact of a ping pong ball with a flat floor.

(a) Assume that the ping pong ball is a point mass with position $p = (x, y, z)$. Assume that the floor has infinite mass, and is horizontal. Assume further a coefficient of restitution of 0.8, meaning that the outgoing vertical speed is 0.8 of the incoming vertical speed. Write down an expression of p' after the impact in terms of p' before the impact.
(b) Now assume that the floor can have any orientation, and that the normal unit vector to the floor is $N = (n_x, n_y, n_z)$ that is orthogonal to the surface of this floor. Note that a unit vector N has the property that $\|N\| = 1$. Write down an expression for p' after the impact in terms of p' before impact.
(c) Now assume further that the floor has mass 5 kg and the ball has mass 2 kg, and that the floor is not moving before impact. What is the speed p' after impact?

www.dbooks.org

5. Consider the following mechanism in Figure 2.13.

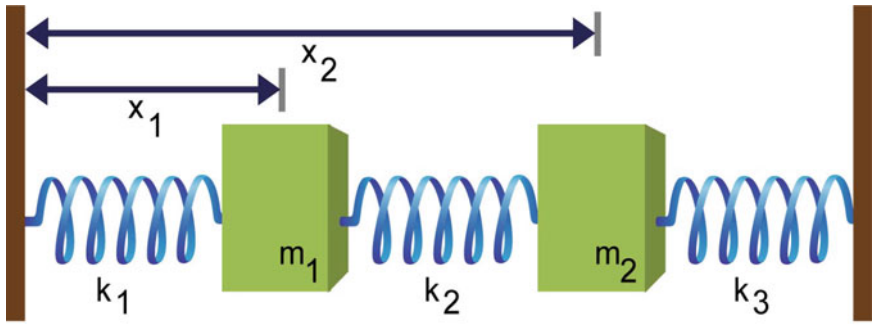


Fig. 2.13 A one-dimensional mechanism involving two objects and three springs

Let the normal (unstretched) length for the springs be zero. Let u_1 and u_2 be input forces on each mass. Let x_1 and x_2 be the positions of the two masses labeled m_1 and m_2 as measured from the left wall. Let the right wall be at position L from the left wall. Assume the width of each mass is negligible (zero). Let k_1, k_2, and k_3 be the coefficients for the three springs. Assume that all objects remain sufficiently far from each other and the two walls at all times (do not worry about impacts).

(a) Write down an expression for the total force acting on each of the two masses.
(b) Write the equations for x_1'' and x_2'', taking into account the effects of the external forces, the springs, and the inertia (mass).
(c) Assume u_1 and u_2 are zero. Write down an equation for x_1 and x_2 where the system can be without any motion (that is, the system is in static equilibrium).

2.14 Lab: Spring Bouncing and Object Creation

The purpose of this lab is to introduce us to some key modeling challenges and concepts that arise when we want to integrate both continuous and discrete dynamics. This is a good preparation for the project work and for the coming chapter.

A simple way to model a bouncing ball is that when it is away from the ground it is in free fall, and when it is touching the ground it experiences a force that we can view as the force of a spring with a high coefficient. We can call this the spring bouncing model. The following idea can be expressed as follows:

```
model Main (simulator) =

initially
x = 10, x' = 0, x'' = 0

always
if (x > 0)
then x'' = -9.8

else x'' = -9.8 - 100 * x
```

Begin this lab by sketching out the expected plots for the variables `x`, `x'`, and `x''`. Then, run this simulation and compare it to the sketch that you had made.

An important Acumen construct used in the ping pong model is object creation. Model definitions allow us to define a type of objects. For example, we can take out parts of a model that relate to a ball from the model Main, which is a special model representing the entire world that we are simulating, to a separate model. This would lead to a model such as the following:

```
model ball (x0) =

initially
x = x0, x' = 0, x'' = 0

always
if (x > 0)
then x'' = -9.8

else x'' = -9.8 - 100 * x
```

Now, we can easily create a world in which there are two different balls that start at different heights as soon as the world that we are simulating starts. This is done as follows:

```
model Main (simulator) =

initially
ball1 = create ball(10),
ball2 = create ball(20)
```

www.dbooks.org

For this lab, explore how to modify and test such a definition to: (1) how to model the presence of a damping force during the process of bouncing (this is a force that simply acts against the direction of movement), and (2) how we can introduce the effect of air resistance to a model of a moving ball.

2.15 Project: Mascot and Ping Pong Game

There are two parts to this chapter's project activities: Creating a mascot for your ping pong player and familiarizing yourself with the ping pong model.

Part 1 Design a mascot for your player using Acumen. You should be able to do this after you have finished going through all the examples in the `01_Introduction` directory of the Acumen distribution, and carrying out the exercises there. The mascot should consist only of a `Main` class and should include a `_3D` statement that creates the 3D form of your mascot. Once you have a good design you can consider animating the mascot or making it act in a short animation (maximum 10 s). Some examples of mascots and animations produced by past students can be found in the following online video.

Part 2 The Acumen distribution comes with a default ping pong model that can be used for the project. If you are using the book in the context of a course, your instructor may provide a custom made version of that model. The ping pong model is designed to illustrate:

1. Key modeling concepts that we need to know to develop almost any cyber-physical system
2. Basic path planning (at an intuitive level)
3. Basics of control
4. Dealing with basic mechanical features of robots
5. Dealing with issues such as quantization and discretization
6. Basic game theory

If you are not already familiar with the game of ping pong, browse through the article Table Tennis (ping pong) to familiarize yourself with it.

A good description of the model we will use for the project can be found in Section 6.2 of Xu Fei's Master's Thesis. Characteristics of this model include:

1. Names of included files suggest their role
2. Names also suggest a relation between these components (in terms of data flow). For example,
   (a) `Ball_Sensor` processes `Ball` data going into `Player`
   (b) `Ball_Actuator` processes signals going from `Player` to `Bat`

3. To learn the most from the physical modeling part of this model, it helps to do some derivation yourself starting from a 1D model of a bouncing ball to build up to a 3D. It is instructive to go over that model in detail to learn how vector and vector calculus operations are supported in Acumen.
   In the flying case, the aerodynamics part motivates the discussion of unit and norm of vectors, and some identities between them (such as what is the value of `unit(p)*norm(p)`, for example).
4. Going over the `Ball_Sensor` model helps in understanding sampling and how it can be modeled.

The initial model for ping pong that is used for the project can be found in the following directory:

`examples/01_CPS_Course/99_Ping_Pong/Tournament1`

in the Acumen distribution. Read and make sure that you understand all the files in this directory. The distribution also comes with default models for different stages of the project (called tournaments in the implementation). If you are using this textbook for a course, your instructor may also provide you with special editions of these models more closely matching the goals of the course you are following. Figure 2.14 depicts the way the ping pong model typically appears in Acumen. Numbers associated with each players are scores, and the bars on the side indicate remaining energy for the player. The white sphere is the ball itself, while the red and cyan dots are values predictions/estimates made by each player about where the ball will be at certain times. By modifying the default models provided you will both develop the design of the robot player and control how different aspects of the design are visualized in simulations.

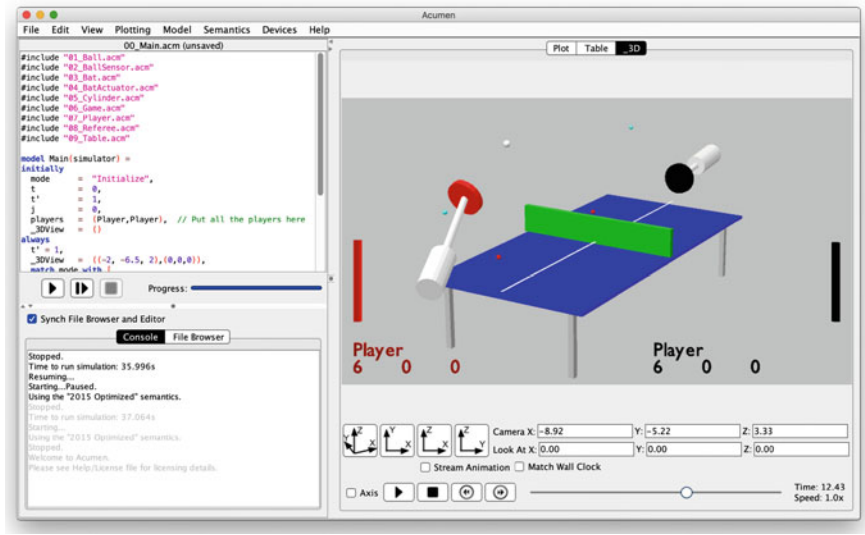


Fig. 2.14 A typical view of ping pong model in Acumen

www.dbooks.org

2.16 To Probe Further

- The examples presented at physicsclassroom.com
- The page by Erik Cheevers
- Background on mathematics, and physics
- You are also encouraged to watch this lecture on Rigid Body Dynamics
- Khan Academy has a great collection on basic physics
- To dig deeper, consult as needed these basic technical references on differentiation and integration, and on resistive circuit elements, analysis of resistive circuits, and RLC circuit analysis
- Sections 6.1 and 6.2 of this draft of David Morin's book on Classical Mechanics
- For fun, you may enjoy checking out this great illustration on the scale of the universe
- Close et al.'s textbook on modeling and analysis of dynamic systems
- The Ethicist: A Heating Problem (Short article)
- Article about 17 Equations that Changed the World
- An example of a model that won a Nobel Prize: the Hodgkin-Huxkley model
- Check out Wolfram's online Alpha tool
- Readers interested in more advanced methods for modeling mechanical systems (beyond the scope of this book) may wish to consult the article Euler-Lagrange Equation
- Much modeling and simulation aims at predicting the behavior of systems. NY Review has an interesting article on three books on prediction

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Chapter 3
Hybrid Systems

What if our models are neither continuous nor discrete? In this chapter we cover hybrid automata. Starting from a continuous setting, we introduce discrete events and look at the issues that arise in making such a transition. We analyze the issues of zero crossing and decidability; mode switching and its effect on derivatives; discrete transitions; and Zeno behavior.

3.1 Introduction

We begin by considering a classic example that builds on concepts that we are familiar with from modeling physical systems. Consider a ball that leaves the ground with a certain speed and at a certain angle, and determine where it lands. In all such high school and college physics questions, every story stops as soon as the ball hits its target. But what happens to the ball *after* it hits the ground?

A bouncing ball is a system that can be described by two different modes of behavior: falling; and bouncing or resting on the floor. How can we model a system that includes such different modes of behavior?

When the ball is **falling** we have only the effect of gravity, which has a constant $g = 9.8\,\text{N/kg}$. In that case, the ball is governed by the Newtonian law of $F = ma$ (force equals mass times acceleration). We should keep in mind that the net or "total" force on any object is the sum of all the forces applied on the object. This summation applies whether we are working in a one, two, or three dimensions. Since we have

$$\text{F}_{\text{gravity}} = ma, \tag{3.1}$$

then if the only force acting on a mass is gravity, then we also know that

$$gm = ma. \tag{3.2}$$

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_3

www.dbooks.org

By dividing both sides by m we obtain $a = g$. By substituting 9.8 for the value of g, we get (in the dimensionless setting, where "m/s^2" is ignored) that $a = 9.8$.

The position vector x is usually defined as the vector going from the floor level to the position of the ball. In this setting the acceleration goes in the opposite direction. Finally, as a differential equation in relation to the position x, the above equation simply means

$$x'' = -9.8. \tag{3.3}$$

Now let us turn to modeling the process of **bouncing**. One solution could be to imagine that, once the ball is in contact with the floor, it experiences another force that we will model as a spring force arising from contact with the ground. The higher the spring coefficient, the faster the bounce. To model this effect, we must return to our original equation and consider the entire situation. The total force on the object must still, as always, be equal to the mass times the acceleration. But the forces acting on the ball are now slightly different:

$$\mathrm{F}_{\text{gravity}} + \mathrm{F}_{\text{floor}} = ma. \tag{3.4}$$

The gravity force and the acceleration are the same as above; but $\mathrm{F}_{\text{floor}}$ will be determined by Hooke's law, generating a force $-kx$ where we can take k to be a large coefficient. Thus, Equation (3.4) can be expressed as

$$-gm + k(-x) = am.$$

Now, when solving for a we get

$$a = (-gm - kx)/m.$$

Now we notice that we need to specify the coefficient k. Interestingly, at this point we may also need to specify the mass (why did we not need to worry about the precise mass of the ball before?). However, we can avoid the need to specify both separately, because the equation can be further simplified to

$$a \;=\; -gm/m \;-(k/m)x$$

which is can be written as

$$a = \; -g - (k/m)x.$$

We can simply consider a situation where the ratio between k and m is high (such as 100); then we have all the information that we need for a model of a ball that can undergo simple elastic bouncing when it hits the floor. With this, we can replace the acceleration a with with x'', and k/m with 100, to get the differential equation:

$$x'' = -9.8 - 100x.$$

We can use Hooke's law to model $\mathrm{F}_{\text{floor}}$, i.e., to model the floor as a spring.

Now all that remains is the question of how to specify the switch between being in flight and being in the bouncing state. This can be described as follows:

$$\text{if } (x > 0)\ x'' = -9.8 \text{ else } x'' = -9.8 - 100x.$$

Once we introduce if-statements into our models we are no longer working with simple differential equations. Rather, we have moved into a world where our models are describing different behaviors when the system is under different *modes*. We can think of a mode as the domain of validity of certain conditions (such as $x \geq 0$ or $x \leq 0$ in the above example). More generally, we can choose to define the notion of a mode as an explicitly named state (such as "Falling" or "Bouncing"), which we treat as an explicit part of our model.

Exercise 3.1 Use Acumen to find out the effect of increasing the ratio k/m. For example, try out the simulation with different values for the spring coefficient to validate the model. What happens when you use 1000, 10,000, or 15,000 instead 100? What happens when you use 25,000 or 50,000?

3.2 Hybrid Automata

Hybrid systems are systems of mathematical equations that combine both continuous and discrete dynamics. Often, they are formalized as a kind of extension to finite state machines called hybrid automata. For our purposes in this book it suffices to have an intuitive understanding of these notions. To get started, to remind ourselves of finite state machines (or finite automata), let us take the example of a simple traffic light.

Example 3.1 Mathematically speaking, we may want to model a traffic light as being, at any point in time, in one of the three states. We can think of a state as simply one of the three constants in the set {Red, Green, Yellow}. Further, we may want to model the *behavior* of the traffic light machine by a set of rules that determine the allowable ways in which the state of the traffic light can change from one instance to another. Rules in the case of a finite state machine are simply pairs of states written as "$s1 \Rightarrow s2$." The first state is the one we are in and the second state is the one we can go to next. So, a typical traffic light can be modeled with the following rules:

- Rule 1: Red $\Rightarrow$ Green.
- Rule 2: Green $\Rightarrow$ Yellow.
- Rule 3: Yellow $\Rightarrow$ Red.

This is a purely *discrete* model. In a synchronous model of finite state machines, we can require that transitions only happen at previously determined "clock ticks."

The notion of a *trace* of a finite state machine is simply a sequence of allowable state transitions that a finite state machine can undergo. For example, the following sequences are valid traces for the machine we described above: Red $\Rightarrow$ Green $\Rightarrow$ Yellow, as well as the trace Green $\Rightarrow$ Yellow $\Rightarrow$ Red $\Rightarrow$ Green. However, the sequence

www.dbooks.org

Green ⇒ Yellow ⇒ Green is not a valid trace because we have no rule that allows the transition Yellow ⇒ Green. Similarly, it should also be noted that in this system there are no transitions from one state to itself. So, Red ⇒ Green ⇒ Green is not a valid trace either.

Now let us consider another example.

Example 3.2 Consider another finite state machine that represents whether an air-conditioning system is either cooling or on standby. In this case, we can have two simple states {Cool, Wait}. As you know, an air-conditioning system cools the room for a while, then switches back to the waiting state, then reverts to cooling, and so on. So, the rules for the possible transitions in such a system are quite simple:

- Rule 1: Cool ⇒ Wait.
- Rule 2: Wait ⇒ Cool.

During the cooling process, the system pumps heat out of a room at a constant rate, and the temperature goes down. In addition to the standard finite state machine model of this system, we may also want to build more details into our model. In particular, we might want to specify something about the temperature of the room explicitly; we might also want to specify the rate at which the room is gaining heat from the outside environment, and the rate at which the air-conditioning system is removing this heat when it is in the cooling state. Such information cannot be included in a traditional finite state machine model. However, we can extend the basic model and arrive at what is known as a *hybrid automaton*. This is where we can add information to each state about some continuous variables that can evolve according to certain rules over time (which itself is modeled as a dense time). For this example, all we have to do is the following: introduce a real-valued variable T to describe the current temperature of the room, and then add the following two rules about what happens to this variable when the system is in each of the two states.

- While in state Cool: $T' = -1°$/s.
- While in state Wait: $T' = 0.1°$/s.

In each of the two states there is a dynamical equation describing the rate of change of the temperature. Such equations can be much more involved than the ones considered here, and could, for example, be non-linear. However, what is important here is that this model specifies clearly what happens to the temperature in the room (in terms of the rate at which it changes) when it is in each of these two states. In addition, we may want to go back to our rules and specify how the *thermostat* (the controller for the air-conditioning unit) determines when it should change from one state to another. We can do this by creating an extended set of transition rules as follows:

- Rule 1: Cool ⇒ Wait is a transition that must occur when T < 20.
- Rule 2: Wait ⇒ Cool is a transition that must occur when T > 25.

Note that we use different temperatures to switch, to avoid alternating too quickly and potentially damaging our air-conditioning system.

Exercise 3.2 Use Acumen's strings and case-statements to express the example system above as an Acumen model. Simulate the system with a starting value for Temperature as 23°.

3.3 Reset Maps

Now we return to our bouncing ball example. In addition to specifying rules for how continuous variables can evolve inside a state, we can also perform a discrete change to a variable as we transition from one state to another. This allows us to perform instantaneous changes in direction without having to use, for example, a spring with a high stiffness coefficient. However it requires a further refinement to our notion of hybrid automata, namely, the introduction of *reset maps*. This extension simply allows us to specify that, as a transition occurs, we want to reset certain variables in our system to new values that are better suited to model the state of our system. Now we can model our bouncing ball example as follows:

- States = {Flying}.
- While in Flying: $x'' = -9.8$.
- Transition 1: Flying $\Rightarrow$ Flying, if $x = 0$ and $x' < 0$ then $x' \Leftarrow -0.9\ x'$.

Note that in this system we only have a single state, "Flying." The second bullet specifies the dynamics while in this state, namely, falling with a constant, negative acceleration. For this system, one state is actually enough because the work needed to model bouncing can be done in the transition from this state to itself, and in particular, in the reset map applied in the transition from this state to itself. The reset map is what is described by the part that says $x' \Leftarrow -0.9x'$, and it means that we want the x' value (which is the speed of the bouncing ball) to be reset to a new value after the transition. That value will be the result of multiplying the value of x' before the transition by 0.9. Furthermore, we will switch the sign of the speed, thus changing its direction, so that the ball that was falling down before the transition will be going upwards after the transition. Note that, as a result of the sudden change to the speed that occurs at that transition point, the acceleration will not be a well-defined notion at that instance.[1]

Exercise 3.3 Write out such a model in Acumen, and simulate the system starting from a height of 10 and an initial speed of 0. How well does your model work? Make a note of any unexpected behavior that your simulation might exhibit.

A finite state machine or an automata is *deterministic* if its behavior is always uniquely defined when it is in any given state. If this is not the case, then the system is said to be *non-deterministic*. Most simulation tools only simulate deterministic systems.

[1] In situations like this it may be possible to use more advanced mathematical notions such as impulse functions (or Dirac delta function), but this discussion is outside the scope of this book.

www.dbooks.org

3.4 Zero-Crossing

Using precise tests (such as $x = 0$) for transitioning from one state to another is generally quite challenging for simulation tools built on traditional numerical methods. This is because real numbers are generally represented in computers as floating points, and functions that change over time are represented as a sequence of floating point values defined at specific points in time, represented also by floating point values. For our purposes in this book, and to keep things simple, we will try to always express our models with more robust conditions such as $x \leq 0$ even though we really do not intend to model the possibility of x being less than 0.

3.5 Zeno Behavior

Zeno's paradox is a phenomenon that can happen in a hybrid system (a system that exhibits both continuous and discrete dynamic behavior). The hybrid automaton model of the bouncing ball exhibits this phenomenon. In particular, the consecutive bounces form a geometric series that can be shown to end in a finite time (which we call the Zeno point), even if the number of bounces that takes place beyond that point is unbounded.

To understand what is going on here for yourself, consider the situation where the ball is at height zero but is moving up. Calculate the time (based on the initial upward speed) until it hits the ground again. What is that ratio between the speed and the time it takes to hit the ground? Now note that there is a well-defined ratio between the upward speed at the start of this problem, and the upward speed of the ball after the bounce. Convince yourself that this ratio also determines a ratio between the current jump and the next jump. Convince yourself further that this ratio will be the same between any two consecutive jumps. Write out the formula for the time at which the ball will stop jumping, giving some initial parameters.

3.6 Modeling Elastic Collision

***Note**: While this topic is discussed here only briefly, it is an important example of discontinuity in physical models (and therefore an example of why we need hybrid systems to model physical systems). Also, it is important for understanding the ping pong model.*

Collisions in basic mechanics are a simple class of problems involving time. Their characteristic is that there is a discontinuity in one of the derivatives. Consider the situation in Figure 3.1, where we have two co-linear masses that experience a collision where energy and momentum are conserved. How do we determine the speeds after the collision?

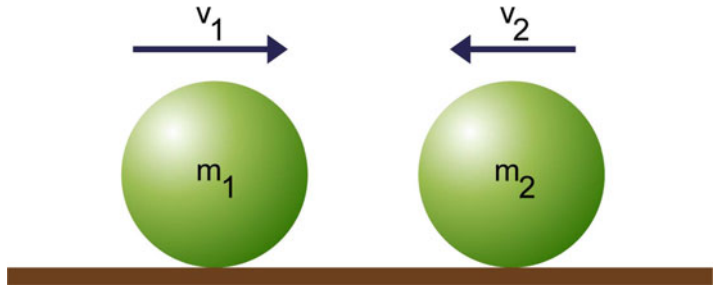


Fig. 3.1 Two balls with masses m_1 and m_2 about to collide

Let the velocity of the two balls be u_1 and u_2 before the collision and v_1 and v_2 after the collision, respectively. Conservation of momentum means that

$$m_1u_1 + m_2u_2 = m_1v_1 + m_2v_2, \tag{3.5}$$

whereas conservation of energy means that

$$\frac{m_1u_1^2}{2} + \frac{m_2u_2^2}{2} = \frac{m_1v_1^2}{2} + \frac{m_2v_2^2}{2}. \tag{3.6}$$

By using (3.5) and (3.6) together we can solve for v_1 and v_2 in terms of u_1 and u_2. The solution is

$$v_1 = \frac{(m_1 - m_2)u_1 + 2m_2u_2}{m_1 + m_2},$$
$$v_2 = \frac{(m_2 - m_1)u_2 + 2m_1u_1}{m_1 + m_2}.$$

This tells us the speeds for both objects after the collision. When kinetic energy is conserved, we says that the collision is *elastic*, and *inelastic* when it is not conserved. The speed of each particle after an inelastic collision can be determined in the same way as above, but with the energy equation modified appropriately. Often, energy loss is specified in terms of a *coefficient of restitution*, which is the ratio between relative speeds before and after the collision.

The *relative speed* between two objects is the speed of the first measured from the second and vice versa. If two cars travel in the same direction at 100 and 120 km/h, respectively, the relative speed will be 20 km/h; if the two cars travel at the same speeds but at opposite directions, the relative speed will be 220 km/h.

www.dbooks.org

3.7 Chapter Highlights

1. Hybrid Systems
   (a) Mix continuous and discrete systems
      - Area of much of the research in formal analysis and verification of CPS today
   (b) Provide a natural model of cyber-physical system
   (c) Are also a more natural model of purely physical systems
      - Impacts and discontinuities are naturally modeled as discrete events
   (d) Cyber systems have not only discrete but also continuous aspects
      - Time, energy, time-to-failure, radiation, relativistic effects

2. Finite State Machines
   (a) Traffic light
   (b) States: Red, Green, and Yellow
   (c) Transitions
      - Untimed
      - Adding a timer and a reset map

3. Thermostat Example
   (a) States: Heating, Cooling
   (b) Equations: Heat equations
   (c) Guards: Leave a gap in between

4. Bouncing Ball Example
   (a) States: Falling, and Bouncing?
   (b) Equation: Falling
   (c) Guards: Hitting Zero (with refinement)
   (d) Zeno-behavior?

5. Computing the Zeno point for a Bouncing Ball
   (a) Equation for flight
   (b) Time in flight
   (c) Flight in terms of max height
   (d) Time in terms of coefficient
   (e) The limit of a series

3.8 Avoid Common Mistakes

These are remarks intended to help you avoid some common points of confusion:

- A state in a finite state machine has no memory (or additional state) inside it. When you are designing or specifying a finite state machine, if it looks like you need one with "memory," split it into multiple states.
- A hybrid system is not just a finite state machine. So, for example, the preceding remark does not apply
- Non-deterministic is not the same as probabilistic. Probabilistic systems require much stronger assumptions about frequency in the longer term (probabilities or probability distributions). Non-determinism just means that you do not know exactly what the behavior will be (although you know exactly what set of behaviors is possible).
- Deterministic, non-deterministic, and probabilistic models are all mathematical objects.

3.9 Study Problems

1. Model the example illustrated in Figure 11 in Branicky's paper (also mentioned below in the To Probe Further list) in Acumen. Include the following in your completed assignment: (a) your complete Acumen model, (b) the plot figure, and (c) an explanation of why Acumen cannot correctly simulate the system beyond the 4-s point.
2. Model the two systems illustrated in Figure 15 in Branicky's paper in Acumen. Assume that $f_0(x) = 10 - x$ and $f_1(x) = 40 - x$. A complete solution would include: (a) your complete Acumen model, (b) the plot figures (for both systems), and (c) an analysis in your own words of how well this simulation supports the point made in the paper about hysteresis.
3. Consider the situation where you are designing a futuristic traffic light.

   (a) Represent each move by an output of the name of the color ("Red," "Orange," "Blue," or "Green"). Draw a state machine that shows the number of states needed for a traffic light that outputs a signal that carries **red**, then **orange**, then **red**, and then goes to **blue,** then **red**, and repeats this process.

   (b) Assume that the light stays in the **red** and **green** states for 60 s, and in all the other states only for 10 s. Write an Acumen object class My_Light that models this functionality. The only required field in this object is a signal my_choice which can be "R," "O," "B," or "G" depending on the first letter of the color. Make sure that the object is self-contained and do not assume any inputs from the outside.

www.dbooks.org

4. Compute the Zeno point for the basic bouncing ball example when the ball is dropped from a height of 10 m, taking gravity at 9.8, and a coefficient of restitution of 90%.
5. Consider the situation where you are designing a paper/rock/scissors player.
   (a) Represent each move by an output of the first letter ("P," "R," or "S"). Draw a state machine that shows the number of states needed for a simple player that repeatedly outputs a signal that carries **paper**, then **rock**, then **paper**, then **scissors**, and then repeats this sequence.
   (b) Assume that the machine must output each move for 0.75 s. Write an Acumen object class My_PRS that models this functionality. The only required field in this object is a signal my_choice which can be "P," "R," or "S." Make sure the object is self-contained, and do not assume any inputs from the outside.
6. Consider the situation where two cars are about to collide, and their speeds and masses are depicted in Figure 3.2.

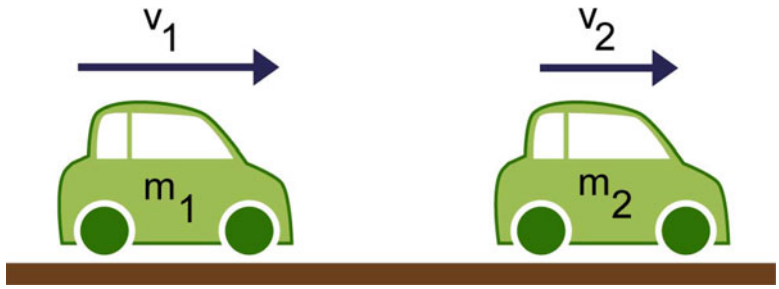


Fig. 3.2 Two cars about to collide

   (a) Assume a coefficient of restitution of C. Write a formula for w_1 and w_2, the speeds after impact, measured in the same direction.
   (b) If both masses are the same ($m_1 = m_2$) and m_2 is static ($v_2 = 0$), by what fraction does the speed w_2 increase if the speed v_1 is increased by 10%? If the answer is not a simple fraction, you may write a formula.
   (c) If m_1 is half the mass of m_2 and m_2 is static ($v_2 = 0$), by what fraction does the speed w_2 increase if the speed v_1 is increased by 10%?
7. Consider Figure 3.3 indicating the "before collision" (above) and "after collision" (below) speeds for a collision between two masses (represented as cars).

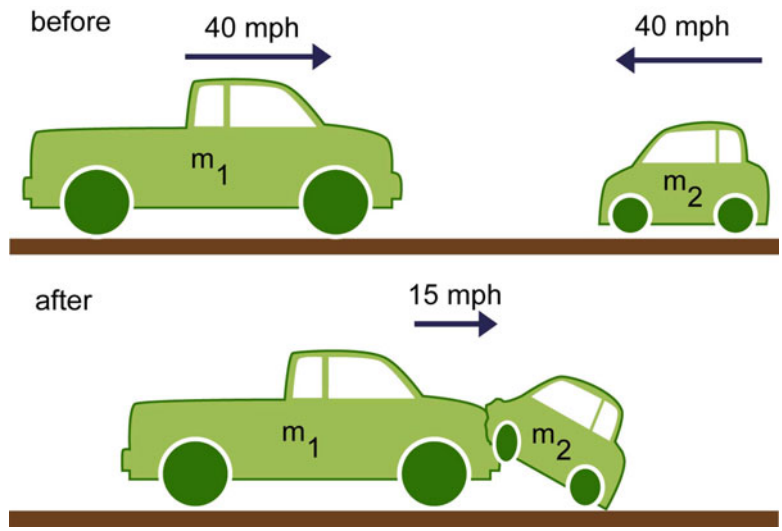


Fig. 3.3 Two cars, before and after collision (Source: scienceblogs.com)

(a) Write down the equation for conservation of momentum. This is the equation that relates the "before" and "after" speeds.
(b) Calculate the coefficient of restitution "c"'in this collision.
(c) Assume that $m_2 = 100$. What is the energy lost in this collision?
(d) Assume that $m_2 = 100$. How much is m_1?

8. Consider the diagram in Figure 3.4 depicting the "before" and "after" situation for a collision between two masses. The first object has mass m and the second has twice that mass. The second object is static before collision, and both objects are attached after the collision.

(a) Calculate the coefficient of restitution c in this collision.
(b) Write down the conservation of momentum equation relating "before" and "after" speeds.
(c) What speed must v_1 be in order to have v' equal to 1000?
(d) Assume that $m = 1000$. What is the energy lost in the collision of part c?

9. Consider the diagram in Figure 3.5. This type of diagram is called a state diagram, and is widely used to express finite state machine models in an intuitive manner. According to this diagram, the initial state is S_1 because it has an arrow that has no explicit source, possible transitions are indicated by arrows going between states (sometimes the same state), and whenever there is a transition the digit indicated on the arrow is assigned to the variable Output.

www.dbooks.org

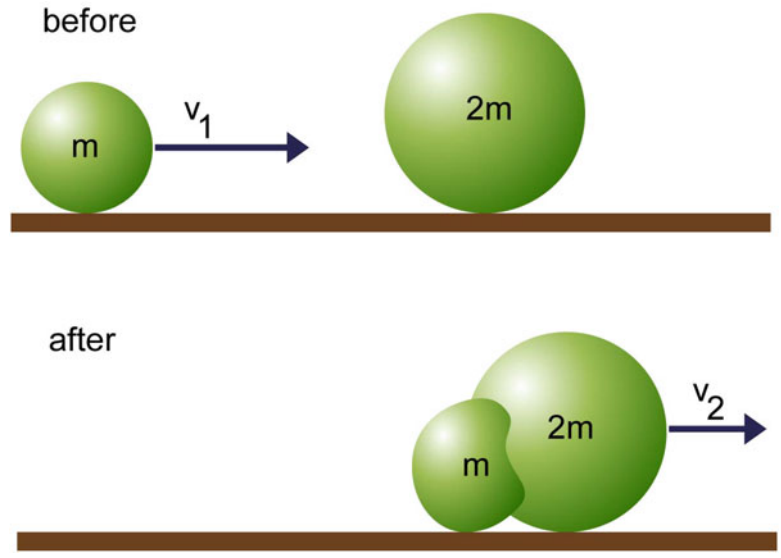


Fig. 3.4 Two spheres, before and after an inelastic collision (Source: scienceblogs.com)

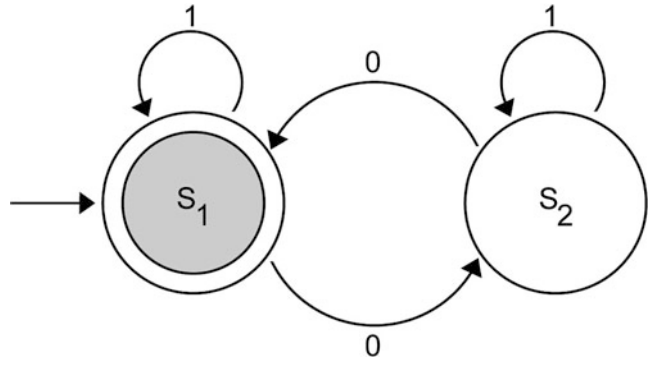


Fig. 3.5 A finite state machine with two states, and initiated at state 1

(a) Starting from the initial state, what state is the machine in after it has assigned to Output the values 0, 0, 1, 0 in that order? Write out the state of the machine after each of these four outputs individually.
(b) Assume that the system has an input called Input, and that both states take the 0 transition if Input is less than or equal to 5, and take the 1 transition if Input is greater than five. Assume further that the system makes a transition decisions and actions together, and does so every 1 s. Describe this situation using the Acumen language in one model that has Input and Output as parameters.

3.10 Lab: Discrete Bouncing

The purpose of this lab is to introduce you to hybrid-systems modeling constructs and to connect the ideas discussed in this chapter to project activities.

Taking as a starting point the models you developed for the previous lab, compare and relate those models to the following model sketch:

```
model ball (x0) =

initially
x = x0, x' = 0, x'' = 0

always
if (x > 0)
then x'' = -9.8
 // - 0.1 * x' * abs(x') // Drag

else  if (x' < 0)
then  x'' = -9.8 - 100 * x + 10
 // Another state split to enable damping
 // during bounce

else x'' = -9.8 - 100 * x - 10

model Main (simulator) =

initially
ball1 = create ball(10),
ball2 = create ball(20)
```

Next, note that bouncing in the original model worked by pretending there is a spring that gets activated when the ball is below ground level. Consider the possibility of creating a model where the bounce occurs instantaneously. Build and test a model that works in this way.

Once you are done with this exercise, compare your model to the following one.

```
model BB()= // Basic bouncing ball

initially
x = 5, x' = 0, x'' = 0

always
if (x <= 0) && (x' < 0)
then x'+ = - 0.5 * x'

else x'' = -9.8
```

www.dbooks.org

```
model Main(simulator)=

initially
b = create BB()
```

The model above has a curious feature. After the ball "reaches zero height," it starts to slip down below level zero. That is because we are actually allowing gravity to be in effect in the case that x< = 0 and x' > 0. This creates a cycle of small computational steps where the ball starts below the ground with a positive speed but is subjected to gravity, at the end of the cycle it has a small negative speed and has lost height, and impact condition is detected and the speed is set to positive half again, and the process repeats, but the ball is still losing height.

A better model is the following one, where we ensure that when the ball is below the ground but is on the rise it maintains this speed and is not subject to any additional forces:

```
model BB()= // Basic bouncing ball

initially
x = 5, x' = 0, x'' = 0

always
if (x > 0) then x'' = -9.8

else if x'<0
then x'+ = -0.5 * x'

else x'' = 0

model Main(simulator)=

initially
b = create BB ()
```

Compare your model to this one as well, and make note of the points that you found required extra care to be able to model instantaneous bouncing when you tried to build it yourself. While arriving at a simple and clear model of an instantaneously bouncing ball is challenging, it is representative of the frequently occurring situations where a dynamic within a continuous domain reaches a well-defined boundary that it is being pushed against, and where the dynamic of the boundary is to push back the object into the domain.

3.11 Project: Speed-Based Player for Ping Pong Robot

This chapter's project activity is to develop a ping pong player that can outperform the other player, and at the very least, the default player. In this activity, your model controls the player by sending a speed signal to move the bat. Be careful, because the larger the speed you use, the faster you will run out of energy! Basically, the default player works by computing a bat speed and predicting two key points on the future trajectory of the ball. The first point (p1) is where the ball will hit the table. This is calculated by using the velocity of the ball. The second point (p2) is the highest point in the air that the ball will reach after it bounces on the table. This is calculated from the predicted speed after bouncing. Then, the bat moves to hit the ball at the highest point (pH) of the ball after it has bounced. It is important to note that the bat loses energy (maxE) whenever it moves and hits the ball, and that each player starts with a fixed energy level. Your modified player must consume less energy than the default player and also predict the second point better than the default player. You will need to develop your own strategy to make the bat move less or improve the velocity of the bat.

If you are using this textbook as part of a course, be sure to check with your instructor about whether there is a special edition of the model you are expected to use.

An important skill that good scientists and engineers need in order to complete projects faster and better is debugging. In essence, you need this skill to determine how to go from a system that is not quite doing what you want to one that is. The key to successful debugging is to work systematically to isolate and localize problems. Scientists and engineers that solve real problems seem to apply this skill extensively. Check out this article on this topic.

Designing any interesting system also involves accumulation of a lot of knowledge. Our memory is only one way of collecting knowledge, and it often does not work as well as we think it does. Documenting the result of each of your project activities is a good way to organize the knowledge that you accumulate. Therefore, think of them not only as something you write for the instructor, but also for yourself, and review them constantly during the process. Another very powerful way of accumulating knowledge is test cases. Build your own test cases and use them to automatically test your player as you are developing it. You can develop test cases by making alternative players that play with different strategies.

In future project activities, make sure that your player avoids each of the pitfalls that you discovered during this activity. We recommend that you share the players that you have developed among yourselves, as this will help you improve player development for the next activity, and increase your chances of winning at the finals.

www.dbooks.org

3.12 To Probe Further

- Articles on Momentum and Coefficient of Restitution.
- Background on engineering, finite state machines, hybrid systems, and theory of computation.
- Online hybrid systems textbook by Lygeros, Tomlin, and Sastry.
- Branicky's introduction to hybrid systems.
- Very cool video by VSause about Zeno behavior and other paradoxes.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Chapter 4
Control Theory

How can we get the output of a system to behave in a certain way if we only have direct control on the input? In this chapter we cover error, feedback (negative and positive), and stability. We look at recognizing these concepts in static systems (op-amp) and in dynamic systems. We also cover PID controller design as a very basic example of how controllers are designed. Finally, we discuss the effect of implementing a controller on a digital computer and the effects of using finite representations of values (N bits to represent values) and time (being able to sample or actuate only at clock ticks).

4.1 Introduction

The function of virtually every cyber-physical system involves *control* in the sense of bringing some variable quantity to have a certain desired value. For example, we may want a car to maintain a certain speed, a ship to maintain a certain bearing, or a plane to maintain a certain altitude. The theory of control concerns itself with such problems.

Intuitively, a system is controlled by determining how certain inputs should be varied to achieve a certain behavior in outputs. In the simple examples above we have single-value and single-parameter goals. In practice, control problems can involve achieving highly sophisticated dynamic behavior in several different dimensions simultaneously. That said, many of the most fundamental principles of control can be explained and illustrated with single-value, single-parameter examples.

When discussing control, it is customary to refer to the system being controlled as the "Plant" and the system providing the inputs needed to achieve the desired output as the "Controller." Block diagrams such as the one in Figure 4.1 are typically used to describe the relation between these two systems.

If our goal is simply to achieve a single-value output, and the operation of the plant is fully understood to us, then we can achieve this result with a controller

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_4

www.dbooks.org

Fig. 4.1 Block diagram illustrating relation between plant and controller

with no inputs (such as the one depicted in Figure 4.1). In general, however, we will want to devise a controller that brings the plant to produce different output values depending on the value of another input that we provide. For that purpose, we will want to have a controller that is parameterized by our goal for the plant output. This more general situation can be depicted as in Figure 4.2.

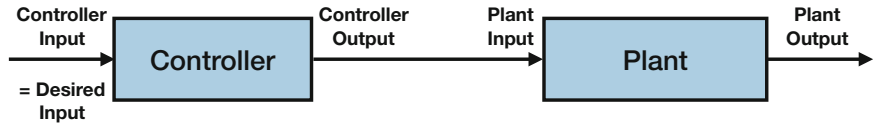


Fig. 4.2 Block diagram with controller parameterized for control objective

It is instructive here to note the following: if the behavior of the plant can be viewed as a mathematical function, then this problem can be solved by a controller that is the inverse of the plant function. Of course, this also means that the problem can only be solved if the behavior of the plant is itself an *invertible* function. A further and more subtle difficulty is that, in order to construct the inverse, we would also have to know the exact plant function in advance (meaning, at the time of designing the controller). In general, we hardly ever have such knowledge. Most systems that we need to control have parameters and components that vary slightly from one instance to the next due to a variety of reasons. These can include production process, temperature effects, age effects, environmental effects, as well as many others.

4.2 Feedback Control

In practice, almost any real controller will have imperfections. The degree of imperfection in the *operation* of the controller at any time instant can be quantified by the difference between the desired plant output and the actual plant output. This difference will be called the *error*. Note that by doing so we are thinking of the controller input as telling us the desired output. This is a common viewpoint when we think of a system as a *control system*.

Rather than simply trying to construct the inverse (which is difficult even in static systems), the error can be used directly to construct a much simpler controller that can be very effective in practice. This is achieved by the powerful idea of *feedback*, whereby the output of the system is fed back as an input (either of the entire system or, in the context of control, of just the controller). Once we allow ourselves to use the output of the system as an input to the control process, the error is easily

computed by a simple component that subtracts the value of the plant output from the desired plant input. In general, we can imagine a situation such as the one shown in Figure 4.3.

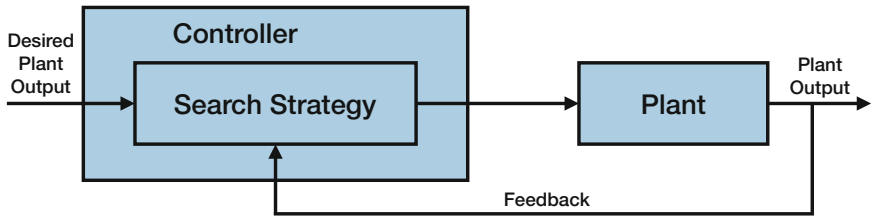


Fig. 4.3 Block diagram illustrating control with feedback and search strategy

A simple strategy can consist of the following: firstly we can "try" a value for plant input; if the error is positive, we increase that value; if the error is negative, we decrease that value; and we can stop when we get close enough.

While this control strategy is helpful to get a simple intuition for how feedback can be useful for controlling a system, it is not necessarily the easiest to analyze formally. For this reason, we will first look more closely at a simpler strategy for designing a controller.

4.3 Proportional Feedback Control

A controller can simply consist of a unit that takes two inputs, computes the error, and then multiplies this error by a certain factor to generate the input that is fed into the plant. This situation can be depicted as shown in Figure 4.4.

Despite the seeming simplicity of a controller constructed like this, the idea of using feedback in this manner is powerful, and has many useful theoretical and practical benefits, in addition to being relatively easy to construct. First, however, we need to be certain that such a controller would actually help us achieve our goal of computing a plant input that would bring the plant output to have the desired value.

Example 4.1: Feedback Control of a Wire Plant Consider the situation where our plant is simply a wire with no interesting behavior. It would be good to know how well the idea feedback control works in this setting. Let us further consider a multiplier gain of G. For brevity, let us also use the following variables to denote the various values in the system as follows:

- e is error,
- x is plant input,
- y is plant output, and
- z is desired plant output.

The equations that govern the behavior of the system are:

www.dbooks.org

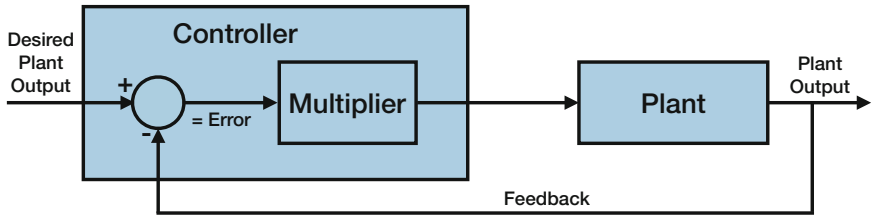


Fig. 4.4 Block diagram illustrating control with feedback and gain (multiplication)

1. the error e is $e = z - y$,
2. the plant is a simple wire, so, $y = x$, and
3. the plant output is the controller output, so, $x = Ge$.

To understand the behavior of this type of controller we have to understand the combined effect of the equations above. If we substitute e from 1 into 3 we get

$$x = G(z - y).$$

Similarly, substituting that value of x into Equation (4.2) we get

$$y = G(z - y).$$

Starting from this equation we can derive an expression for y in terms of z as follows: first we note that the equation implies that

$$y = Gz - Gy,$$

by distributivity of multiplication across subtraction. From this we can derive that

$$y + Gy = Gz$$

by adding Gy to both sides. By distributivity of multiplication on addition we get

$$(1 + G)y = Gz.$$

If we assume that G is positive, then $1 + G$ is non-zero, so, we can divide both sides by $(1 + G)$ to get

$$y = \frac{G}{G + 1} z. \tag{4.1}$$

Now we can consider what happens to the relation in (4.1) as we change the value of G. If G is 0, then we have $y = 0$. This is clearly a very poor controller. But if G is 1, then $y = (1/2)z$. This is by no means a perfect controller, as the output is always only half what we want it to be. But it is interesting that this simple construction causes y to behave in an interesting manner as z varies. The relation of the overall gain with the gain G is depicted in Figure 4.5.

If G is 99, then $y = 0.99z$. This suggests that, as the gain gets larger, this controller causes y to track z more closely. Obviously, 99% of a value is not the same as the

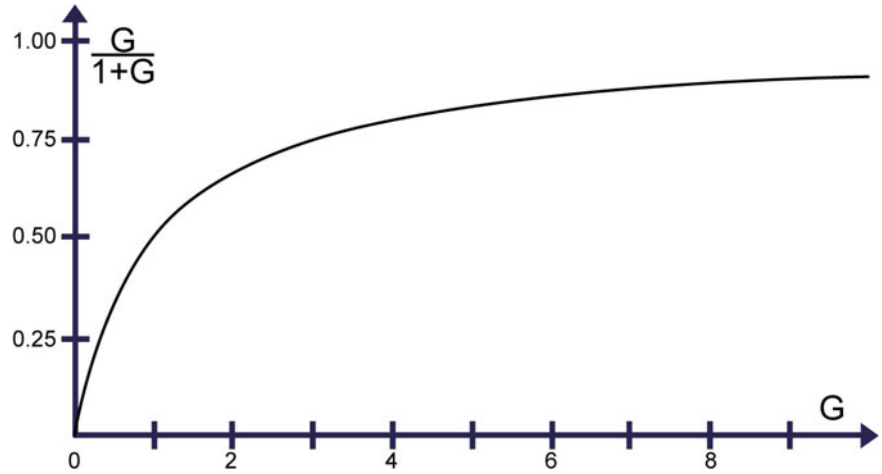


Fig. 4.5 Relation between the total gain $\frac{G}{1+G}$ and gain G

value itself, but it is pretty close, and this is an impressive result to get from such a simple controller. Finally, note that, as the gain is increased arbitrarily, this controller can make the output value arbitrarily close to the desired value.

Exercise 4.1 Derive a formula for determining G given the desired overall gain of the feedback system described in Example 4.1.

4.4 Operational Amplifiers

It is instructive to consider the effect of this controller on a completely passive plant that consists entirely of a wire from its input directly to its output. But we may wonder if we could ever need to control such a simple system in practice. In particular, why is using such a construction better than just wiring the controller's input directly to the plant input? As it turns out, this is a common construction in electronics that is known as the voltage follower circuit, and it is often depicted diagrammatically as in Figure 4.6.

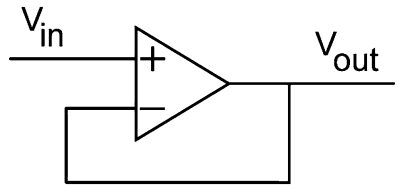


Fig. 4.6 Illustration of voltage follower circuit

www.dbooks.org

The entity represented by the triangle is an operational amplifier, which consists of a differencing unit followed by a multiplication unit. The gain is usually called β (read "beta") and is usually around hundreds of thousands. One of the biggest benefits of this construction is that it allows us to create a (largely) *directed relation* between the input and the output: the operational amplifier will force the value of the output to follow the value of the input closely, but not the other way around. The power that the operational amplifier consumes to achieve this comes from a power source not explicitly represented on such diagrams. Because of their usefulness in building controllers in various circuits, operational amplifiers are one of the most commonly used components in electronic circuits.

Example 4.2: Constant Gain Plant Now consider the situation where our plant is no longer simply a wire but rather an amplifier with a gain H. Again assume that our multiplier has gain of G. We will use the same conventions:

- e is error,
- x is plant input,
- y is plant output, and
- z is desired plant output.

This new system is described by the following equations:

1. $e = z - y$ is the error,
2. $y = Hx$ is the behavior of the plant,
3. $x = Ge$ is the behavior of the controller.

Substituting e from 1 into 3 yields

$$x = G(z - y).$$

Substituting x from this equation into Equation (4.2) yields

$$y = HG(z - y).$$

Following a similar set of steps, and assuming that $1 + GH$ is not zero, we can arrive at

$$y = \frac{HG}{1 + HG} z. \tag{4.2}$$

Note that the equation we derived in the first example is a special case of this equation, where H is equal to 1. But it is more instructive here to consider carefully how the overall gain factor $(HG/(1 + HG))$ changes as H changes. Assuming that G is positive, then for a positive value of H, we get behavior similar to what we saw before: the output will be close to the goal value. As H grows arbitrarily large, the ratio $HG/(1 + HG)$ also goes to 1, which means that this controller is effective at getting the output to have the desired value. It is also interesting to note that if H is negative and it goes to negative infinity, the ratio also goes to 1, although it does so from above. This is an interesting observation because it shows that a feedback controller can work effectively even when the gain of the system is negative.

However, neither of these observations imply that such a controller will work for any combination of H and G. In particular, if the term $1 + HG$ is equal to zero, we cannot derive the last equation. In fact, what we have in that situation degenerates to $0 = HG \cdot \text{goal}$, which contradicts the assumption that goal is any value other than zero. This is a situation where our controller has a singularity. In fact, the overall behavior of the system is problematic for values of H between $-1/G$ and 0. The singularity occurs exactly at the value $-1/G$, but the overall gain is negative up until the point when H is 0. To get a sense of how H affects the overall gain, let us take G as 1 and plot the term $H/(1 + H)$ as H. The result is shown in Figure 4.7.

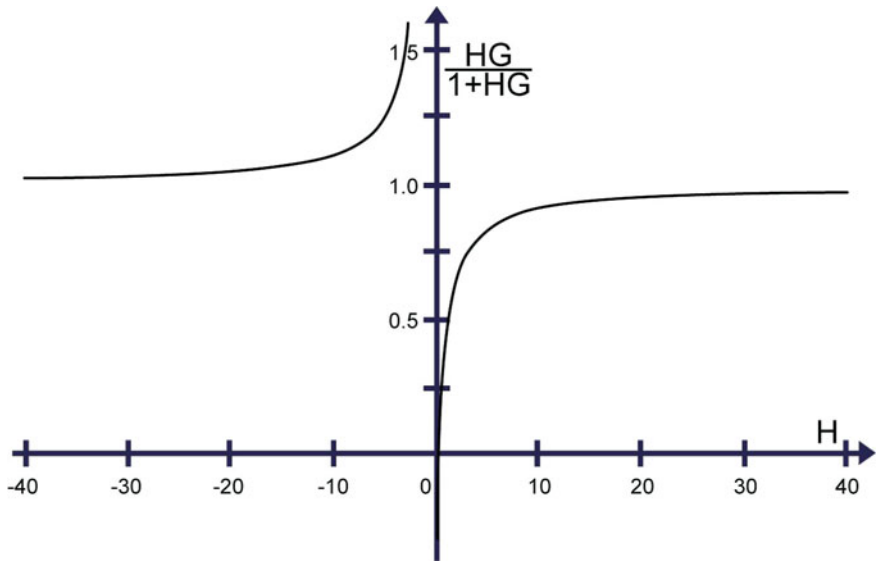


Fig. 4.7 The total gain $\frac{HG}{1+HG}$ plotted against H

Thus, when we were just trying to find a G to control a system with known (positive) gain, it was enough to take any positive G, and the higher the gain the closer we were able to get to the goal. However, if the gain of the system that we are controlling is not strictly positive, then we may need to do (significant) additional work to make sure that our negative feedback control strategy does not lead to singularities or undesired behavior.

While checking that H is positive can be relatively easy to do for certain kinds of systems, it is not necessarily easy to do when the "gain" of the system itself depends on the state of the plant. In that case, H can be viewed as the derivative of the output with respect to the input for the system in that particular state. Ensuring that such a controller is well-behaved requires the analysis of the system for *all* possible states.

Exercise 4.2 For the system in Example 4.2, take z to be 1 and G to be 1, and plot e as it varies with H from -50 to 50. On the same graph (if possible), plot the curves when G is equal to 10 and 100.

www.dbooks.org

Example 4.3: Dynamical Systems Consider the situation where our plant is not just a simple (constant) gain but, rather, a dynamical process that accumulates the input variable. An example of such a system is an object whose position we can observe and whose speed we can control. This is the case when we are driving a car. Now the plant is no longer a simple function of input to output but, instead, a notion of time must be factored in.

Following a convention similar to the one we used before, at any time instant *t* we will say that:

- $e(t)$ is error,
- $x(t)$ is plant input,
- $y(t)$ is plant output, and
- $z(t)$ is desired plant output.

This new system is described by the following equations:

1. $e(t) = z(t) - y(t)$ is the error
2. $y(t) = y(0) + \int_0^t x(s)ds$ is the behavior of the plant
3. $x(t) = Ge(t)$ is the behavior of the controller

Substituting equation $e(t)$ from Equation (4.1) into Equation (4.4) we get $x(t) = G(z(t) - y(t))$. Substituting x from this equation into Equation (4.2) we get

$$y(t) = y(0) + \int_0^t G(z(s) - y(s))ds \tag{4.3}$$

If this is our first encounter with such an equation it may seem imposing. However, there are many ways to make progress towards solving it. In fact, because it can be viewed as a *linear differential equation*, there are techniques to transform it to the frequency domain and proceed almost exactly in the same way we did with the two previous examples, which are static in the sense that they do not have any notion of time.

The reader interested in linear systems (such as students of electrical engineering) can consult the ample literature available on linear circuits and linear control theory to explore this path. Here, instead, we will explore a path that does not require the elegant but somewhat specialized machinery used for linear differential equations.

Returning to our Equation (4.3), we can proceed by making several simplifications that help us get a sense of how this equation behaves. Once we have a solution under simplified assumptions, we will see how we can remove the assumptions. To get started, let us assume that G is 1 and also that $z(t)$ is the constant function 0. Now we are left with the equation

$$y(t) = y(0) + \int_0^t -y(s)\ ds.$$

Because integration, as well as differentiation, is a linear operator, we can take the minus sign outside the integration operator to get

$$y(t) = y(0) - \int_0^t y(s)\, ds.$$

We can further simplify this equation by differentiating both sides, which yields the equation

$$y' = -y.$$

This equation does not uniquely determine the solution of the previous equation, because it loses the information about $y(0)$. However, it is still a very useful equation to arrive at. It tells us that, whatever the solution $y(t)$ will be, it must have the property that it is equal to the negative of its derivative. One function that has this property is the exponential function $y(t) = e^{-t}$. In fact, for any constant a, the function $y(t) = ae^{-t}$ satisfies the last equation. We can pin down the value of a by equating $y(0)$ to ae^{-0} which is simply a. Thus, if $y(0) = 2$, the output of the system is $y(t) = 2e^{-t}$. The solution is shown in Figure 4.8.

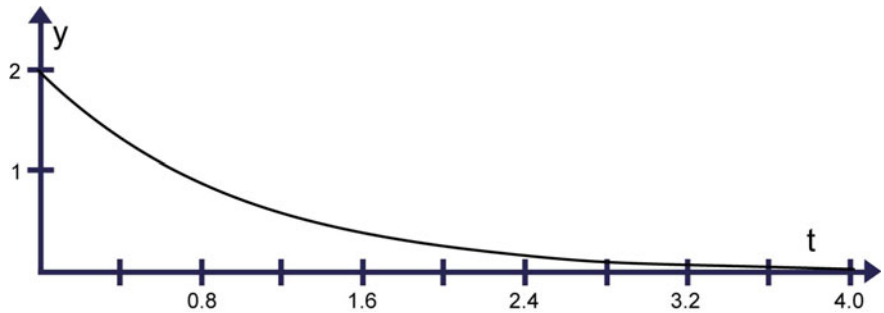


Fig. 4.8 Solution of $y' = y$, $y(0) = 2$

Judging from this example, our feedback strategy appears to provide us with the controlling effect that we want. If the desired goal is the function that is 0 at all times, our simple feedback controller with gain G of 1 seems to cause the system to approach that goal. Furthermore, the system appears to approach this goal relatively quickly. The graph in Figure 4.8 indicates that, within 5 time units, the output of the system becomes quite close to zero.

Now let us consider the case when the output is not zero but instead some other constant value Z. Here the equation for the system becomes

$$y(t) = y(0) + \int_0^t (Z - y(s))\, ds. \tag{4.4}$$

www.dbooks.org

Differentiating both sides in (4.4) yields

$$y' = Z - y.$$

Because we know that the derivative of a constant value is zero, we can consider trying the function $y(t) = Z + ae^{-t}$ as a solution to this equation. Indeed, it works as expected, because subtracting Z from Z also yields zero. The specific value for a is determined in the same way as before: namely, using the initial condition $y(0) = Z + a$, thus, $a = y(0) - Z$. Of course, we assume that Z is given. For $y(0) = 1$ and $Z = 5$, $a = -4$, and $y(t) = 5 - 4e^{-t}$, which has the form given in Figure 4.9.

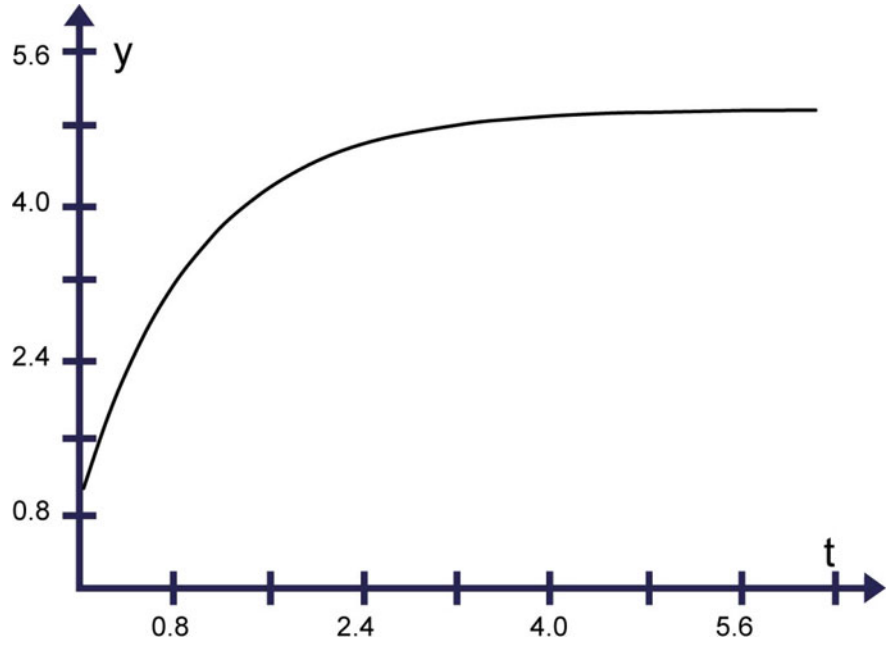


Fig. 4.9 $y(t) = 5 - 4e^{-t}$ plotted against t

Here again we notice that the output value quickly goes to the desired value: in this case 5.

It is also possible to analytically compute a closed form for some time-varying functions. For example, we may wish to determine the output of the system when $z(t) = Z - t$. In this case the equation becomes

$$y(t) = y(0) + \int_0^t (Z - t - y(s))\, ds \tag{4.5}$$

and differentiating both sides yields

$$y' = Z - t - y.$$

By a similar process of inspection to the previous example we can see that the solution is

$$y(t) = Z + 1 - t + ae^{-t},$$

which when substituted into both sides of the last equation yields

$$-1 - ae^{-t} = Z - t - Z - 1 + t - ae^{-t}.$$

This can be reduced to

$$-1 - ae^{-t} = -1 - ae^{-t},$$

which is clearly satisfied.

For $Z = 5$, $y(0) = 2$, the solution equation at time 0 gives us $2 = 5 + 1 - 0 + a$ which is $2 = 6 + a$ and so $a = -4$. Thus, $y(t) = 6 - t - 4e^{-t}$, which is the black curve in Figure 4.10.

The red curve here is the desired output function. Thus, this controller does a reasonable job of tracking the desired output function. But it does not track it as precisely as we might expect. This brings us naturally to the question of how changing the gain G in our feedback controller can affect our system. To take into account the gain G we would revise (4.5) to be

$$y(t) = y(0) + \int_0^t G(Z - t - y(s))\, ds. \tag{4.6}$$

Differentiating both sides we get

$$y' = G(Z - t - y). \tag{4.7}$$

A good trick to understand how we solve this analytical problem is to step back and ignore for a moment $z(t)$ (which shows up in the last equation as the term $Z-t$). Thus, with $z(t) = 0$ the system would be governed by the equation $dy(t)/dt = -Gy(t)$. If we naively plug in the (incorrect) solution $y(t) = ae^{-t}$, we get a good hint of what needs to be done to get a correct solution. The left- hand side is $-ae^{-t}$ and the right-hand side is $-Gae^{-t}$. To fix this disparity we need to put G into the solution form so that it "comes down" as a multiplicative factor when we differentiate, that is, to use $y(t) = ae^{-Gt}$. For this expression, both sides are $-Gae^{-Gt}$, and the equation is satisfied.

With this as the starting point for the solution with $z(t) = 0$, we can go back to Equation (4.7) and try to construct the solution for this problem. Again, we can substitute an educated guess for the solution and see what happens. For example, we might consider

$$y(t) = Z + 1 - t + ae^{-Gt},$$

www.dbooks.org

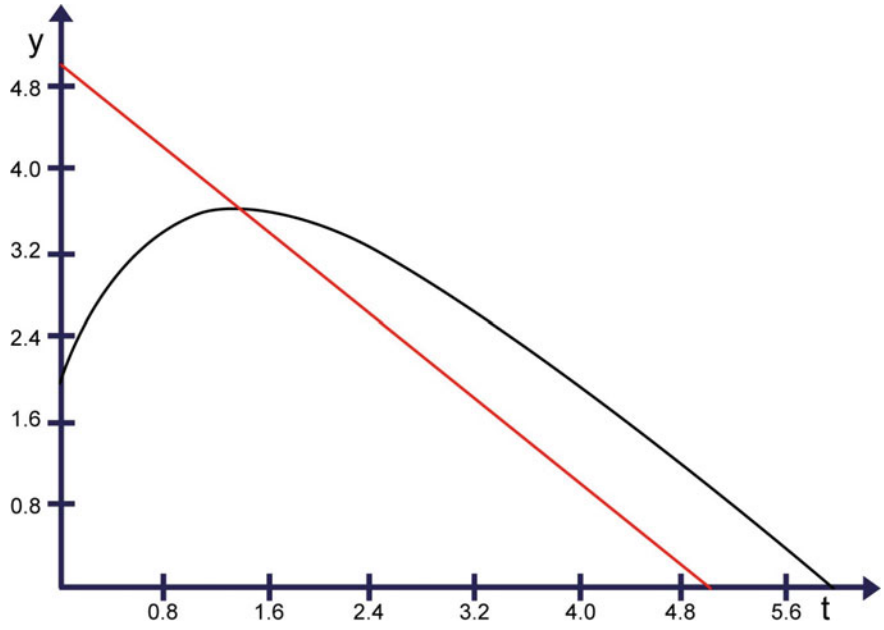


Fig. 4.10 The desired output, red, and output, black, are shown for the suggested controller

which is just a small tweak to the solution for the case of $G = 1$ that we had derived above. Now the left-hand side is

$$-1 - Gae^{-Gt}$$

and the right-hand side is

$$G(Z - t - Z - 1 + t - ae^{-Gt}) \; = \; -G - Gae^{-Gt},$$

which means that we just need to find a way to get -1 instead of $-G$ on the right-hand side. This suggests a solution function

$$y(t) = Z + (1/G) - t + ae^{-Gt}.$$

Plugging this term into the left- hand side yields

$$-1 - Gae^{-Gt}$$

and into the right-hand side

$$G(Z - t - Z - (1/G) + t - ae^{-Gt}),$$

which is $-1 - Gae^{-Gt}$ and we confirm that this is the right form for the solution. For $Z = 5, y(0) = 2, G = 10$, the solution equation at time 0 gives us $2 = 5 + 0.1 - 0 + a$

which is $2 = 5.1 + a$ and so $a = -3.1$. Thus, $y(t) = 5.1 - t - 3.1e^{-10t}$, which is the black curve in the following in Figure 4.11.

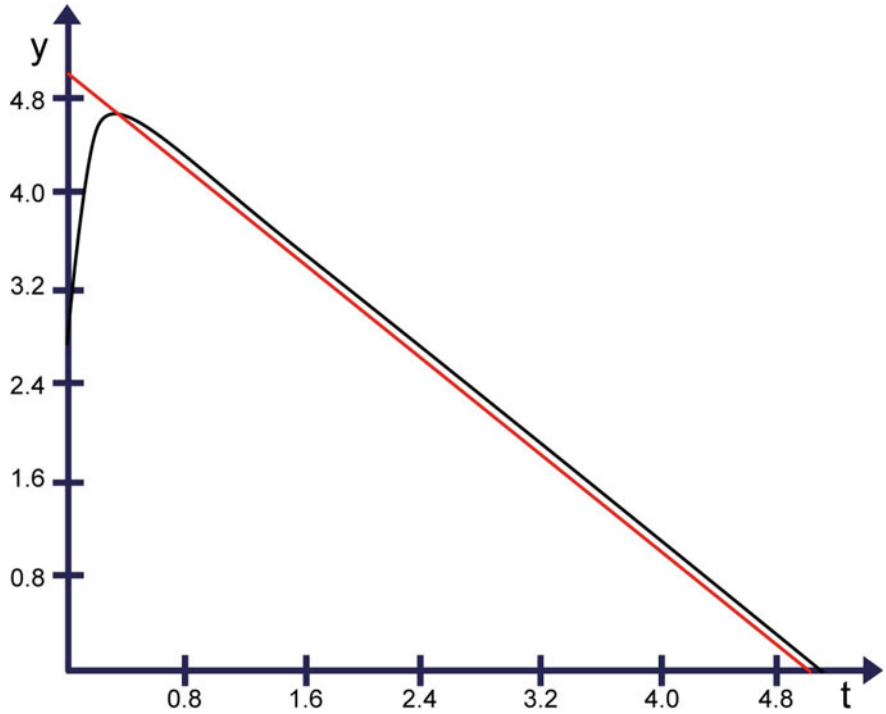


Fig. 4.11 The desired output, red, and output, black, are shown for the suggested controller

In this graph, it is clear that our new controller is tracking the target function much more closely, both in terms of how quickly it intersects with the trajectory of the target function and in the difference in amplitude between the two functions after this intersection happens.

To summarize, we have seen that proportional feedback control can work for (a) a do-nothing plant where the output is equal to the input, (b) a proportional plant where the output is proportional to the input (and where some care is needed to avoid singularities), and (c) for a simple dynamical system.

Exercise 4.3 Implement the last system discussed above in Example 4.3, using Acumen. Run a simulation that confirms the analytical results that we derived for the output function. Using your Acumen model, vary the value of G so that the error between the desired output and actual output at time 1.6 is no more than 1/100.

www.dbooks.org

4.5 Multi-Dimensional Error and Proportional/Integral/Differential Feedback Control

We noticed above that by increasing the controller gain we can often get better performance in terms of how close the output value tracks the desired output value. However, simply increasing the gain is not always the most desirable approach to improving the performance of a controller. For instance, using high gain can entail using large amounts of energy. In addition, if the feedback reading is faulty or the link is broken, the system would have high "open-loop" gain, which can lead to feeding the plant inputs that can cause it damage.

Stepping back to look at the big-picture of proportional feedback control, we can consider the possibility of performing different computations on the feedback; then using them to compute different notions of error, and combining them into a single error term that is used to drive the plant. This can be depicted as shown in Figure 4.12.

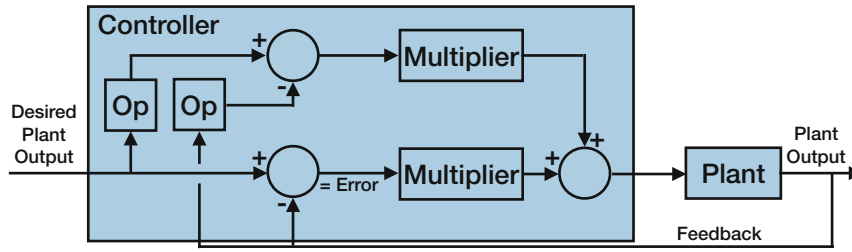


Fig. 4.12 A generalized block diagram for a controlled plant

Here Op could be, in principle, any operation. For example, in the case of the car where we are controlling the speed, we can take the Op operator to be the differentiation operation, thereby converting the position reading into a speed reading. Alternatively, we can take Op to be integration, in which case we would have an estimate of the distance traveled. There is also no reason to limit ourselves to one or the other, and we can add another Op path and support both operations at the same time. Each path that contributes to the error in this layout can be viewed as one dimension of a tuple that together represents the error. Then, the input is computed as a weighted linear sum of these individual components. The question then is what type of behavior we get from the system by incorporating each of these paths.

Example 4.4: Throttle-Controlled Car Consider a system that consists of a car that is controlled by means of a throttle parameter (assumed to be proportional to acceleration), and where the output of the system is the car's speed. The controller has as input the desired speed. This example is instructive for getting a sense of the effect of each of these two choices of operators (differentiation and integration), which are used relatively frequently. But it is easier to imagine that we disable the other path(s) when we try to understand the action of each one. Then we can consider

the effect of adding the errors from each component at the end. Using differentiation alone would give us a controller that tries to get the car to have the same *acceleration* as the controller's input appears to have. This could be a useful control strategy, for example, when we want to satisfy certain comfort constraints (high acceleration interferes with passenger comfort). Practically, some care is often needed with such controllers, as derivatives can change much more suddenly than the actual value of the desired speed signal. In addition, estimating acceleration from the speed signal can be prone to noise and measurement artifacts. When we used simple proportional feedback on a dynamical system that had an integrative effect, the key underlying differential equation had the form $y' = -y$. When the differentiation operator is introduced, our system remains essentially of the same form. In a sense, we just have more ways to achieve the same coefficient in such an equation.

Using integration alone would give us a controller that tries to get the car to have the same *position* (up to some constant offset) as the controller's input appears to add up to. This method can be used to avoid constants such as the ones observed in the last system studied in Example 4.3.

In general, it can be very helpful for realizing a controller that keeps some history of its past behavior, as well as for getting a system to function properly in situations where the entire past behavior can affect the current state (such as the way in which a car's entire speed history affects the position where it ends up).

However, with the use of the integration operator in the controller we have an equation of the form $y'' = -y$. An important characteristic of such equations is that they can lead to oscillation. In particular, $sin(t)$ can be a solution to such an equation. This is one reason why it is often useful to combine different dimensions of error to produce the final result. For example, discounting the acceleration signal in proportion with the current speed generally produces a damping effect on such oscillations. The essential equation underlying such combined (proportional-differential) systems can be viewed as $y'' = -y - y'$.

Exercise 4.4 Construct a system where there is a single-dimensional error using the derivative of the feedback that has an output described by the equation $y' = -y$.

Exercise 4.5 Construct a system where there is a single-dimensional error using the integral of the feedback that has an output described by the equation $y'' = -y$.

Exercise 4.6 Construct a system where there is a two-dimensional error that has an output described by the equation $y'' = -y - y'$.

The lab will provide some further experience with such systems. This experience will help you develop a competitive project submission.

www.dbooks.org

4.6 Chapter Highlights

1. Control Theory
   (a) An almost inseparable part of CPS
      - Examples: HVAC, cars, airplanes, Segways
   (b) At the level of mathematical models, what does it mean?
      - We want to find an input that gives us a certain output
      - Basically, an inverse
        – Example: Water tap with angle
   (c) But is an inverse function really what we want?

2. Negative feedback in a proportional system
   (a) A simple multiplicative system (H)
   (b) Simple negative feedback gain (G)
   (c) Careful deviation of composite gain
   (d) Limit as G goes to +/− infinity
   (e) What if H is + or −
   (f) What about the singularity?

3. Negative feedback in an integrative system
   (a) A simple integrative system
   (b) Derivation of corresponding equation
   (c) Form of response
   (d) Effect of changing parameters
   (e) Energy cost

4. Negative feedback in a doubly integrative system
   (a) A double integrative system
   (b) Derivation of corresponding equation
   (c) Form of response
   (d) Possibilities for stabilization
      - Pick up feedback earlier on
      - Estimate feedback

5. Negative feedback in 2D and 3D
   (a) Equations in two dimensions
   (b) Equations in three dimensions

4.7 Study Problems

1. Solve and submit Exercises 4.1, 4.2, 4.3, and 4.4 of this chapter.

2. Consider the operational amplifier (op-amp) in the configuration shown in Figure 4.13. Assume that the gain for this operational amplifier is G.

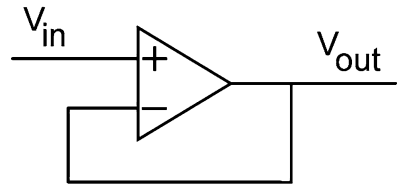


Fig. 4.13 A configuration with op-amp

   (a) Write down the equation governing the relation between V_{in} and V_{out}, and using the gain G.
   (b) Use the equation you wrote above to derive an equation for the ratio $V_{\text{out}}/V_{\text{in}}$ in terms of G.
   (c) Calculate the smallest value for the gain G that would ensure that V_{out} is always within 10% of V_{in}, that is, V_{out} is always between 90% and 110% of V_{in}.
   (d) Insert a gain H between V_{out} and the negative $(-)$ input of the op-amp. Determine H to ensure that V_{out} equal exactly V_{in} for the gain you determined in part (c) of this problem.

3. Consider the situation where you have been assigned the task of building a controller for an elevator. The controller gets two signals: position_goal, and speed_goal. It also gets two measurement signals, called current_position, and current_speed. The controller must produce one signal called desired_acceleration.

   (a) Write down a mathematical expression for a simple proportional-integral-differential (PID) controller that will compute the value at any given time for output desired_acceleration in terms of the inputs available to the controller. Assume constants K_1, K_2, K_3 ... for any gain coefficients that may be needed in your expression.
   (b) Write an Acumen model that captures the effects of quantization and discretization of all input signals and discretization of all output signals to the controller. Assume all values are quantized by 0.01 sized steps and discretization is for 0.1 time steps.
   (c) Create a test scenario for your system, and use it to determine reasonable parameters for each of these parameters. Justify your choice of test scenario and the results that you have arrived at.
   (d) What kind of damage can result from instability in this system, and what methods would you use to manage any destabilization effects that could result from any unexpected disturbances on the elevator system?

www.dbooks.org

4. You are designing a control system that allows an object at point Q with mass 1 to follow another object at point P that is continually moving. Both points have a two-dimensional coordinate. We will say that Q is at position q and P is at position p. You are using a controller of the form:

$$q'' = A(p - q) - Bq'.$$

You start your design process by making a reasonable initial guess for the constants A and B.

(a) Should you select positive or negative values for A and for B? Explain your choice.
(b) Suppose that you run a few tests (simulations) and find that Q either oscillates around P or overshoots P when P moves. What change would you make to the parameters A and/or B to address this problem, and why?
(c) After making this change, suppose you find that Q is following P but it appears to be slow or constantly falling behind (lagging). What change would you make to the parameters A and/or B to address this problem?
(d) Suppose your selection appears to work well except when the point P moves faster the lag (the distance between the two points) gets bigger. Without making any changes to A or to B, what changes can you make to your controller to improve it?
(e) Identify some practical difficulties that one can encounter in realizing such a controller.

4.8 Lab: Exploring Control

The purpose of this lab is to get some practice with the basic ideas of control in the context of simulations of some simple but representative dynamical systems.

The following model describes a first order dynamical system with its initial conditions:

```
model Main (simulator) =

initially
x = 1, x' = -1

always
x' = 5-x
```

Let us begin by considering an important technicality. In Acumen, one specifies more than the initial conditions usually needed for differential equations. In the above example, as a differential equation we would only need to specify the initial value for `x`, but here we also specify one for `x'`. For this model, the extra initialization

will have no significant effect, as the equation in the `always` section will update the value of `x'` as soon as the simulation starts.

Going back to this system itself, this simple differential equation can be seen as a complete control system driving the value of `x` to `5` using a negative feedback scheme where the speed with which `x` changes, that is `x'` is proportional to the difference between our target `5` and the current value of `x`.

This type of controller will bring the value of `x` relatively quickly to be quite close to the target. If we want to speed up this process, we can multiply the term `5-x` by a factor greater than one.

Simulate the original model above. Next, add a gain factor as described above to see its effect on how the value of `x` changes over time. Confirm that changing the gain factor does not change the target value that the system seems to converge towards. Change the target value to another value to confirm that the new controller (with the higher gain) still seems to converge to the target value you specified.

A disadvantage of such a controller is that the value of `x` can get arbitrarily close to the target, but will never really get there in any finite amount of time. This motives investigating second order systems. The simplest way to get a second order system is in fact to differentiate both side of the question above. This gives us the following model:

```
model Main (simulator) =

initially
x = 0, x' = 1, x''= -1

always
x'' = -x'
```

If we give the derivative `x'` the variable name `y`, then this equation is `y' = -y`. This helps us see that this itself can be seen as a control system where we are driving the variable `y` to the target value `0`. This is a good property for our original model to have, where our explicit goal is to drive the value of `x` to five, but we implicitly would also like to have the value of `x` to stop changing when it reaches that value.

Explore modifying the above model to drive the value of `x'` not to zero but rather to a non-zero speed, say 10. As a hint, you can look at our original model and how we drove the value of `x` to 5 there. Run the simulation to verify that the model behaves as you expect.

But as long as we have an equation which we can reduce to a first order equation, we are not really exploring the full power of a second order equation. The following model is a more representative example of a second order system:

```
model Main (simulator) =

initially
x = 1, x' = 0, x''= -1

always
```

www.dbooks.org

```
x'' = -x
```

Again, we can view this system as control system. Here, the acceleration is being used to actuate the system, and the value of the variable x is being driven via negative feedback to the target value of zero. At the same time, we can view this system as a spring mass system, where the acceleration, which is equal to the spring force, is proportional to the length of the string. With this analogy, however, we can expect the system will oscillate around the target rather than converge towards it. That said, we are making a bit of progress, since we are now able to get the value of x to cross the target value. In a sense, what we need to do now is to get rid of the oscillation, or at least reduce it to an acceptable level.

Simulate the above system to verify this expectation.

Compared to the previous model above, we have lost the term that allowed us to drive the speed to zero. In a sense, we can "mix" the two effects by having both terms on the right-hand side of the equation, as in the following model:

```
model Main (simulator) =

initially
x = 1, x' = 0, x''= -1

always
x'' = -x -x'
```

This equation can also be seen as a control system where we are driving x to zero and damping the actuation using the speed x'. The term x' is considered to damp the system because the higher the speed the lower the acceleration (as determined by the equation) will be. A key benefit of working with such a system is that now the variable x can actually reach the intended value, overshoot it, but still gradually converge towards it.

Simulate the above to confirm these observations, and in particular, that the damping diminishes with time.

We can generalize the above model with the following template, introducing additional undefined variables A,B, a, b:

```
model Main (simulator) =

initially
x = 1, x' = 0, x''= -1

always
x'' = -A*(x-a) -B*(x'-b)
```

The capital letter variables are gains, and the small letter variables are offsets. These variables remind us that we can tune the gain factor that we feedback to the actuation of acceleration, and we can choose target values for both position and speed as we see fit for our application.

Simulate instances of the above model template with different values assigned to these variables to better understand their role and their interactions. Try values such as 1 and 10 for `A` and `B` in alternation. Do you notice qualitative differences in response? Try other values. Can you identify qualitatively different behaviors and build a hypothesis about when (that is, based on what gains) they arise?

Next, go back to the value 1 for both of these variables, and explore different values for the small letter variables. What can you say about how they work? Can you imagine situations where the small letter variables come from another dynamical system?

To conclude this lab, consider how you can define in your model a notion of power for the amount of energy being put into the system. As a hint, consider that in dynamical systems power is force times speed. Next, consider how you can use this definition to define the total amount of energy being put into controlling the system. Next, define a cost function that includes both energy and distance from target integrated over time. What applications can you think of for this function?

4.9 Project: Acceleration-Based Player for Ping Pong Robot

This chapter's project is still aimed at developing a ping pong player that can outperform the other player, and at the very least, the default player. The new challenge that we consider this time is that the bat has mass. To capture this more realistic model of the system, the new bat Actuator can only receive a signal for acceleration. The force that you can apply is also limited. Your task is therefore to continue to develop the player that you created for the last chapter's project activity by determining what forces must be exerted on the bat in order to realize the action that you were previously able to specify directly in terms of speeds. Note that the more acceleration exerted, the more energy is consumed (maxE). For this activity, your modified player must also consume less energy than the default player and predict the second point better than the default player. You will need to develop your own strategy to make the bat accelerate slowly.

Beyond this formal requirement you should feel free to enhance and improve your player in other creative ways. You have a lot of freedom in how you approach this task, and you have several issues that you can explore. Remember that your goal is to make the best player, and that at the same time you do not know how the other player will play. But you can imagine that your task is to build a player that can beat as many players as possible. Issues you may choose to consider can include:

- Optimizing the parameters in the default controller (given to you as part of the default player for this tournament)
- Developing an alternative controller (either as a continuous equation or as a hybrid controller)
- Maximizing the accuracy of this player while reducing its energy consumption
- Improving the methods of estimation in your player, either by tuning parameters or developing completely new methods

www.dbooks.org

- Changing the playing strategy (path planning)
- Change parameters of the playing strategy

Feel free to explore any of or all these possibilities, as well as any others that you may feel can give your player a competitive edge!

4.10 To Probe Further

- Article on Control Theory.
- Article on Operational Amplifiers.
- Video on TED talk about CPS with quadcopters
- Video on control of an under-actuated system, inverted triple pendulum
- Video on quadcopters playing racket
- Article on a complex control loop: Sleep and mood
- Robohub article about control with quadcopters and an inverted pendulum
- Article on Amazon developing quadcopters for delivery service.
- Video of TED talk about how a fly flies

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Chapter 5
Modeling Computational Systems

How do we model digital computers operating in a physical context? This chapter looks at the sense-compute-actuate model of implementing controllers. It addresses the physical aspects of implementing computation, both for analog computers and digital computers, and covers quantization (and quantization levels), discretization (and sampling), the Nyquist–Shannon Theorem, embedded hardware and software, and real-time systems and constraints.

5.1 Introduction

Now we want to implement control systems. How do we do that? We need to build a machine! Often, such a machine includes a sensor, an embedded computer, and an actuator. An embedded computer can be just a regular computer that has been turned into a dedicated machine inside another device, such as a watch or a car. Such systems often have demanding constraints in terms of size, energy utilization, unit cost, reliability, and real-time responsiveness. Meanwhile, real-time responsiveness typically means that the system must respond to some inputs (such as commands from a user) within a certain time limit.

We can, in principle, implement a controller with an analog circuit consisting of resistors, capacitors, and inductors. In fact, at one time people did just that. Nowadays, however, it is more common to use digital circuits. Why did this change happen? It is instructive to consider the general circumstances that created the right conditions for this fundamental shift in computing technology to take place. For example, analog computers can be quite fast and versatile. Why would they be phased out?

By and large, the biggest concern with analog computing was reproducibility. Performing a computation precisely required precise parameters to components. Not only that, but these components and their characteristics were sensitive to temperature and surrounding electromagnetic fields, which themselves can be a function of the ambient environment or how the components themselves are used and connected. To

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_5

www.dbooks.org

make matters worse, over time, most the characteristics of most components changed as they aged.

Digital circuits use analog electronic components but, in contrast to analog circuits, their primary function is to connect them and build circuits out of them to transmit, process, and output only two levels of voltage. These two levels represent a binary digit. Any value outside those two levels is considered an indicator that the system has not yet finished transitioning between those two levels.

Our first impression of such a system might be that it seems like a vast underutilization of electronics: Instead of cherishing the continuum of possible values that an electronic circuit produces, and viewing it as a distinct value, we are concerning ourselves only with two levels (or, more precisely, two ranges of levels) that represent a single binary digit.

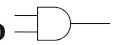
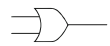
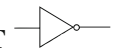
This design choice of working with "two levels" also means that we compute only with "quantized" values; that is, values that have been explicitly turned into a finitary, discrete quantities or representations. The big advantage of this is that it enables the transition from working with components that have uncertainties about their characteristics, to circuits where an answer is clearly either right or wrong. This has made it possible to understand and overcome the challenges of building very fast, small, and energy-efficient systems that are nevertheless deterministic and highly reliable. The broad success of this transition is witnessed by the pervasive presence of digital computing technologies around us today.

5.2 Quantization

Quantization helps make computing with circuits robust. In essence, quantization is the mapping of ranges from an infinite set to ones in a finite set, usually preserving order. Digital circuits are quantized in their voltage ranges. In binary digital circuits, there are just two main ranges, one below a certain level, and one above, with the middle being viewed as an invalid, transient state. Quantization is one of the key ideas that makes it possible to build highly reliable digital systems out of analog components that may, individually, be much less reliable.

Conceptually, the basic building blocks for digital computers are gates that operate on binary inputs to produce binary outputs, such as NOT, AND, and OR gates. The following table introduces some standard textual and graphical notation, as well as the relation that these gates impose between inputs and outputs:[1]

[1] This table is derived from the article Logic Gate.

Type	Symbol	Expression	Truth Table
AND		$\mathbf{A \cdot B}$	**A B | A · B** 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$\mathbf{A + B}$	**A B | A + B** 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$\mathbf{\overline{A}}$	**A | $\overline{\mathbf{A}}$** 0 | 1 1 | 0

In actual electronic circuits such as those on an integrated chip, gates are realized (or "implemented") using transistors, which are continuous systems. However, they are designed to be "stable" only for certain "high" and "low" ranges of values that correspond to a 0 or a 1 (not necessarily in that order). Then, entire circuits built out of these gates are designed to operate in a reliable manner only for such special values that correspond to 0s and 1s.

The exclusive use of only two ranges of values (one corresponding to 0, the other to 1) immediately brings up a fundamental problem: If we need infinite precision, then we need an infinite number of bits to represent the number. More often, builders of digital computers adopt a finite representation for values. This is a challenge that we must address whenever we use a digital computer.

5.3 Discretization: How Fast Can Your Circuit Go?

From the time that the inputs arrive, each gate needs a minimum amount of time before it can produce the correct value on the output wire. Such propagation delays can range from picoseconds to tens of nanoseconds, depending on the technology used to create them. No matter how minor such delays may be, any combination of gates will also have to take some time before the right answer is produced. When all propagation of signals in a circuit is completed, it is said to be in a *stable state*. Any well-designed circuit should have a clearly defined time after which we can guarantee that it has had enough time to stabilize.[2] Most digital circuits have this well-defined

[2] Note that not all circuits can become stable. For example, some circuits are used to create periodic signals (such as a clock signal) that are meant to oscillate indefinitely between 0 and 1.

www.dbooks.org

value, but this introduces another limitation of digital computers. In particular, it determines the maximum rate at which the circuit can sample (and process) external inputs.

To support *memory* and *iteration*, digital computer circuits generally contain wiring "loops" that feed certain outputs back as inputs to the circuit. As a result, determining the amount of time that a circuit needs to produce the final, stable output can be challenging. Circuits that are created by freely combining gates are generally called *asynchronous circuits.* An alternative strategy is to use certain wires as *clock* wires to make groups of gates work in a lock-step manner. Such circuits are called *synchronous circuits* and clocking tends to simplify and help organize their design. Both circuit types have advantages and disadvantages for different applications, but predominantly synchronous circuits are used in microprocessor design today.

The typical approach to modern circuit design involves having a single clock that drives the operation of the whole circuit. The clock rate puts an absolute bound on the maximum possible sampling rate. There are, however, other considerations (such as available memory and the rate at which the information can be processed by the rest of the circuit) that may lower the maximum possible sampling rate. This means that any system implemented using a digital computer can only observe a continuous signal at specific times, and with a minimal gap between such samplings. The process of mapping continuous time to such a (countably infinite) notion of time is called *discretization.*

5.4 Detour: Boundedness of Digital Memory

Binary gates can be used in a variety of ways to construct memory cells by using the idea of feedback in a digital circuit. In particular, consider an AND gate with two inputs A and B, and an output C. Now consider the situation where we connect the output C back to the input B. The result is that we have a circuit with only one valid input A (the B input is already "plugged" with the output C). However, C can still be viewed as an output (as we are allowed to connect one output to several inputs). How would such a circuit work? To answer this question, we must consider the behavior of the system not only based on what the single input A is, but also on whatever the current value for the output C happens to be. A careful analysis would show that if we ever start with a situation where the input was 1 (or True), then it would stay that way as long as the input A was also 1. Once the input A changes to a 0, the output would be forced to switch to 0 and would stay that way forever, no matter what value we input for A.

This is a simple circuit that exhibits a very basic type of memory. In a way, it is a single-event memory: Assuming that we started it in a state where the output was 1, it would "remember" any occurrence of a 0 input by immediately turning the output to 0, and staying that way forever.

The exercises in the Study Problems section will show us, among other things, that storing one bit of memory takes a few gates and a bit of wiring. Furthermore,

as long as our CPUs, external memory, and external storage devices are finite, our digital computers can only store a finite amount of information, although the cost of such storage seems to be perpetually decreasing.

5.5 Detour: From Hardware to Software—Storing Executable Commands in Memory

So far, this chapter has focused on hardware. The reason for this is that the most fundamental effects that arise when we connect an embedded digital computer to a physical system, namely, quantization and discretization, are due to the nature of hardware rather than software. Of course, software can contribute to the *extent* to which these effects occur, and can certainly help to mitigate them. Furthermore, these effects have a significant influence on the way embedded software is built, giving rise to the need for responsiveness, real-time deadline, reliability, fault tolerance, and many other features. But the goal of this chapter is to help the reader develop a sense for the nature of the problems that arise and for how these problems can manifest themselves in the overall behavior of basic Cyber-Physical Systems. After mastering the material discussed in this chapter, the To Probe Further Section will introduce the reader to some basic concepts relating to the development of embedded systems.

5.6 The Effect of Quantization and Discretization on Stability

It is clear that quantization and discretization are two new issues that must be addressed in the process of implementing a controller. Both control theory and embedded systems techniques provide conceptual methods for developing such implementations. For the most part, these topics are beyond the scope of the current chapter. What we want to do instead is to help the reader develop an intuitive understanding of the effect of quantization and discretization on the operation of a controller.

5.7 Abstract Modeling of Computational Effects

Interestingly, there is no need to switch completely to a specific hardware and software platform, and to model in detail what happens when we implement the entire controller using a digital computer. Rather, we can get a pretty good idea of the effect of these two transformations by making small modifications to a model of a continuous controller.

We will consider the effect of quantization and discretization on a controller attached to a point mass. Without any quantization or discretization, the idealized

www.dbooks.org

controller produces a force proportional to 10 times the distance of the mass from the original:

1. error $=$ 0 $-$ position,
2. controller$_{\text{force}}$ $=$ $10 *$ error,
3. $f = ma \Rightarrow$ a $=$ f/m $=$ controller$_{\text{force}}/1$.

In the first equation we are saying that we are taking the goal position as 0. Thus the error will simply be the negative of the position. Equations 1, 2, and 3 together imply that $a = x'' = -$position $= -10x$. Thus, the equation for the entire system (mass and controller) is $x'' = -10x$. This is an idealized, continuous system that also happens to be critically stable, as the system continues to oscillate indefinitely. In Acumen, this system can be modeled as follows:

```
model Main(simulator) =

initially
x = 1, x' = 0, x'' = 0

always
x'' = -10*x
```

Note that the equation above is a very compact representation of the system we described above (proportional control). We can introduce extra intermediate "dummy" variables to point to the places where an idealized "sensor" and an idealized "actuator" would be carrying a signal of a certain value. It is enough for our purposes here to identify a value representing what the sensor reads, and we can do it by modifying the Acumen model above as follows:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0

always
sensor = x,

x'' = - 10*sensor
```

This model captures essentially the same behavior as the first model. It can be seen as representing an idealized sensor that can read the exact position of the mass. Naturally, in practice, converting a physical quantity (such as a process) into a signal representing this value on a wire is a non-trivial process. In fact, it is generally impossible to capture such a quantity exactly and without delay. So, both models should be viewed as representing a highly idealized system.

5.8 Modeling Quantization

Now we refine our model to reflect the reality that sensed values are generally represented in a quantized manner. We can model the quantization process as a continual process of trying to "track" the continuous value by means of a "sensor value" that can only be changed by fixed "quanta." The following model captures this tracking process:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0

always
if ((sensor+0.03)<x)
then sensor+ =sensor+0.3

else if ((sensor-0.3)>x)
then sensor+ =sensor-0.3

noelse,

x'' = -10*sensor
```

Here we have replaced the simple `sensor = x` relation with a more involved relation that moves the sensor values in steps of 0.3 in such a way that they are never more than 0.3 away from the actual value. This means that, while it is possible that there is an error in sensing, in terms of the difference between the actual value of x and our representation of the value of x, this error is bounded. This is representative of what happens whenever we have to quantize a value. The result of this quantization propagates throughout the system. First, the value of the sensor only changes when the value of x has changed to be far enough from the current value of the sensor to trigger a change. Second, it is worth noting that this also means that the value of x'' will only change when the value of the sensor changes. Thus, there is no real need for us to use a continuous assignment for the relation sensor = x, and we can convert it into purely discrete assignments in the branches of the if-statements. This makes it easier to see that what we have now has more quantization than may be evident at first glance, but it is not strictly necessary. In fact, it is convenient to be able to mix discrete and continuous assignments in the manner above, in order to analyze the effect of quantization and/or discretization at very specific points in what would otherwise be a continuous-time system.

www.dbooks.org

The signal for sensor and x'' will therefore have discontinuities. However, the resulting signal for x' and x will not have such discontinuities, as the integration relation that determines them based on the higher derivatives will smooth out such jumps.

The most significant effect of the quantization can only be seen by simulating the model and observing what happens to the variable x: with the addition of quantization, the system is no longer stable. Intuitively, we can view quantization as having introduced a type of *delay*, whereby the system does not really see the value of the input until that input has sufficiently changed from the last reading. Of course, that change itself takes time, and that is what leads to the delay. Thus, the instability that we see here is very similar to that seen in systems where a delay is introduced. In many situations, we can overcome this instability by improving the operation of our controller to have a more stabilizing effect on the resulting system. For linear systems, we can more precisely quantify the effects of discretization and incorporate them accurately in the design of the entire system. For non-linear systems, more specialized analysis is needed for different kinds of systems.

To quantify the "amount of instability" in this example, we can simply look at the maximum height of the last wave at the end of the simulation. In this case, the last wave has a height of last full peak of x that we see in the simulation. Its height is about 3.4. This can be viewed as a "gain" of 3.4 times in the system's oscillation, because the original system (without quantization) had a signal with a maximum height of only 1.0.

5.9 Modeling Discretization

We now turn to discretization. Discrete sampling of continuous signals can be seen as a way of taking "still photographs" of a moving object. The key to being able to take such samples is to have a mechanism for triggering the recording of such values. This requires a variable (such as the sensor in our previous example) to record the value, and an event to trigger the writing of the value of the external continuous variable into the computational units representation of that variable. Sampling (and discretization) can be *periodic* (occurring with equal gaps between samples) or, more generally, *event-driven*. Periodic sampling can be viewed as event-driven sampling that is triggered by a clock event that occurs periodically. All we need to model this type of sampling is a "bucket" that increases at a fixed rate, and the ability to trigger the sampling when the bucket reaches a certain threshold. The following model represents such a situation, where the threshold (sampling period) is 0.05:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0,

bucket = 0, bucket' = 1

always
bucket' = 1,

if (bucket > 0.05)
then bucket+ = 0, sensor+ = x

noelse,

x'' = -10*sensor
```

Just as with the previous example, the variable sensor now changes with discontinuities, and so does x''. The difference now is that the changes occur at a fixed frequency.

To quantify the amount of "instability" introduced by discretization, we can take note of the height of the last full peak in the simulation, which is 2.6. Again, we can view this as a "gain" in the amount of oscillation in the system, because the original system (without discretization or quantization) had a maximum height of 1.0.

5.10 Detour: Discretization, Sampling Rates, and Loss of Information

It can be difficult to capture all the information in a dense-time signal by a finite number of samples (which we can think of as a discrete-time signal). To convince yourself of this fact, consider any discrete-time signal, and then construct two different dense-time signals that pass through all the same points but are slightly different between any two points of your choosing. Note that it is useful that this difficulty only exists when we consider *all* possible functions; if we are willing to restrict ourselves to certain classes of functions, the situation improves dramatically. For example, we often only need to consider signals that have a maximum frequency component. This restriction can be viewed intuitively as saying that the signal "does not change faster than a certain rate." More technically, it means that the frequency-domain representation of the signal is zero beyond a certain frequency. In many cases this is a very reasonable assumption, because many physical systems can be

www.dbooks.org

viewed as "low-pass" filters that essentially ensure that this requirement is true. For such systems, the *Nyquist–Shannon sampling theorem* has good news for us: If the maximum frequency component in a signal is B, sampling that signal at $2 * B$ is enough to capture all the information in that signal.

Two remarks are necessary in relation to this important and widely cited theorem. The first is that, in the context of building control systems, it is *only* telling us that no information is lost *in the sampling*. It is a different matter to ensure that the computation performed on this signal does not introduce *additional loss*. The fact that no information is lost in the signal does not mean that performing a naive analog of the dense-time computation is the right thing to do to get corresponding behavior. The second remark on the theorem is that it is certainly not the only situation where a system can be reconstructed from bounded-rate samples. There are many instances where, based on different kinds of assumptions about a signal, limited sampling can lead to significant or complete information about the signal.

5.11 The Effects of Quantization and Discretization Easily Compound

So far we have considered quantization and discretization separately, but in a setting where we use a digital computer to implement a controller, we have both effects. It is easy to model the effect of doing both things at the same time as follows:

```
model Main (simulator) =

initially
x = 1, x' = 0, x' '= 0,

sensor = 0,

bucket = 0, bucket' = 1

always
bucket' = 1,

if (bucket>0.05)
then if ((sensor+0.3)<x)
then sensor+ =sensor+0.3

else if ((sensor-0.3)>x)
then sensor+ =sensor-0.3

else bucket+ = 0
```

```
noelse,

x'' = -10*sensor
```

It is highly instructive to note that the gain now becomes 7.8, which is noticeably larger than it was for just the quantization or discretization effects taken separately. This example reminds us of the importance of taking into account both quantization and discretization in the design of a system. It also illustrates the importance of ensuring that actual implementations of a control system on a digital computer are able to realize the sampling rate (or rates) upon which the design of a particular Cyber-Physical System is based.

5.12 Chapter Highlights

1. Modeling Computational Systems
   (a) We started from a continuous substrate
   (b) Introduced hybrid systems
   (c) Most computational systems today are not "continuous"
      - Discretized
      - Quantized
2. Quantization
   (a) To represent at zero's and one's
      - How computers work
        – Basic gates
        – These components are really analog!
   (b) Why do we work with zeros and ones?
      - More reliable than analog computers
      - Components can be much cheaper, much smaller
      - Components can be composed more easily
3. Discretization
   (a) To work at discrete time steps
   (b) Why do we work at discrete time steps?
      - Timing is easier to analyze
        – Avoid race conditions
        – Avoid complexities with feedback
4. Effect on stability of control systems
   (a) Discretization
   (b) Quantization

www.dbooks.org

(c) How can we manage this?
- Pick resolution
- Pick sampling rate
- Tune gains

5.13 Study Problems

1. Extend the circuit described in Section 5.4 with an extra input D that causes the output C to become 1 when that input has the value of 1. You may choose up to two additional gates to use.
2. Extend the circuit described in Section 5.4 with an extra input CLK that causes the output C to take whatever value the input has when the input CLK is 1. You can assume that the input A does not change during the time that the input CLK has the value of 1. When the input CLK is 0, the output stays the same independently of the behavior of input A.
3. Run the model of Section 5.7 in Acumen and confirm that it produces the oscillatory behavior described above. Determine the period of the signal based on the Acumen simulation. Assuming that the signal is a cosine wave, confirm your determination about the period by substituting it into the equation and checking the result.
4. There is one difference between the sensor signal and the x signal in the last model of Section 5.7. Can you spot it?
5. In Section 5.8 there is a claim that "the resulting signal for x' and x will not have such discontinuities, as the integration relation that determines them based on the higher derivatives will smooth out such jumps." Confirm these observations by running the model appearing before this claim in Acumen. Modify the code to determine the *shortest time* between two different changes of value to the variable sensor. Hint: It is OK to introduce your own timer into the model to compute this. Also, you only need to compute the shortest time for the transitions that actually occur within the duration of the simulation.
6. Modify the last model in Section 5.9 to determine the size of the biggest jump in the value of the sensor during the simulation.
7. Modify the controller in Section 5.11 to damp the system by taking into account the speed of the mass in addition to its position. You can use the variable x' on the right-hand side of the equation for x'' initially. However, keep in mind that x' cannot "magically" appear inside a digital computer. To address this problem, your final model should include a method for computing an estimate of speed based only on the value of the variable sensor.

5.14 Lab: Stability Exercises

The purpose of this lab is to review and expand on the discussions in this chapter on quantization and discretization with an emphasis on connecting it to what we have learned so far about dynamical systems and control. Working through the activities of the lab will prepare us for the issues that need to be addressed by the upcoming project activities, and more importantly, with how to have a basic appreciation of the effects of quantization and discretization on cyber-physical systems.

In this chapter we have already seen the following model:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = -10,

always

x'' = -10 * x
```

This is a valuable model for us because it can be viewed as a classic example of a control system with negative feedback, and where the resulting behavior is critically stable, that is, just in between being stable and unstable.

Simulate this model to confirm that the result is that x is oscillating around zero.

We also noted that the following variant is essentially the same, but with a variable `sensor` introduced to indicate what we view as the sensor input to the controller expressed by the equation:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0

always

sensor = x,

x'' = -10 * sensor
```

This variant is a good starting point to explore how to represent what happens to a signal as it is measured and passed on to the process that computes the control signal. Note that by simply setting the sensor signal to the quantity we want to measure we are making an assumption that the sensor is highly idealized. It will take additional modeling and transformation of this mapping to have a more accurate representation of what happens in reality when a signal is measured.

www.dbooks.org

Quantization results from the fact that we are using machines that use bits as discrete representations of values. We need not worry about the details of how such representations are realized to model quantization in a simple fashion, and to explore its effect on a basic control system such as the one we are considering here. The following model illustrates a basic method for modeling quantization:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0

always

if (sensor + 0.3) < x
then sensor+ = sensor + 0.3

elseif (sensor - 0.3)>x
then sensor+ = sensor - 0.3

noelse,

x'' = -10 * sensor
```

Notice that we have removed the direct couple between `sensor` and `x` which was provided by one equation in the previous model, and replace it by an `if` statement. This statement compares the value of `sensor` to the value of `x` and corrects any large difference by incrementing or decrementing `sensor` by discrete steps to get it as close as possible to the value of `x`. This is not typically how sensors work, but it illustrates the ease with which quantization can be modeled.

Before simulating this model, write down a description of what the `sensor` will look like. Also, write down a description of the impact that you expect this quantization will have on the overall system behavior. Then run the simulation. Were your expectations accurate? Were there aspects of the simulation plot that you did not expect or did not write down ahead of time?

If we consider the plots resulting from this simulation, we will notice that in 10 s (that is, by the end of the default simulation time in Acumen) the amplitude of the oscillation has grown significantly. Thus, this example illustrates that quantization can have a destabilizing effect on a control system. Before investigating any techniques for mitigating this effect, it is useful to make a mental note of the fact that this is only an example. It is quite conceivable that quantization could also have a stabilizing effect on systems.

Now let us go back to our template model (the one where we first introduced the variable `sensor`) and consider how we can model quantization. There are several ways in which this can be done, including updating sampled values based on changes

in the values being sampled, which can lead to an event-based model. Here we will illustrate quantization based on a clock. The following model gives an example of how this can be achieved:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0,
bucket = 0, bucket' = 1

always

if bucket>0.005
then bucket += 0,
     sensor += x

noelse,

bucket' = 1,

x'' = -10 * sensor
```

Here we have the variable `bucket` acting as a quantity that gets filled at a constant rate. When the value of bucket passes `0.005`, it is reset to zero (thus emptied) and the value of `x` is sampled by copying it into `sensor`.

Notice that here we are quantizing only the sensing and not necessarily the calculation of the control equation of the update of the control output. For simplicity, they are modeled by a simple equation that holds all the time. However, since it is a simple linear calculation that depends solely on the value of `sensor`, it also gets revised whenever `sensor` is revised.

Before simulating the model, write down what you expect to see in the signals for `bucket` and `sensor`, as well as a description of what you expect will be the impact of discretization on the overall system. Simulate the model above, and then make a note of whether all your expectations were met, and whether there were any unexpected features in the result of the simulation.

If we consider the resulting plots from the last model we will notice that by the end of the first 10 s the amplitude of the oscillation goes up significantly, which is roughly the same as what we saw with quantization. The closeness in the magnitude of both gains is a coincidence. The key point is that for this system, both quantization and discretization have a destabilizing effect.

www.dbooks.org

Naturally, we may be interested in modeling both effects at the same time, which is achieved by the following model:

```
model Main (simulator) =

initially
x = 1, x' = 0, x'' = 0,

sensor = 0,
bucket = 0, bucket' = 1

always

if (bucket > 0.005)
then if ((sensor+0.3)<x)
then sensor+ = sensor+0.3

elseif ((sensor-0.3)>x)
then sensor+ = sensor - 0.3

else bucket = 0

noelse,

bucket' = 1,

x'' = -10 * sensor
```

As may be expected, combining both effects also leads to an increase in destabilization of the system.

A natural way to reduce these effects is to increase precision and sampling frequency. In general, however, these can be directly proportional to the cost of the computational resources needed to perform the computation.

An alternative approach that can have lower costs is to introduce damping into the system. If this can be done, it can stabilize the system, albeit possibly at the cost of accumulating more delays into the overall system behavior.

To introduce damping, however, there is another important challenge: In the examples we have seen before, this requires the use of speed. But we are in a situation where we are modeling sensing and have no direct access to the quantity itself, not to mention the rate at which it changes.

To estimate speed, extend the model with two variables, `sensor_last` and `xp`. Extend the model so that `sensor_last` always keeps the last update to `sensor`. Then, using the two sensor readings and the time-step for sampling, make an estimate of the speed. Use this estimate to update the controller to introduce a damping effect. Find an appropriate gain constant so that the system returns to being (approximately) critically stable.

Compare your estimate of speed to the actual speed variable `x'`. Are there consistent differences between your estimate and the actual speed? Can this lead to problems for the system? Are there well-motivated ways in which you can improve your speed estimation method?

5.15 Project: Quantization and Discretization

The goal of this project's activity is to help you understand the use of estimators to improve the quality of data from the sensor. Your task is to take the player developed from last week and observe how its behavior changes when you switch to a new model that involves discretization and quantization effects for both actuation and sensing signals. With this model, you should find that the controller that you had previously developed under the assumption of more idealized sensing does not quite work as well any more. You want to develop an improved player that is able to handle this increased level of realism in the model. To address this, the new variable `estimate_ballv` can be used to overcome the discretization and quantization effect of the velocity of the ball. You will also need to estimate the ball's velocity in a discrete manner.

In the context of this project's activity it will be useful to reflect on how you have overcome this new challenge, paying particular attention to the design of your controller and the various state estimation methods being used by the player.

As usual, feel free to continue to develop all the capabilities of your player in creative ways. Neglecting any one aspect (such as the effects of quantization and discretization) would put you at a disadvantage, but to really excel you need to take a holistic approach to the development of your player, and to make sure that you learn as much as possible from past experiences.

Incidentally, the approach that we have encouraged you take in developing this project is to start with a high-level model that allows you to focus on some big picture questions and then gradually go to lower level models that allow you to address the lower level challenges as well. Reflect on whether this has been a helpful strategy for you, and whether other approaches could have worked better, or at least equally well. In particular, would it have been helpful to expose all of the problems that you need to address from the outset?

5.16 To Probe Further

- General introduction on real-time systems and embedded systems.
- Article on Nyquist–Shannon Sampling Theorem.
- Washington Post article before software patents are revisited by US Supreme Court.
- Watch Grace Hopper's elegant explanation of what is a nanosecond.

www.dbooks.org

- Find out about the conference named after Grace Hopper, and think about what you may be able to contribute there.
- Watch the video recording of Edward A. Lee's lecture on Heterogeneous Actor Models in the Halmstad Colloquium series. You may also want to checkout Synchronous Data Flow after watching this lecture.
- Article on the Philosophy of Computer Science.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Chapter 6
Coordinate Transformation (Robot Arm)

What do we do if our physical system is not perfectly matched to our coordinate system? This chapter takes a closer look at one aspect of modeling physical systems, namely modeling the most basic robot that has rotational joints. A key concept needed for approaching this and similar problems is coordinate transformation. We focus in particular on mapping Cartesian to spherical coordinates, and vice versa. In addition to refreshing our skills in computing derivatives, special attention must be paid to singularities.

6.1 Introduction

Because the three dimensions are independent, it is convenient analytically to work with forces in a Cartesian coordinate system. However, robots that actuate a mass in a Cartesian manner would be bigger and more expensive than necessary. In particular, to build a robot that can reach all points in a $1\,\mathrm{m} \times 1\,\mathrm{m} \times 1\,\mathrm{m}$ model, we would need machinery that is at least 3 m in length, and that would require substantial space to move. A robot that can rotate and extend needs only to be about $1.73\,\mathrm{m}$ long at most; almost half the size. In situations where we have more complex requirements, a multi-link robot can have significant advantages in terms of reach and flexibility, since it can more effectively approximate the shortest feasible path between one point and another. As such, robots with rotational joints and with multiple links have important applications in practice.

To design such robots, we need to have the analytical tools required to model them mathematically. The tools will allow us to determine the location of the links (and joints) at different times, and will also help us to determine the rotational speeds and angular accelerations needed at various joints. One complication to note about the mapping from Cartesian to polar is that there is a singularity (or at least ambiguity) when we are at the origin. This is a technical point that is not so easy to get around, and will require constant attention when we are working with such mappings.

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_6

www.dbooks.org

Another key complication to keep in mind is that mapping *changes* such as speed or acceleration, from Cartesian to polar, depends not only on the magnitude of the change itself but also on the absolute position. Fortunately, this is a complication that can be managed pretty well by using analytical differentiation. In particular, by being careful about independent and dependent variables, and using the chain rule, we can compute closed-form representations of the mapping of changes (or derivatives) from one coordinate system to the other. In doing so, the key analytical skill to keep in mind is looking for patterns in the results of each differentiation step, and to replace any repeating variable by a single variable that we have seen before. This technique of avoiding duplicate expressions keeps derivations manageable and it is rather easy to do. If we skip this step, however, things can easily become complicated.

6.2 Coordinate Transformation

Let us consider the following illustration, depicting the representation of a point in both Cartesian and polar coordinates:

Using geometry, we know that we can compute the 2D Cartesian representation from the polar one as follows:

$$x = l\cos(\theta), \tag{6.1}$$

$$y = l\sin(\theta). \tag{6.2}$$

This mapping is well-behaved, in that we have a unique value of the Cartesian coordinates for any given value of the polar coordinates. Thus, there are no singularities in this mapping. A singularity is a point where a function is not defined. In general, it is easier to work with functions that have no singularities. A function defined for all possible inputs (of its input type) is called a total function. For example,

$$f(x) = 2x$$

is a total function. A function that is not defined for some inputs (of its input type) is called a partial function. For example, $g(x) = 1/x$ is not defined when $x = 0$, so, it is a partial function. Note that we refer to the two functions above as one mapping. We can also think of them as one function from pairs to pairs.

We can now compute the derivatives in Cartesian coordinates by differentiating both sides of the equation above. The key thing to note in this case is that we must make extensive use of the chain rule of differentiation, but without being able to simplify it fully.

The chain rule can be expressed in two ways. If we use f and g to represent functions, then we can write it concisely as follows:

$$(f \; o \; g)' \; = \; (f' \; o \; g)g'.$$

A more familiar form might be the following:

$$\frac{dv}{dt} = \frac{dv}{du}\frac{du}{dt},$$

or,

$$v' \; = \; \frac{dv}{du}u'.$$

Writing it this way helps us see that, if we cannot compute v' directly, then we can still write a useful expression for it if u' is a meaningful quantity and we are able to explicitly differentiate v with respect to u. For our example in (6.1) and (6.2), this means that we can differentiate the expressions to obtain the following:

$$\begin{aligned} x' &= \; (-l \cdot \sin(\theta)) \cdot \theta' \; + \; l' \cdot \cos(\theta), \\ y' &= (l \cdot \cos(\theta)) \cdot \theta' + l' \cdot \sin(\theta). \end{aligned}$$

Now we come to the rather important step of looking for patterns in this result, and reducing duplication of expressions as much as possible. This process is important both to help us understand the *meaning* of the expression that we just computed and also to keep the expression itself *small.* The latter is especially important if we need to differentiate it again, which is the case, for example, if we eventually want to compute the mapping of accelerations. In our example here, we notice that there are terms corresponding directly to the definition of y and x. Thus, we can simplify the expression for derivatives as follows:

$$\begin{aligned} x' &= \; -y \cdot \theta' \; + \; l' \cdot \cos(\theta), \\ y' &= x \cdot \theta' + l' \cdot \sin(\theta). \end{aligned}$$

Exercise 6.1 Compute the expression for the second derivative of x and y.

Now we can consider the conversion in the opposite direction, namely going from 2D Cartesian to 2D polar coordinates. We determine the polar coordinates in terms of the Cartesian coordinates as follows:

$$\begin{aligned} l &= \sqrt{x^2 + y^2}, \\ \theta &= \sin^{-1}(y/l). \end{aligned}$$

The first point to note is that this transformation has a singularity. In particular, if $l = 0$, then the result of the division is not defined.

www.dbooks.org

The second point to note is that, unlike the first one, this transformation is not unique. In particular, there are many other transformations that could give us a result that can be mapped back (by the first transformation) to the same result. An example would be

$$l = -\sqrt{x^2 + y^2},$$
$$\theta = \sin^{-1}(y/l) + \pi,$$

or

$$l = \sqrt{x^2 + y^2},$$
$$\theta = \sin^{-1}(y/l) + 2\pi K,$$

where K is any integer.

The difficulty introduced by these issues is that we have to carry around with us the requirement that we cannot have $l = 0$. Other than that, working out how to map the derivatives in Cartesian space back to polar is no different from doing it in the other direction.

Exercise 6.2 Compute the expression for the first and second derivatives of l and θ. The situation is very similar when we go to three dimensions as follows:

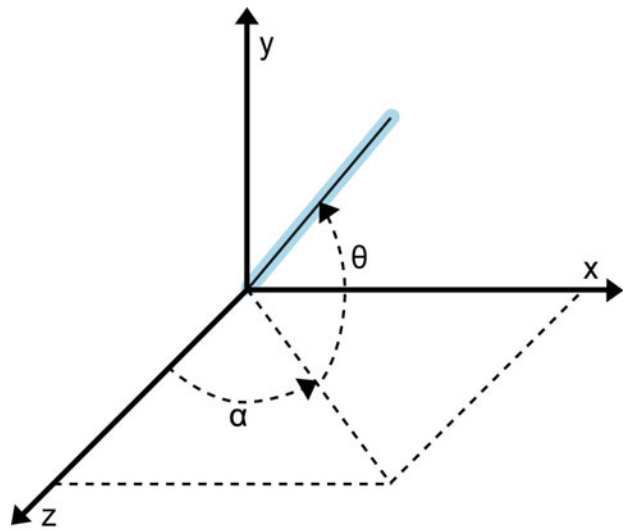


Fig. 6.1 3D projections between polar and Cartesian coordinates

The main activity of this chapter's lab is to demonstrate mastery in deriving the Cartesian to spherical (and vice versa) in a 3D setting, such as the one depicted in Figure 6.1.

The study problems illustrate that the same techniques can be used to convert the local (polar) coordinates of a two-link robot (in 2D) to the global (Cartesian) coordinates.

6.3 Chapter Highlights

1. From Translational to Rotational Joints
   (a) Throughout the project, we have progressed gradually
      - First, we had to determine speeds—this gave us the planning level
      - Second, we determined accelerations (with feedback)—this gave us a simple model of actuation
      - Third, we tuned things to deal with sensing and actuation details
      - Now, we want to figure out how to build an actual robot
        – and that will mean how to work with a single-link robot

      We will focus on 2D, but 3D is the subject of the project
2. From Polar to Cartesian in 2D
   (a) Basic equation
      - Using geometry
      - Note that there are no singularities
      - First derivative in detail
      - Second derivative

   (b) Note about forces and torques
3. From Cartesian to Polar in 2D
   (a) Basic equation
      - Still using geometry
      - Note the singularities
      - Derivative of arcsin(x) is $1/\sqrt{(1-x^2)}$

   (b) Beyond a single-link
      - A two-link example
      - Defining positions equations using geometry
      - Inversion gets more complicated
      - Calculating derivatives

6.4 Study Problems

This study problem focuses on practicing basic analytical differentiation.

1. Assume that a, b, x, y are dependent variables that vary as a function of the independent variable t. Consider further the following relations:

www.dbooks.org

$a = \sqrt{x^2 + \sin(y/x)}$,
$b = a + x * y/(\sin(a))$.
Assume further that the notation x' means dx/dt. Compute the following values:

(a) da/dt in terms x, x', y, and y'.
(b) db/dt in terms of a, a', x, x', y, y'.

Show all your intermediate work. You may use notation "del a/ del b" for the partial derivative of "a" with respect to "b."

2. Consider the simple two-link system shown in Figure 6.2.

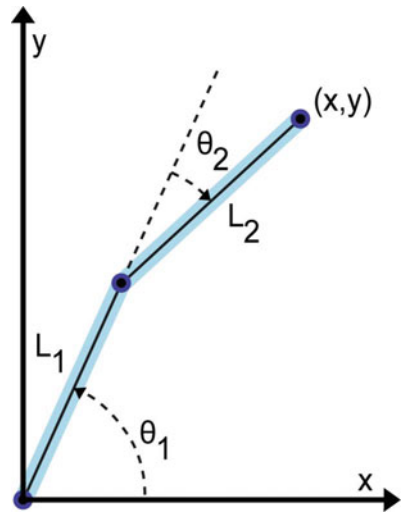


Fig. 6.2 A simple two-link system

(a) Write the formula for (x, y) in terms of the other variables in the diagram.
(b) Assuming ALL variables can change with time, write down the formula for the first derivative of x and the first derivative of y.
(c) Assuming that ONLY the length variables (not the angle variables) can change, write down the equations for the second derivative of x and the second derivative of y.

3. Consider the mechanism in Figure 6.3

(a) Assume that the length of the first link is L, and that the length of the second link is L. Assume further that the end of the second arm is a point at (x, y) with respect to the coordinate system (x_0, y_0). Write an expression for the value of x and y in terms of L, θ_1, and θ_2.

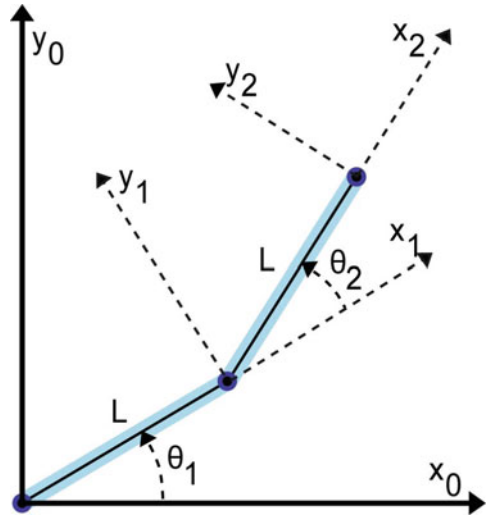


Fig. 6.3 A two-link mechanism

(b) Write an expression for the derivative of x in terms of θ_1 and θ_2. Do the same for y.

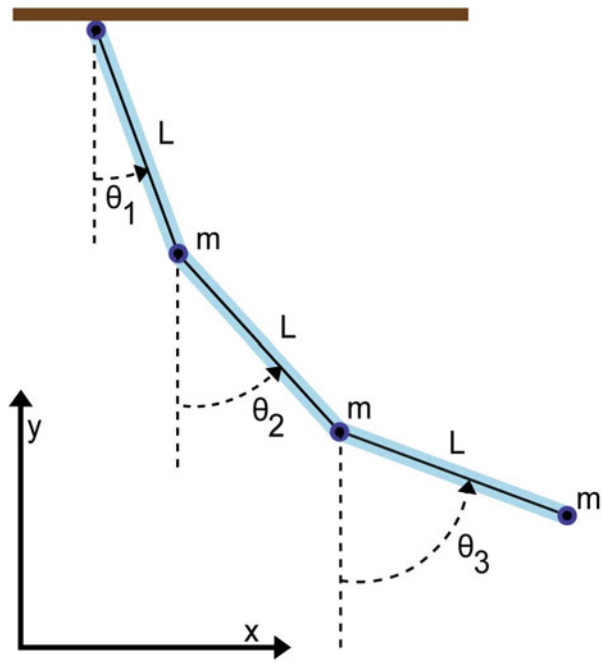


Fig. 6.4 A triple-pendulum mechanism

www.dbooks.org

4. Consider the triple-pendulum mechanism shown in Figure 6.4

   (a) Assume that the x axis points to the **right,** the y axis points **up**, and the origin is at the **bottom** of the diagram (the exact position is irrelevant for this problem). Assume the first point from the base is at (x_1, y_1), the second is at (x_2, y_2), and the third is at (x_3, y_3). Write the expression for (x_3, y_3) in terms of θ_1, θ_2, θ_3, and l. Your answer should **not** expand x_1, y_1, x_2, and y_2.
   (b) Write an expression for the **second** derivative of y_3 in terms of θ_1, θ_2, θ_3, and l.

5. Consider the mechanism shown in Figure 6.5:

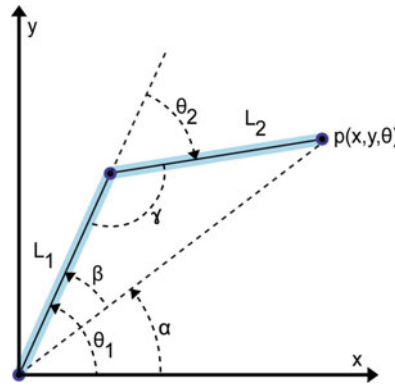


Fig. 6.5 A two-link mechanism

   The positive x axis points to the **right,** the positive y axis points **up**, and the origin is at the **bottom left** of the diagram.

   (a) Write the expression for x and y for point $p(x, y, \theta)$ in terms of θ_1, θ_2, l_1, and l_2.
   (b) Let l_3 be the distance of point $p(x, y, \theta)$ from the origin. Write an expression for the α and l_3 in terms of x and y.
   (c) Write an expression for the **second** derivative of both x and y in terms of the other variables and their derivatives. Make sure that your answer provides equations that clearly define these values in terms of θ_1, θ_2, l_1, and l_2 and/or their derivatives.

6. Go back to the first assignment and consider your explanation of the challenges. Write a single page containing the following:

   (a) Your original submission from the first assignment, with title Original Submission. This part should be at most one page.
   (b) A first revision of this submission is based on what you know now. In this revision, your goal is to simply say what you were trying to say at the

beginning of the semester, but to say it more clearly using the concepts that you now know. For example, do not add any new challenges. This revision is also a good opportunity to use proper terminology that you may have not been familiar with at the start of the book. This part should be at most one page.

(c) Based on this first revision, make a second one that explains what you see as the most important challenges from your point of view now. In addition, it should explain how you would go about building such a robot. This part can be up to two pages.

6.5 Lab: Coordinate Transformations

The purpose of this lab is to connect the theoretical discussion of coordinate transformation already seen in this chapter with the concrete use in the context of a simulation of a particular control problem.

Let us consider a situation where there is a ball being moved in a circular orbit with a single-arm robot. There are only two degrees of freedom: The angle around the center of motion and the length (extension of the arm). This can be represented using the following model:

```
model Main(simulator) =
initially

// Red (r):  The target ball
  theta = 0, theta' = 0,
  l = 1, l' = 0,
  x_r = 1, y_r = 0,// This is the target

  _3D = ()
always

// Red(r): The target ball

  theta' = 1, l' = 0.5*sin(theta),

   x_r = l * cos(theta), y_r = l * sin(theta),

  _3D = (Sphere center= (x_r, 0, y_r)
                size= 0.1
                color= (1, 0, 0)
                rotation= (0,0,0)).
```

www.dbooks.org

To facilitate visualization in this example, we have also used the `_3D` mechanism for generating animations. As we add more objects you should also extend that statement to visualize those objects as well.

Note that here we chose to give the red ball the dynamics of having a constant angular velocity `theta'` but a fluctuating radial speed `l'`. This makes it move in an almost-but-not-quite circular orbit.

Simulate the model to confirm that it behaves as described. Inspect the plots to familiarize yourself with the behavior of the different variables involved.

Now imagine that we would like to build a controller for a robot arm that should follow this path. There are in general many choices for how to do this, but imagine that we have to build to separate controllers, one for the angle of the arm and one for the extension of the arm. This means that the angular actuation can only depend on angles and the extension actuation can only depend on extensions. For simplicity, we decide to use PID control. The following controller can be modeled as follows:

```
model Main(simulator) =
initially

// Red (r):  The target ball
  theta = 0, theta' = 0,
  l = 1, l' = 0,
  x_r = 1, y_r = 0,// This is the target

// Green (g):  The spherically controlled ball

  x_g  = 0, y_g  = 0,
    theta_g = 0, theta_g' = 0, theta_g'' = 0,
  l_g = 1,  l_g' = 0, l_g'' = 0,

// Control gains
  k_p = 5, k_d = 1,

  _3D = ()
always

// Red(r): The target ball

  theta' = 1, l' = 0.5*sin(theta),

   x_r = l * cos(theta), y_r = l * sin(theta),

// Green(g): The spherically-controlled ball

  theta_g'' = k_p * (theta - theta_g) - k_d * theta_g',
```

```
  l_g''      = k_p * (l - l_g) - k_d * l_g',

  x_g = l_g * cos(theta_g),
  y_g = l_g * sin(theta_g),

  _3D = (Sphere center= (x_r, 0, y_r)
                size= 0.1
                color= (1, 0, 0)
                rotation= (0,0,0),
         Sphere center= (x_g, 0, y_g)
                size= 0.1
                color= (0, 1, 0)
                rotation= (0,0,0)).
```

Simulate this model to confirm that it behaves as described. Note that the performance of the controller can be improved by tuning the gains, but for our purposes smaller gains make it easier to have a distance between the two balls, and to remember which one is which.

Now imagine a situation where our robot was upgraded, and the new robot was Cartesian rather that polar. That is, the new robot consisted of two orthogonal belts, one moved an entire platform in the x direction, and another sitting on this platform could be moved in the y direction. The new robot is also more computationally powerful, so, it can calculate trigonometric functions quickly. Our new task is now to calculate the control signals that the controller for the new robot needs to generate so that it acts exactly as the old controller did. In other words, we have to transform all of our signals from polar to Cartesian.

Use the methods that you have learned about in this chapter to convert the above signals for the green ball to signals in polar coordinates. First do this using pen and paper and then map it back to the model. To make room for it in the model, introduce a new object consisting of a blue box as follows:

```
model Main(simulator) =
initially

// Red (r):  The target ball
  theta = 0, theta' = 0,
  l = 1, l' = 0,
  x_r = 1, y_r = 0,// This is the target

// Green (g):  The spherically controlled ball

  x_g  = 0, y_g  = 0,
    theta_g = 0, theta_g' = 0, theta_g'' = 0,
  l_g = 1,  l_g' = 0, l_g'' = 0,
```

www.dbooks.org

```
// Blue (b):  The Cartesian analog

  k_p = 5, k_d = 1,
  x_b  = 1, x_b' = 0, x_b'' = 0,
  y_b  = 0, y_b' = 0, y_b'' = 0,

  _3D = ()
always

// Red(r): The target ball

  theta' = 1, l' = 0.5*sin(theta),

   x_r = l * cos(theta), y_r = l * sin(theta),

// Green(g): The spherically-controlled ball

  theta_g'' = k_p * (theta - theta_g) - k_d * theta_g',
  l_g''     = k_p * (l - l_g) - k_d * l_g',

  x_g = l_g * cos(theta_g),
  y_g = l_g * sin(theta_g),

// Blue(b):  The Cartesian ball

  x_b'' = 0, // Fix me

  y_b'' = 0, // Fix me

  _3D = (Sphere center= (x_r, 0, y_r)
                size= 0.1
                color= (1, 0, 0)
                rotation= (0,0,0),
         Sphere center= (x_g, 0, y_g)
                size= 0.1
                color= (0, 1, 0)
                rotation= (0,0,0),
         Box    center= (x_b, 0, y_b)
                size= (0.14,0.14,0.14)
                color= (0, 0, 1)
                rotation= (0,0,0)).
```

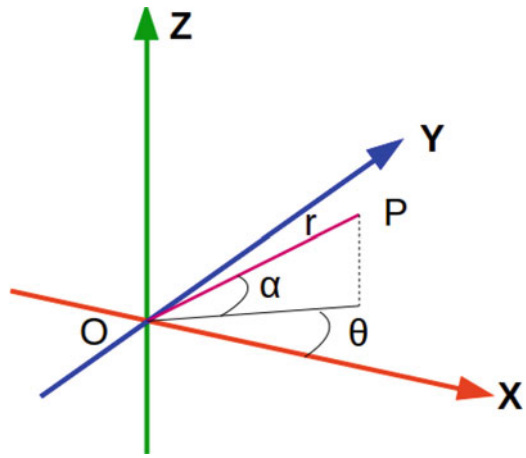
When you have calculated the new controller signals, insert in place of the zero on the lines that say `Fix me`. The formulae that you insert may use all variables of the

green ball and the blue cube, but should not make any direct reference to the red ball.

When you are done, simulate your model. If your answer is correct, your blue cube will coincide exactly at all times with the green ball. This should confirm to you that you have successfully mapped the acceleration signals for the green ball into equivalent acceleration signals for the blue ball.

6.6 Project: Spherical-Actuation for Ping Pong Robot

So far in the project we worked in Cartesian space rather than the space that is most natural with respect to how the components of our robot player move. To remove this simplifying assumption, your task in this project's activity is to convert the accelerations computed by the controller that you built in the previous activity into forces and torques for the single-arm robot that we are working with. The conventions for the coordinate transformation used for this stage of the tournament are represented by the following diagram:



Develop the mathematical equations for this transformation so that you produce the correct actuation signals to your robot. Your change should be made in the player model, and specifically where it says:

```
// You need calculate the following equations according

// to correct coordinate transformation

//r'' = 0 // Wrong value!!!

//Alpha'' = 0 // Wrong value!!!

//theta'' = 0 // Wrong value!!!
```

www.dbooks.org

As part of this activity, you should reflect on how you derived the correct coordinate transformation. Consider whether you think it was useful to ignore these effects initially or if it would have been better to include them from the outset. You can use your old path planning strategies developed for tournament 1 and 2. Also, since this is a project, it is useful to reflect on how you developed your player and what you learned from this experience.

6.7 To Probe Further

- Articles on Polar(2D) and Spherical coordinate (3D) systems
- Video about a boy that gets a 3D-printed prosthetic hand
- Video about a multi-link robot

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Part II
Selected Topics

www.dbooks.org

Chapter 7
Game Theory

Often we are interested in systems that can reason about the results of their actions, and that act to maximize these results. Game theory allows us to predict how such systems will act. This chapter starts by introducing basic concepts from game theory, such as the notions of autonomous choice, utility, rationality, and intelligence. These concepts give us a framework to rigorously identify conditions under which players have incentives to act in manners that can be seen as independent, coordinated, or competitive. Matched to these conditions, respectively, game theory provides us analytical tools for predicting the behavior of players in these situations; namely, strictly dominant (or dominated) strategies, Nash Equilibrium, mixed strategies, and mixed strategy Nash Equilibrium.

7.1 The Role of Game Theory in CPS Design

With the increasing pace of CPS innovation, the number of interacting systems is also increasing. At least two factors have contributed to this trend. First, as devices and systems are endowed with more computational power, their functionality increases. More computation implies more decision-making. Unless we delegate more control to these devices—which means endowing them with more autonomy—we will simply be unable to process all the information needed to make such decisions. Second, because networking of devices provides more opportunities for both controlling and optimizing their performance, it is increasing and, as a result, the interaction between systems is also increasing.

Predicting the behavior of interacting autonomous systems can be challenging, and increasing the number of such systems only makes the situation more so. Game theory is a discipline that provides important analytical and computational methods that can help us in analyzing such systems and predicting their behavior.

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_7

7.2 Games, Players, Strategies, Utilities, and Independent Maximization

For the purposes of this chapter, a *game* consists of two *players*, both having a finite set of *strategies* (or *plays*) to choose from. It also consists of two *utility function* (one for each player) that assign a *utility* to each outcome, where an outcome is determined by the combined choices made by the two players. We assume that the utilities possible for each player are *ordered*, that is, there is always a sense in which one utility is greater than, or greater than or equal to, another utility. We then further assume that both players are *rational*, in that they are trying to choose the play that maximizes the payoff that will be determined by the utility of the combined play of the two players.

The challenge is that each player can choose their own play, but they have no control over the other player's play. Thus we can see each player as trying to *independently maximize* his or her utilities, in a setting where the other player's choice can influence the final utility. We further assume that the players are *intelligent*, in the sense that they are aware of the other player's utilities, and are able to reason about their expected behavior, given the assumption of their rationality.

The general problem posed by a game is "what strategy will the players choose, given the definition of the game?"

7.3 Rationality, Independence and Strictly Dominant (or Dominated) Strategies

Since utility functions capture the quality that the players are trying to maximize, they also induce a pattern of *incentives* for the players to act in a certain way. In fact, when players are working strictly to find the choices that maximize their utility, the utilities can be seen as *forcing them* to act in a certain way. This is not to say that all systems in the real world are working to maximize a specific utility. Rather, in a situation where we can justifiably identify a utility that players are working to maximize, the utility function can be productively used to analyze their choices in this manner.

In the rest of this chapter, we will consider three *patterns of incentive* induced by the utilities. For each of these patterns, we will also introduce a powerful analytical tool that can be used to predict the behavior of players in games that exhibit this pattern. We will start with the simplest and work our way to more challenging ones.

www.dbooks.org

7.3.1 The Independence Pattern

The most elementary utility pattern is independence, which is the situation where one player's utility for a strategy is superior to the others and independent of the other player's choice. In more concrete terms, consider a two-player game where each player can choose between two strategies, A and B. Both players are hungry and can either (A) do nothing or (B) go to the store and buy lunch. If we assume that the actions of the two players are independent, then it is reasonable to believe that each one's utility for (B) is higher than (A) no matter what choice the other player makes. We will call this the "Independence Pattern."

One of the most basic questions we can ask about games is "What would the players do if they were to act rationally?" Here we can define a *rational player* as one that chooses the strategy that maximizes his or her utility. In the case of lunch, the independence of the two players' actions makes it easy to see that choice (B) is rational for both. But how can we answer this question for games where players are not independent? Those cases are, in fact, the primary focus of game theory.

To analyze such games, it is often convenient to represent combined utilities using a table. The rows will represent the first player's choice, and the columns will represent the second player's choice. In each cell, there will be two values, representing, respectively, the utilities of the first and second players. To represent utility patterns, we will indicate lower utilities with the minus sign – and higher utilities with the plus sign +. The lunch example would be represented as follows:

1\2	A	B
A	- -	- +
B	+ -	+ +

The utility pair $u_1 u_2$ such as $+ -$ in the cell at row (B) and column (A) represents, respectively, the positive utility for Player 1 and the negative utility for Player 2, when Player 1 chooses to get lunch (B) and Player 2 chooses to do nothing (A). Ignoring for a moment that we already know the answer to the question, let us consider how we can systematically use this table to determine the rational preference for each player. The way to do this is to analyze the table from the point of view of each of the two players, completely ignoring the *utilities* of the other player (but not their choices). We can visualize this analysis using two tables, one from the point of view of Player 1:

1\2	A	B
A	- *	- *
B	+ *	+ *

and one from the point of view of Player 2:

1\2	A	B
A	* -	* +
B	* -	* +

In both cases, we simply replaced the plus/minus signs by a star sign * to indicate that we are ignoring the other player's utility in that utility pair. Hiding the other player's (irrelevant) utility makes it clear for each player that strategy B is always positive.

What this exercise illustrates is that we only need to know that, for all possible choices that the other player could make, one of our choices will always have greater utility than the other. Visually, this means that, for Player 1, the justification for (B) is based solely on comparisons between utilities in the same column. Going "down" in this table always leads to an improvement in utility, making (B) always a better option for Player 1. Similarly, for Player 2, the justification for (B) is based solely on comparisons between utilities in the same row. Going "right" in this table always leads to an improvement in utility, making (B) always a better option for Player 2.

When an option has this kind of relation to another option, as is the case with options B and A above, respectively, we say that the first *strictly dominates* the second. Strict dominance is an important concept in game theory, as it captures effectively a pattern of reasoning that can be correctly used by a rational system to decide to *exclude* certain choices in favor of other ones that will always be more effective at maximizing the utility of the strategy choice.

Because the rational choice for both players is (B), the "solution" to this game is the play (BB), which represents the choice made by both players, respectively, when we assume that they are rational.

When we get to more complex examples of the Independence Pattern the user will find it helpful to remember two key observations. The first is represented by the two last tables above: the right way to read utility tables from the point of view of each player is to ignore the other player's utilities when you are doing that.

The second observation is more subtle, and requires discerning from the analysis above a pattern that is deeper than the Independence Pattern. In particular, in deciding that (B) is better than (A) from both players' point of view, we do not really need to know that all the lower utilities are equally low, or that all the higher utilities are equally high. The following series of examples will help us understand the significance of this observation.

Example 7.1: A Basic Lunch Returning to our lunch example, we can imagine that the players' utility represents their need to get a meal, and that we will represent one meal's worth by the number 1. In this case, the Independence Pattern can be instantiated to the following concrete game:

1\2	A	B
A	0 0	0 1
B	1 0	1 1

www.dbooks.org

Here the solution is still (BB). Note that a + or − on the original table is a valuation for the respective player's utility for that choice and assuming unchanged choice by the other player.

Example 7.2: An Asymmetric Lunch The way we determined that (BB) is the solution in the case of the Independence Pattern also applies to other—possibly less obvious—situations. For example, it applies in exactly the same way when the utility of one player is greater for his or her lunch than the other player. The following table represents such a situation:

1\2	A	B
A	0 0	0 2
B	1 0	1 2

The utilities of Player 1 still always increase when we go down, and those of Player 2 always increase when we go right. Thus, (BB) is still the solution to this game.

Example 7.3: A Split Lunch The previous example may appear to suggest that the Independence Pattern only applies in cases when the two players' utilities are independent. This would mean that it only applies when there is little or no "real" interaction between the two players. This is not the case. This means that this rather simple analysis can be useful in cases when there is a substantive interaction between players. As a first example of such a situation, consider the case when the two players go to the same supermarket, to buy the same type of lunch, and there is only enough for one person. To avoid the use of fractions in utilities, we will now count getting one lunch as a utility of 2, and half a lunch as a utility of 1. With this convention, we can represent this situation using the following table:

1\2	A	B
A	0 0	0 2
B	2 0	1 1

Does the above analysis still apply to this case? It may be a bit surprising to find out at the answer is yes. One way to see why the analysis still applies is that the first player's utilities still improve when we go down, and the second player's utilities improve when we go right. To make it easier to see that this is the case, we will break up the table into two, each one representing a player's view. Player 1 sees:

1\2	A	B
A	0 *	0 *
B	2 *	1 *

and Player 2 sees:

1\2	A	B
A	*0	*2
B	*0	*1

So, the solution is again (BB). The outcome of (BB) does not change if there is "waste" in sharing the lunch, which would be represented by a higher value in place of the "2" in these tables.

Example 7.4: A Small-Auction Lunch Imagine a situation where both players only have a penny, and when only one goes to buy lunch an auction will view this as low demand, and make available for a penny a great meal with a utility of 5. But when both go at the same time, the auction sees this has high demand, and makes a modest meal with a utility of 1. In this case, our table looks like this:

1\2	A	B
A	0	0 0 5
B	5	0 1 1

Seeing that (BB) is the solution to this game even when the "lunch alone" option has significantly higher utility may lead us to be a bit suspicious of dominant strategy; and it may even lead us to question the way in which we have been analyzing games to determine how their constraints play out in terms of player choices. This is healthy skepticism, because it prepares us for the next example, which pushes the Independence Pattern to the limit.

Example 7.5: A Small-Auction vs. Fridge Lunch Imagine a slightly different situation where both players actually have a readily available lunch in their fridge that they could prepare only if they both chose to stay at home. Imagine further that this lunch was so good that they would both give it a utility of 4. But the auction lunch, which would have a utility of 5 if one of them goes alone, is still slightly better. The following utility table illustrates this situation:

1\2	A	B
A	4	4 0 5
B	5	0 1 1

This situation still fits the Independence Pattern, and the choice of (B) still dominates that of (A) for each player, independent of what the other player chooses to do. Thus, the rational solution to this game is the choice (BB). This is a peculiar outcome, because the utility for both (AA) is higher for *both* players than (BB). So, how can the rational choice for both lead them to (BB)?

We can confirm the pattern's logic by checking that the first player's utilities always increase when we go down, and the second player's utilities always increase when we go right. To see why this down/right pattern really does force any two

www.dbooks.org

rational players to choose (B), it helps to consider what happens if they make any other choice. Making his or her choice independently, Player 1 can only pick one of the two options. From the point of view of Player 1, picking (A) means he or she could end up with a nice lunch if the other player stays in, but they could end up with no lunch if the other player goes out. In contrast, picking (B) means they would get the best lunch if the other stays in, and a passable lunch if the other goes out. So, *whichever choice the other player makes*, choosing (B) improves Player 1's lunch.

This example has the same features as The Prisoners' Dilemma, a classic example in game theory. More background about this game can be found in the article Prisoner's Dilemma.

7.3.2 The Cost of Lacking Communication and Trust Can Be Unbounded

To convince ourselves of the soundness of the above analysis, it is important to realize that each player must make his or her decision independently. This does not mean that this is what any two people in this situation *should do*, rather, it is clarifying how the formal notion of games that we are studying works. We said that we are studying games where each player is trying to maximize utility, and that, for this particular game, the utilities are as shown in the table. We did *not* say anything about players' ability to communicate or their ability to trust each other; as a result, we have to exclude the possibility of the players coordinating, because the ability to communicate and trust are strong assumptions that we cannot make without changing our original problem statement. In fact, a profound lesson that can be drawn from this example is that the costs of lacking communication and/or trust can be unbounded: we can replace 4 and 5 in this example by any pair of arbitrarily large values, and as long as the first is less than the second, rationality and self-interest forces both players to choose (B). Lacking communication and trust can be arbitrarily costly for everyone involved.

7.4 Coordination, Intelligence, and Nash Equilibrium

In the last section we saw the Independence Pattern, where utilities had this form:

1\2	A	B
A	- -	- +
B	+ -	++

We also saw how strict dominance can be used to determine that the rational behavior of two players in such a game has to be (BB). At the same time, we also saw that (BB) may not be the highest *possible* payoff for both players, but it is the highest payoff that they can *guarantee independently*.

The power of strict dominance lies in its usefulness in narrowing down the set of possible rational strategy pairs to a smaller set. However, it will not always be possible to find strictly dominant strategies (or more specifically, strictly *dominated* strategies to exclude). It is therefore useful to consider how to interpret games where there are no strictly dominated strategies, and where we have more than one possible rational outcome.

7.4.1 The Coordination Pattern

Consider a two-player game where each player can choose between two strategies: going to a movie (A) or going to a play (B). Both players only care about being together. Let us call this the Coordination Pattern. The following table represents this pattern:

1\2	A	B
A	++	- -
B	- -	++

It is clear that in this case there is no strictly dominant strategy: For each player, (A) is better if, and only if, the other player chooses (A), and the same holds for (B). We have two cases where there is a win-win choice, (AA) and (BB), but achieving either depends critically on *coordination*.

When we considered our Small-Auction vs. Fridge Lunch example, we noted that communication and mutual trust would have been needed to arrive at a better outcome than that provided by the dominant strategy. Here, there is no dominant strategy at all. The absence of a dominant strategy can be viewed as the absence of a reward to always unilaterally select one strategy versus the other. In such cases, communication is key. However, trust is no longer necessary: the utilities put both players in a situation where (a) it is in their interest *only* to communicate their intent truthfully and (b) once they have shared their intent, the other player is only motivated to act in a manner that is optimal for both of them.

7.4.2 Nash Equilibrium

Note that this type of reasoning reflects *intelligence* on the part of the players, in the sense that it takes into account that they are aware of the other player's utilities and

www.dbooks.org

decision-making process. The observation that we can predict the outcomes of games more precisely when we take into account not only each player's rationality but also their ability to reason about the other player's decision-making process is attributed to John Nash. It is his name that is acknowledged in the term "Nash Equilibrium," which refers to the set of plays (strategy combinations) out of which no player has an incentive to depart unilaterally. In the example above, the set { (AA), (BB) } is the Nash Equilibrium for this game. The game motivates both players to only be in one of these plays. And once they are in one of them, they would only be motivated to move to another one *in coordination* with the other players.

7.4.3 Determining the Nash Equilibrium

With one additional condition, the Nash Equilibrium for a game pattern is simply the set of all strategy combinations with (++) utilities. The extra condition is that each player should only have a plus (+) option as the maximum utility for any one of his strategies. This is the case for both the Lunch and Coordination Patterns. Thus, in the Independence Pattern, the Nash Equilibrium is the set { (BB) }.

Example 7.6: An Asymmetric Four-Strategy Game To check our understanding of this method of computing the Nash Equilibrium set, we will consider a game with four strategies and asymmetric utilities:

1\2	A	B	C	D
A	7 1	2 4	4 8	6 4
B	1 3	3 7	5 6	6 2
C	3 2	4 4	7 5	8 3
D	9 7	2 8	1 9	5 3

When we mark the highest utility in each choice for the first player, we get the following table:

1\2	A	B	C	D
A	7 1	2 4	4 8	6 4
B	1 3	3 7	5 6	6 2
C	3 2	+ 4	+ 5	+ 3
D	+ 7	2 8	1 9	5 3

When we mark the highest utility in each choice for the second player, we get the following table:

1\2	A	B	C	D
A	7 1	2 4	4 +	6 4
B	1 3	3 +	5 6	6 2
C	3 2	4 4	7 +	8 3

1\2	A	B	C	D
D	9 7	2 8	1 +	5 3

Combining all the marks into one table we get the following, where the cells that have utility now marked (++) form the Nash Equilibrium set:

1\2	A	B	C	D
A	7 1	2 4	4 +	6 4
B	1 3	3 +	5 6	6 2
C	3 2	+ 4	+ +	+ 3
D	+ 7	2 8	1 +	5 3

As this table shows, the Nash Equilibrium set is { (CC) }. Thus, if both players reason rationally, taking into account the other player's utilities as options, the first player would choose (C) and the second would play (C). These are the choices that each player can make independently *and* secure the maximum possible payoff, given the utilities for the different choices for both players.

7.4.4 Eliminating Strictly Dominated Strategies Preserves Nash Equilibria

In games where there are a large number of possible strategies, it is useful to remove from consideration (or eliminate) strategies that a rational player would never choose. Strict dominance gives us just the right tool for doing so, as any strictly dominated strategy can be safely eliminated in this manner. What is more, eliminating one choice for one play can reveal other dominated strategies for the other player (since they are also intelligent, and can determine for themselves that the first player would never play that strategy).

This technique is synergistic with the notion of Nash Equilibria: eliminating strictly dominated choices does not remove any elements of the Nash Equilibrium of a game.

Exercise 7.1 Remove strictly dominated strategies from the game presented in this last example. Repeat this process until there are no more strictly dominated strategies. Draw the table for the reduced game. Once you have done so, determine the Nash Equilibrium for the reduced game.

www.dbooks.org

7.5 Competitiveness, Privacy, Mixed Strategies

So far, we have seen an example where strict dominance alone can be used to determine how two rational players will behave (the Independence Pattern), and one where it cannot be applied (the Coordination Pattern). In the latter case, we were able to use the idea of the Nash Equilibrium to determine the set of plays (strategy pairs) that the two players would be simultaneous motivated to choose. There are, however, games where the Nash Equilibrium would have no elements. The following table represents an example game pattern:

1\2	A	B
A	-	+
B	+	-

1\2	A	B
A	+	-
B	-	+

We will call this the Competitive Pattern. In this pattern, there are no win-win plays. In fact, every play is win-lose. In concrete instances of this pattern, if the values of minus and plus in each cell are consistently equal in magnitude but opposite in sign, this is what would be called a *zero-sum-game.*

7.5.1 Mixed Strategy Games

Whereas the Coordination Pattern incentivized both players to communicate truthfully, the Competition Pattern incentivizes them to keep their decisions as private as possible. In fact, if anything, this utility pattern could give each player an incentive to *mislead* the other player.

How would rational players act in such situations?

If we are looking at just a single round of the game, what we can say in this situation is very limited. In fact, all we can do is advise both sides to work hard on keeping their planned strategy secret. But in addition to the usual difficulties in keeping secrets, this situation becomes harder if the game is played multiple times. Then players can simply observe each other and infer the decision-making process of the other side. If one side succeeds in doing this, they can ensure only desirable utilities, and the other only undesirable ones.

This situation gives rise to the idea of a *mixed strategy*, in contrast to what we have discussed so far, which was a *pure strategy*. To play with a pure strategy is simply to select one of the possible strategies. To play with a *mixed strategy* is to select a probability distribution and use it to select from among all the available strategies. As long we are able to make random choices, this can be an effective way to mitigate the risk of being subjugated by the other player. The key to doing so effectively becomes the selection of the right distribution.

7.5.2 *Selecting a Mixed Strategy (or, Mixed Strategy Nash Equilibria)*

In selecting the random distribution, each player's goal will, in fact, have to be to reduce the *other player's* incentive to pick a particular strategy. To do this, we (and each player) will have to analyze the other player's *expected* payoff. We will illustrate this concept in more detail when we consider a concrete example.

It is important to note that we cannot select a mixed strategy at the level of game patterns, but rather, must do so within the concrete games. This is different from strict dominance and pure strategy Nash Equilibria, which could be determined at pattern level. The reason for this is that the expected payoff is sensitive to the concrete value of the utility in each situation.

Example 7.7: Feud Consider the following concrete instance of the Competition Pattern:

1\2	A	B
A	1 6	5 5
B	2 7	3 8

If Player 1 is deciding on a mixed strategy, he or she must select a distribution for choosing between (A) and (B). The distribution consists of two probabilities, p_{1A} and p_{1B}, both of which must be values between 0 and 1, and together they must also add up to 1. The two probabilities represent the relative frequency with which, in the long term, Player 1 will choose A and B, respectively.

Now we need to focus on the payoff of the second player in the case of each play by Player 1. If Player 1 plays A, then Player 2 will choose the A to maximize their outcome (which will have value 6). If Player 1 plays B, then Player 2 will choose B (which will have value 8). What Player 1 can do through its choice of distribution is to equate the *expected payoff* for the second player so that it is equal in the cases of Player 2's playing (A) or (B). The expected value for each of Player 2's options is determined by summing the product of utilities and probabilities in each case. For Player 2's option (A), that would be

$$E(2A) = 6p_{1A} + 7p_{1B},$$

and for Player 2's option (B), that would be

$$E(2B) = 5p_{1A} + 8p_{1B}.$$

If we want these two expected values to be equal, then we want to solve for $E(2A) = E(2B)$, or substituting the right-hand side from the two equations above we get

$$6p_{1A} + 7p_{1B} = 5p_{1A} + 8p_{1B}.$$

www.dbooks.org

This is one equation with two unknowns, which means we need another equation. The equation we have is $p_{1A} + p_{1B} = 1$, from which we can determine that $p_{1B} = 1 - p_{1A}$. We can use that to replace all p_{1B}'s in the above equation by a term that only has p_{1A}. This yields the following equation:

$$6p_{1A} + 7(1 - p_{1A}) = 5p_{1A} + 8(1 - p_{1A}).$$

By simplifying, we get

$$7 - p_{1A} = 8 - 3p_{1A}.$$

From which we can determine that $2p_{1A} = 1$, or $p_{1A} = 0.5$, and so, $p_{1B} = 0.5$ as well. Here we got an even split between the two choices, but that is not always the case. In fact, even in this game, Player 2's optimal strategy will not be an even split. But first, to check our answer, let us make sure that these probabilities do ensure that the expected payoff for Player 2 is the same between the two choices. We do that simply by substituting these values in the equations we wrote above

$$E(2A) = 6p_{1A} + 7p_{1B} = 6 \cdot 0.5 + 7 \cdot 0.5 = 3 + 3.5 = 6.5,$$
$$E(2B) = 5p_{1A} + 8p_{1B} = 5 \cdot 0.5 + 8 \cdot 0.5 = 2.5 + 4 = 6.5.$$

So, indeed, our calculations were correct. And if Player 2 knows (or sees) that Player 1 will be making choices according to this distribution, there will be no immediate incentive to choose between the two strategies.

Player 2 still has an incentive to make sure that Player 1 cannot benefit by choosing one strategy over the other. To do so, he or she will make choices based on a random distribution, and will determine two probabilities p_{2A} and p_{2B} analogous to the ones Player 1 used. To make this determination, Player 2 analyzes Player 1's expected payoffs as follows:

When player 1 chooses (A), the payoff is

$$E(1A) = p_{2A} + 5p_{2B}.$$

For Player 1's option (B), that would be

$$E(1B) = 2p_{2A} + 3p_{2B}.$$

Equating both expectations we get

$$p_{2A} + 5p_{2B} = 2p_{2A} + 3p_{2B}.$$

Using the substitution $p_{2B} = 1 - p_{2A}$ we get

$$p_{2A} + 5(1 - p_{2A}) = 2p_{2A} + 3(1 - p_{2A}),$$

which simplifies to

$$p_{2A} + 5(1 - p_{2A}) = 2p_{2A} + 3(1 - p_{2A}),$$

which simplifies to $5 - 4p_{2A} = 3 - p_{2A}$ and then to $2 = 3p_{2A}$, which means that $p_{2A} = 2/3$, and $p_{2B} = 1/3$.

The uneven split in probabilities in this case can be explained as follows: if Player 2 played an even split between (A) and (B), Player 1 would eventually notice, and start playing (A) more often, because, on average, it gives a higher payoff than playing (B). With Player 2 playing even split, Player 1's expected payoff for (A) is $1 \cdot 0.5 + 5 \cdot 0.5 = 3$, which is higher than that for (B), which is $2 \cdot 0.5 + 3 \cdot 0.5 = 2.5$. Even though this might seem like a small difference, if Player 1 notices, he or she will start playing (A) consistently to maximize their expected payoff. In contrast, the probabilities we calculated for Player 2 make Player 1's payoffs be $1 \cdot 2/3 + 5 \cdot 1/3 = 7/3$ and $2 \cdot 2/3 + 3 \cdot 1/3 = 7/3$, giving Player 1 incentive to *both* maximize its expected payoff *and* minimize the chance that the other player can predict the next play.

The distributions (p_{1A}, p_{1A}) and (p_{2A}, p_{2B}), together, constitute the *mixed strategy Nash Equilibrium* for this game.

7.6 Chapter Highlights

1. Game Theory and CPS
   (a) Trends
      - Networking is going up
      - Automation is going up
      - Result: More interaction

   (b) Game theory gives us tools to
      - model self-interest and awareness of others
      - predict and encourage certain outcomes

   (c) A game consists of
      - Players (2)
      - Choices/Strategies (2)
      - Utilities (at least ordered)
      - What choices are made to maximize each ones OWN utility

   (d) Patterns of utilities direct us to different analysis tools

2. Independence Pattern and Dominance
   (a) Basic pattern table (basic example: going out to lunch)
   (b) How to read the table from each player's point of view
   (c) How to determine what's the best decision
      - The concept of a strictly dominant strategy

      Each one can make that decision independent
   (d) Warning: The pattern is not as simple as it may seem!

www.dbooks.org

3. Cooperation Pattern and Equilibria
   (a) Basic pattern (example: going out)
      - No dominance
      - Coordination need
      - Communication valuable
      - Note that these are often sub-patterns
4. Competition Pattern and Mixed Strategies
   (a) Basic pattern (example: two shops that cell lunch)
   (b) No dominance
   (c) No simple equilibrium
   (d) Must not share information!
   (e) Must mix
   (f) Concrete values in utilities become critical!

7.7 Study Problems

1. Calculate the mixed strategy Nash Equilibrium for this concrete game:

1\2	A	B
A	1 6	7 5
B	7 7	3 8

   Be sure to include the four probabilities, and to check that your answer does equalize the expected payoff for the other player.

7.8 To Probe Further

- Videos on topics covered in this chapter (and more) at Game Theory 101
- Video lectures from Summer School on Games and Contracts for CPS Design.
- A Harvard Business Review article on when is competition a good thing?
- The New York Times article mentioning cooperation in the Riddle of the Human Species
- The related topics agent-based systems, agent-based modeling and simulation, and cooperative games.
- Background on the more abstract idea of actors underlying agents
- Perspectives on what are multi-agent systems (MAS) by Sycara
- Business Insider article about Boston Dynamics being acquired by Google in 2013 and CCN article about later acquisition by SoftBank in 2017.
- Paper by Lavalle on Game Theory and Motion Planning

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

www.dbooks.org

Chapter 8
Communications

At a time when we are surrounded by mobile phones and Internet-based services, communication is a central component of virtually every aspect of our life. At the same time, communication itself is a rich and multifaceted notion. For example, we can consider *why* or *how* we communicate. We touch on the first question in the Game Theory chapter. In this chapter, we turn to the second question, and in particular, the fundamental concerns of what constitutes communication, what are the hard limits on telecommunications, how such limits arise, and how we can model their effects.

8.1 Communication, Certainty, Uncertainty, and Belief

We begin by defining some basic concepts in a manner that is as independent as possible of the details of current technologies. There are two reasons for doing so. The first is to clarify the terms used in this chapter and to reduce the chance of misunderstandings. For all the notions discussed here there are alternative interpretations, and a comprehensive treatment of these alternatives is beyond the scope of this book. The second is to settle on notions that we hope are simpler and may last a bit longer than the rapidly changing current technologies, and that may be compatible with the confluence of different technical disciplines.

To communicate is to share information. A speech by the leader of a nation shares information with interested citizens who may be present in person or watching the speech over modern telecommunication infrastructure. A server storing a digital copy of this book shares information with a smartphone or a digital device where it is downloaded over the Internet.

That communication involves sharing is straightforward. That it involves information is more interesting. Information is an abstract notion related to attaining certainty, which, in turn, is firm or absolute belief. For example, consider a value drawn from the set of Booleans $\{True, False\}$. If we are uncertain about this value,

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_8

we hold no belief about it beyond its possible values (which can be viewed as its "type"). When we are certain about the value, we either believe that it is True or believe that it is False. This is a set-based notion of uncertainty, as illustrated in Figure 8.1.

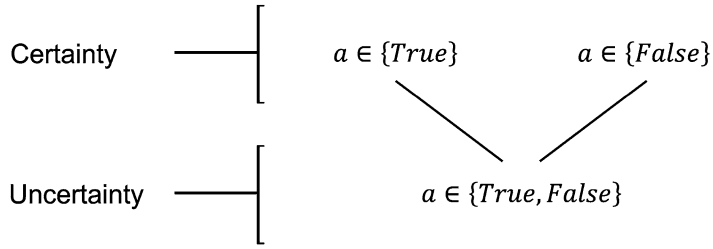


Fig. 8.1 Set-based uncertainty about a Boolean quantity

This is the notion of uncertainty, and in turn, the notion of information, typically addressed by set-based and interval analysis methods, whether applied to numerical computations or programs.

Other notions of uncertainty exist. For example, in cases where it is either impossible or unreasonable to hold absolute belief, we may associate probabilities to different values. For example, we may believe that the value may have a probability of 0.95 of being $True$. This is a distribution-based, probabilistic, or stochastic notion of uncertainty (See Figure 8.2). It is this notion that we typically see being used in information and coding theory and in probabilistic and statistical methods.

For cyber-physical systems we need to consider both notions because we are often interested not only in stochastic guarantees (as is typical in communication theory) but also in deterministic ones.

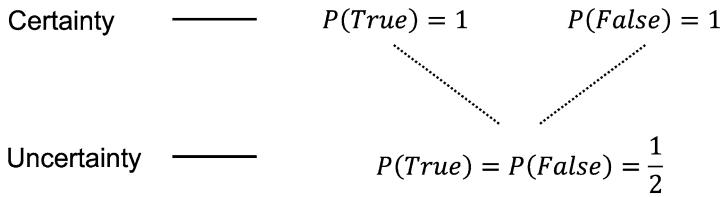


Fig. 8.2 Probabilistic uncertainty about a Boolean quantity

8.2 Messages: From Information to Representation

In contrast to information, which is a notion centered around the belief of an agent, a message is data (a string) that is used to represent and realize the transfer of

www.dbooks.org

information. For example, the message "Tom and Jerry are here" carries information for someone who is unaware of their presence. It also carries no information to someone who is already believes they are here. Accordingly, whether a message carries information is critically dependent on the receiver and, in particular, the receiver's prior beliefs. This is illustrated by the following table:

Message	Belief before	Belief after	Information
Tom & Jerry are here	ϕ	Tom & Jerry are here	Tom & Jerry are here
Tom & Jerry are here	Tom & Jerry are here	Tom & Jerry are here	ϕ
They are here	They are Tom & Jerry	They are Tom & Jerry	Tom & Jerry are here
		Tom & Jerry are here	
They are here	They are Tom & Jerry	They are Tom & Jerry	ϕ
	Tom & Jerry are here	Tom & Jerry are here	

The table also illustrates how correct transmission and message content are different from the information that the message carries. The message "they are here!" also carries the same information for the first receiver if they know the context, but the raw data (the sequence of letters) being transmitted is clearly different.

The following two exercises, one for each notion of uncertainty, provide more examples that illustrate the difference between the message and the information it carries.

Exercise 8.1 Consider an agent that believes that $x + y = z$ where $+$ is addition on the real numbers and that $x \in \{1..2\}$, that is, x is in the set of all real numbers between 1 and 2, inclusive.

1. If it receives a message that $y \in \{3..4\}$, what should it believe about z?
2. If it receives a message that $z \in \{3..4\}$, what should it believe about x?

Exercise 8.2 Consider an agent that believes that $(x \wedge y) = z$ where $\wedge$ is conjunction (AND) on Booleans and that the probability of x being $True$ is 0.5.

1. If it receives a message saying that the same probability holds for y, what should it believe about z?
2. If it receives a message saying that the same probability holds for z, what should it believe about x?

8.3 Belief, Knowledge, and Truth

Knowledge is belief that is true. This is consistent with our intuitive understanding of these notions: In everyday life we would not use the term Knowledge to describe the false belief that "The planet Earth was entirely pink on January 1st 2019." Rather, we would describe it as a misconception or simply a false belief. Similarly, we would say that Jane knows that $1 + 1 = 2$ when she believes that statement, since it is a true statement. We may also say that Jane believes that a variable x has a value of 17. When we say that we leave it open whether her belief is correct. In contrast, when

we say that Jane knows that x has a value of 17, we are also asserting that her belief is true. Thus, even in everyday language we take knowledge to be the intersection of belief and truth (Figure 8.3).

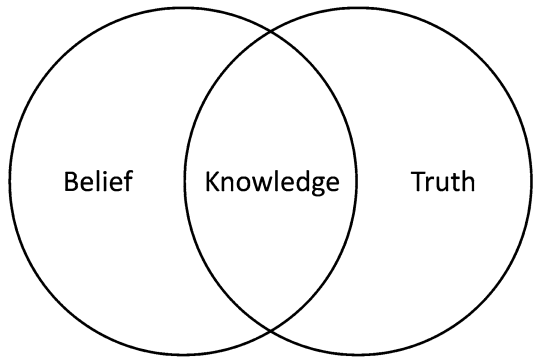


Fig. 8.3 Knowledge is the intersection of belief and truth

The distinctions are important for two reasons. The first is that they can often guide us in comparing different ways to solving a problem. Consider the following exercise:

Exercise 8.3 The length x of some rope is exactly 3.14 m.

1. A camera-based app on a smartphone estimates it to be 3.15. Can we say that the phone has belief or knowledge about the device?
2. What if, instead, the phone estimates the value to be in $\{3..4\}$?

This exercise illustrates that awareness and tracking of the error in a measurement can have a significant impact on the validity of calculations based on this measurement. Now consider the following situation.

Exercise 8.4 The altitude (distance from sea level) x of a ship is varying over time t according to the equation $x(t) = \sin(t)$. An aircraft trying to land on the ship has a device the provides a measurement of the height of the ship. Explain for each of the following cases if the aircraft has belief and/or knowledge:

1. The device provides a perfect and continuous measurement y where $y(t) = x(t)$.
2. The device provides a delayed continuous measurement $y(t) = \sin(t - d)$ where the amount of positive delay d is fixed but unknown.
3. The device provides a delayed continuous measurement $y(t) = \sin(t - d)$ and the value of the delay d.
4. The device provides a delayed continuous measurement $y(t) = \sin(t - d)$, the value of the delay constant d, and the aircraft knows that the value being measured is a periodic signal.
5. The device provides two discrete measurements, $y_1 = \sin(t_1 - d)$, $y_2 = \sin(t_1 + 1 - d)$ the value of the delay constant d, the measurement time $t_1 + 1$, and the aircraft knows that the value being measured is a periodic signal.

www.dbooks.org

This example illustrates the value of not only of keeping track of measurement error but also of keeping track of other effects that may arise with measurement, such as delay, and of knowledge about the nature of the signal being measured.

8.3.1 Broader Implications

The above distinctions help us understand what constitutes correct communication in cyber-physical systems. The distinctions are critical for two main reasons.

First, since truths in general are not affected by what we share or do not share about them, information affects primarily belief, and only affects knowledge to the extent that the change in belief overlaps with truth. Information is thus related to belief but not necessarily to knowledge. Logic and its rules are generally based on the assertion that truths need to be consistent and free of contradiction. This does not apply to beliefs. We generally work to ensure their consistency, but such efforts may well fail.

Second, whereas data processing systems are developed with some particular intent in mind, computations themselves are generally oblivious to this intent. Today, this dichotomy is touched upon quite often in the context of the so-called smart contracts, which are digital contracts that execute automatically. Our intent when writing a contract is based on our beliefs at one time, but smart contracts must run correctly for a very long time. Even if we consider such instruments to be "only" affecting money (money still has tremendous impact on people's livelihood), with increasingly more services that take orders online to perform real-world functions (such as Amazon, Uber, or a wide range of other services), unintended actions from such systems can have tremendous undesirable impact on our life, and at a very large scale.

For these reasons, the importance of maintaining awareness of the real-world meaning and real-world veracity of the information being communicated and manipulated by automated systems cannot be understated. As innovators we have significant responsibility if not legally or socially then ethically towards the programs, controllers, and cyber-physical systems that we develop.

8.4 Carrier Signal, Medium, and Link

Let us now turn our attention away from information back to sharing. In particular, let us consider how sharing is carried out and the characteristics of such processes. For a transmission to occur in the real world, a message needs to be represented and physically transmitted. Transmission occurs using a carrier signal. Often, the carrier is transmitted over a medium. For example, consider the process of handing a cone of ice-cream to a friend (See Figure 8.4). Here, the carrier signal is ice-cream cones, the medium is the space in which it travels between the first and the second

person. The message is whether or not we send a cone. As usual, the information being transmitted depends on the beliefs of the receiver, but one can imagine in this example the message is intended to communicate a positive sentiment. When we consider the communication between you and your friend in this situation, we can think of this entire process as a communication link.

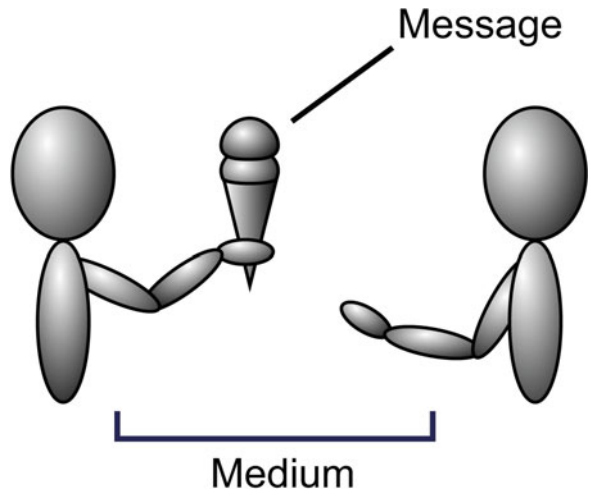


Fig. 8.4 A simple communications channel

There is a wide variety of possible carrier types, and which we can call communication modes. The following table gives some examples.

Mode	Medium	Examples
Transportation	Optional	Postal service. Cell biology. Olfaction
Light	Optional	Lighthouses. Gestures. Sign language. LiFi.
Radio waves	Optional	Cellular. Bluetooth. WiFi.
Electric current	Mostly	Phone lines. Twisted pairs.
Electric potential	Rarely	Across capacitors
Vibration	Necessary	Sound (sonic). Ultrasonic.
Pressure	Necessary	Steering. Pressure modulation.
Temperature	Necessary	Covert communication.

The first three modes require no medium. While traditional treatments of communication will not typically include transportation as a mode of communication, in a cyber-physical setting it can be very useful to consider it as such. It is also useful to recognize that it happens both in man-made and natural systems. For example, the transport of molecules within and between living cells is essential for regulating the processes of a living organism.

www.dbooks.org

Light and radio waves, which are both electromagnetic waves, can be viewed as transportation of photons. Physically, electromagnetic signals are different from what we consider as everyday transportation because photons have both wave and particle properties.

The next three modes typically require a medium, but it is useful to note that it is not always necessary. Electric signals involve the transportation of electrons, which is generally more controlled through a medium. It is possible to move electrons in free space, but managing them in this manner is quite different from designing electric circuits. In addition, traditional electric circuits also involve components such as capacitors and inductors, which involve a discontinuity in the conductive medium. Such gaps can be a medium, in which case they are called dielectrics, or it can be a vacuum. Dielectrics are more common as they would generally help avoid physical contact between the components involved.

The last three modes require a physical medium and cannot exist without it because they are, in effect, changes to the physical state of a given medium. Vibration in general and sound in particular are interesting not only because we as humans have used them since the beginning of time, but also in the cyber-physical setting, they can be related to security. For example, it is known that a malicious agent can implant a virus on computers with no traditional communication ports so as to use cooling fans to export information from such devices.

The last two are examples of more rare modes of communication, but it is useful to be aware of such examples for both cooperative and uncooperative situations. Pressure modulation, for example, has been used by oil exploration companies to send signals from deep in the ground to the surface during drilling, a situation which makes other modes of communication difficult. We are not aware of uses of temperature for communication in cooperative situations, but it is known that temperature can leak significant information about encryption keys when, for example, the temperature of a smart card chip is used to extract information that can be used to find the key.

8.5 Link Characteristics

The wealth of possibilities for communication modes is challenging and inspiring. At the same time, when we want to design specific systems, it is useful to have ways to compare different possible choices. The question of cost is of course always an overarching one. The next question then is how to quantify what is being communicated so that we can compare costs between solutions of similar or comparable performance.

In general, the performance of a communication link is not quantified as a single quantity, but rather, as a composite of different characteristics. Commonly considered characteristics are latency, bandwidth, and various notions of reliability. While ice-cream, of course, is not the only means of communication, it can illustrate some common physical characteristics of links. In the context of cyber-physical systems, the mobility of the communicating entities makes the situation more interesting, as

all of these characteristics can depend on both the relative location of the entities as well as their environment.

8.5.1 Latency

Latency is the time it takes from sending a signal to receiving it. If we imagine that the two people are 50 cm apart and the ice-cream can be moved at a speed of 1 m/s, then it will take 0.5 s to transmit the ice-cream. While for interpersonal communication such delays may be acceptable, for applications such communicating a signal from car brake pedals to wheels, much shorter delays are required.

Clearly, faster transport speeds can lead to shorter latencies. This is an important reason why media such as electric current and electromagnetic waves (light and radio) are popular. Light can move much faster than our ice-cream in this example, in fact, almost 300 million times faster. But we should keep in mind that for any non-zero distance and finite speed, transmission over this distance will experience non-zero delay.

Physics limits the minimum latency more than we may realize at first. In particular, the theory of special relativity suggests that not only is there always a non-zero delay, but there may be an absolute, minimum delay between two objects with a non-zero distance between them. In particular, the theory suggests that it is impossible to travel faster than the speed of light. This means that the shortest time any transmission can take between the two people in our ice-cream example is about 1.67 ns (nanoseconds). Another way of looking at this is that no signal can travel more than about 30 cm in a nanosecond. This constraint is significant in large-scale systems such as communication via satellites or when communicating with someone on the moon. Light takes about 1.3 s to travel between Earth and the moon.

As an aside, for a historic illustration of the significance of understanding these basic constraints, and if you have not done so already, we recommend that you find and watch the two-minute YouTube video entitled “Admiral Grace Hopper Explains the Nanosecond.”

8.5.2 Bandwidth

Bandwidth is the number of messages that can be sent per unit time. Note that this notion cannot be meaningful unless messages can only be split into a finite number of indivisible messages. Thus, the notion of bandwidth requires that messages are discrete entities. For uniformity, a message can be taken to be one of exactly two possible values, that is, one bit. Note also that bandwidth is based on the data being transmitted rather than the information it conveys, as the latter is always a function of the beliefs of the receiver.

www.dbooks.org

Considering our example above, if we assume that there are no verbal or visual hints given by the first person, the "message" can be seen as being one of two things: Either one ice-cream is handed over, or none are. If we further consider that this event can occur only once per day, then the maximum transfer rate is one message per day. Since there are only two possible events, let us consider the message to be one bit.

Assuming that the information the receiver takes from getting the ice-cream is that the sender likes them, this is a like/neutral signal. If the sender and/or the receiver would like more detailed information, such as really-like/like/neutral, then more bandwidth would be needed. This can be achieved, for example, through the use of two ice-creams. So that the information mentioned can be represented by two-ice-creams/one-ice-cream/no-ice-cream. Of course, such transmissions may cost more or require more work, but the amount of information that can be transmitted increases. As we will often see, physical resources can often limit the rate of transfer. For example, there is only so many standard sized ice-cream cones and scoops on the planet. But what is physically transmitted is only one source of limitation. In the following exercise, we consider some others.

Exercise 8.5 In the above example, using twice as many ice-creams did not double the levels of "like" that we have.

1. Are there ways in which a maximum of two ice-creams per day can be used to communicate four like levels, such as like-a-lot/like/like-a-bit/neutral?
2. What is the key idea that you are using to achieve this higher level of information transfer? In other words, is there a reason why this method can be expected to generalize to other situations?

8.5.3 Reliability

Under idealized conditions, for example, the universe consists only of you, your friend, and the ice-cream, the transmission of the ice-cream should be quite reliable: Once you start the process of handing over the ice-cream to your friend, the expected outcome for them should be that they receive it and recognize the message. But idealized conditions may be hard or even too difficult to provide. Instead, you and your friend could be standing outdoors on a windy day, they may be looking the other way as you get the ice-cream that you wish to give to them, and a wind might come and blow away the ice-cream before you are able to offer it to your friend. Alas, the physical evidence of the ice-cream is now gone. This kind of situation is a simplified example of the reliability issues that arise in almost all real-world communications. In general, they can also become more challenging as we try to transmit more information, over larger distances, in dynamic environments, and between mobile entities.

8.6 Fundamental Limits from Physics

Nature poses fundamental limits on link characteristics. These limits tend to become significant at extremes of transportation speeds or energy usage. For example, as we approach extremes of low energy, the smallest possible unit of energy transmission is one photon. If we also reduce our sampling period to a small enough period, to correctly detect a photon as it arrives would mean that we would know its speed and position, and that would lead up to other known limits posed by what is called the Heisenberg uncertainty principle.

Another fundamental physical limitations on bandwidth is to consider that the highest known frequencies for electromagnetic waves, gamma rays, are about 10^{25}. Even if we assume that we can pass one wave or skip it to encode a bit, this would be the maximum bandwidth. In practice, there are numerous reasons why even this assumption cannot be realized. But at least we have a relatively easy way to see that there are some hard limits on a single-bit communication channel.

While we can see these physics-imposed limits on latency and bandwidth as constraints on a space of possible solutions, they can also be seen as sources of inspiration for further research and innovation.

8.7 Limits Due to Component Dynamics

While nature can limit latency and bandwidth at a fundamental level, dynamics of the components used to build the communication link will generally pose significantly greater effects that will lead to the dominant practical concerns. To illustrate this concretely, we consider the common case of what happens when we use electric circuits to communicate.

8.7.1 Electrical Signal Transmission

The simplest example of an electric circuit where a signal can be transmitted is one where there is a constant current or voltage being transmitted from one entity to another. Let us focus on the case when we are transmitting a voltage (a similar analysis can be carried out if we transmit via current). As noted above, due to the laws of physics, our ability to observe any physical phenomena is always limited by some minimal quantity that can be measured. For this reason, we consider only discrete levels of voltage difference. The simplest case is to have two levels. We can after all use a series of such transmissions to represent any number of levels. For the sender to build up the voltage to be transmitted, a sufficient number of electrons must be moved from one side of the circuit to the other. Voltage difference is proportional to the amount of electrons moved. The rate at which electrons move is called current. Current generates heat and thus consumes energy, and so has to

www.dbooks.org

be limited in any circuits otherwise it will overheat. Limiting current means that building up the voltage takes time. This means that there is a minimum time needed to change from one voltage level to the other. This, in turn, limits the rate for data transfer (bandwidth) on this wire.

The situation described above can be modeled by a series RC circuit (shown in Figure 8.5) where a voltage source V_i is connected in series to a resistance R and then a capacitance C. We will call the current flowing the circuit I and the charge across the capacitor Q. Using the principles introduced in the chapter on physical modeling, this circuit's dynamics is governed by the following two equations:

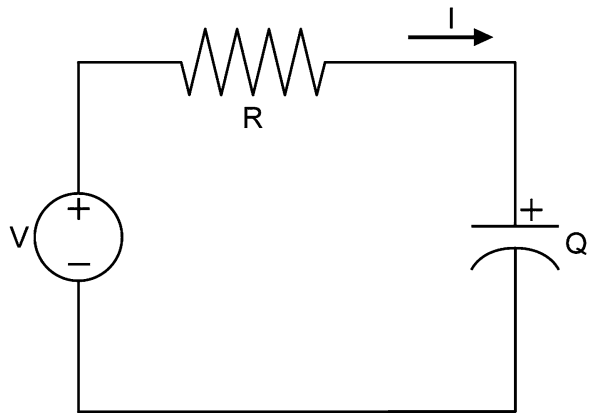


Fig. 8.5 A series RC circuit model of an electric signal transmission channel

$$V_i(t) = I(t)R + Q(t)/C, \tag{8.1}$$

and

$$Q(t) = \int_0^t I(s)\, ds. \tag{8.2}$$

The first equation reflects the fact that the input voltage must be equalized by the voltage from the rest of the circuit. The second equation models the way the voltage at the target of the signal (represented by the capacitor) is a function of the current being transmitted and the time lapsed. This equation captures the effect of the physical movement of electrons that is necessary to build up the voltage at the target, and that will make it possible for the target to measure a change in the circuit.

To make these two equations easier to recognize we will note that since Q is the integral of I (which is what the second equation states) then we also know that I is the derivative of Q. That means

$$Q'(t) = I(t). \tag{8.3}$$

With this observation we can rewrite the first equation as

$$V_i(t) = Q'(t)R + Q(t)/C. \tag{8.4}$$

Using basic arithmetic we can turn this equation into

$$Q'(t) = \frac{V_i(t) - Q(t)/C}{R}, \tag{8.5}$$

which is an ODE. In Acumen, this would be written as

```
Q' = (Vi - Q/C)/R.
```

To have a full simulation model one only needs an initially section that provides some example parameters such as

```
R=1, C=1, Q=0, Q'=1, Vi=1.
```

The constants are selected here only for illustration and not for resemblance to any concrete circuit parameters. The key take away from running this simulation is that it takes some time for the voltage at the target, Q/R, which is simply Q in this case, to reach the value of the source.

This simulation suggests several observations that can be confirmed through further mathematical analysis of the equations. For example, if we consider the start of the simulation, we can see that for any non-zero sensitivity to detecting voltage change there is a non-zero time needed to allow the voltage to grow to this level. But it is also important to note that this simple experiment does not tell the full picture. If V_i is to be changed to transmit both zeros and ones in sequence, then in general it may not be easy to detect voltage changes, but rather, we may want to have the voltage to reach certain specific values to consider this a reliable measurement. Such a requirement would further increase the time that we must allow for the signal at the target to reach a measurable level. The key take away here is that detecting signals requires time.

8.7.2 Variability in Component Parameters

Bandwidth and latency also suffer due to both capacitive and inductive effects of electrical wires. In addition, another important practical source of limitations is the variability in individual components. With any manufacturing technology, it is hard to create components that have identical characteristics, due to natural variations in the environment, materials, processes, and other factors. In poorly designed systems, the variability in individual components can be greatly magnified when we put them together. Techniques such as feedback, discretization, and quantization all provide important tools for managing this problem. For the purposes of this chapter it suffices to be aware of this issue and the need for these methods to address it.

www.dbooks.org

8.7.3 Light and Radio Transmission

In contrast to electrical signals, light and radio wave transmission can have an advantage in terms of maximum bandwidth and latency. To give a concrete example, whereas twisted pairs can have speeds of up to 10 GHz, fiber optics transmission can go to 200 GHz and beyond. Electromagnetics in free space can have frequencies in the THz, therefore, bandwidth can in principle also approach these frequencies. However, by definition they are in general not directed, and therefore can be subjected to large dissipation effects that can limit their range. In many cases, however, there are physical obstacles on a transmission path, which stop or significantly reduce the signal.

8.8 Limits Due to Noise

Noise is a term that is used to describe environmental factors that can make sensing difficult. The simplest example is when we are talking to a friend in a busy gathering and find it difficult to hear each other because others are talking in the background. In this case, the medium you are using to communicate is also being used by other messages in a way that makes it hard for you to receive your friends message correctly.

Noise arises in virtually all known modes of communication. Ambient vibration, sound, light, heat, and radiation are all phenomena that can be considered to be types of noise, and that can affect a communication channel. In transportation, a message can be influenced by the many other messages that go through the system, as well as the occasional failures in the process. Electrical signals can be influenced by cross talk due to effects of varying electric potentials and electromagnetic effects from the rest of the system and the surrounding environment. Especially at high altitudes and in outer space, they can also be influenced by background radiation. Noise can also arise from contention over shared resources, which can be seen, for example, in the effect of nearby channels in radio transmission.

The inevitable presence of noise has several significant effect on signals. A particularly significant issue is that for all practical purposes there is a minimal precision for measurements. This is an important justification for why we thinking of messages as having discrete values. In essence, this decision reflects the fact that below a certain level it is impossible to make any measurement reliably. Another effect is stochasticity: Noise can be unbounded and then all we can hope for is that it follows a probabilistic distribution. Depending on this distribution and the possible magnitudes of the noise, it may be that correct measurement cannot be guaranteed with absolute certainty. This gives rise to the need for probabilistic methods in communication. This is a highly significant concern since sufficiently large levels of noise (or, alternatively, sufficiently low signal levels) lead to a situation where the probability of a correct reception of a message is indistinguishable from a guess by flipping a coin.

The situation in the last case deserves some attention, as it can be undesirable to receive a random message believe that was the message intended by the sender. A wide range of methods are used to allow the receiver to check the integrity of the received message. At the very least it allows the receiver to ignore the message, but more commonly it makes it possible to request a retransmission.

8.9 Limits Due to Energy Dissipation

While electrical transmission and light transmission are guided (for example, by placement of the wire or optical fiber, respectively), electromagnetic transmission in free space is unguided. Guiding is significant for preserving the energy in a signal, thus facilitating its travel to greater distances. To illustrate, you may be familiar with the experiment of creating a mechanical telephone. Such a device can be created by piercing two metal cans or plastic cups and using them to stretch a plastic fishing wire at a distance. As long as the wire is stretched, it can guide the transmission of vibrations from one end to the other rather efficiently. This allows a person holding one end to hear what the other person holding the other ends says into their side of the device. This experiment offers a hint that a signal traveling along a wire (a one-dimensional space) can be preserved quite well. The only limitations to the transmission of such a signal are dissipation due to inelastic effects in the fishing wire, which would transform the vibrations into heat, or the leakage of the mechanical vibration into the surrounding environment. Ignoring these effects, this gives us a good starting point for thinking about what happens to a wave when it is not guided.

For example, let us imagine that we have a signal propagating in two dimensions, such as what we might see if we drop a marble into the middle of a still pool. For simplicity, let us once more ignore secondary effects and assume that the energy of the wave that starts right where the marble was dropped is preserved as the wave spreads out. We know from basic geometry that the radius of the circle that represents the center of the wave as it goes out is linearly proportional to its circumference. Given the symmetry of the situation, it is reasonable to assume that the energy will be spread equally around the circumference. This means that if we observe the energy at any point on the circle (the wave), the energy of the signal at this point will go down in inverse proportion with the distance from the center. Similarly, in a three-dimensional setting where the wave is propagating spherically, the energy will go down in proportion to the square of the inverse of the distance.

8.10 Other Sources of Limitations

Several other practical aspects can also lead to limitations on communication. One is the clock speed of the sending and receiving systems, which occasionally need to be shared with other components of the communication system. Clocking is a

www.dbooks.org

discretization technique that facilitates the design and operation of large digital circuits. However, clock speeds often have to fit with the timing need of the most complex components on the system. In simple designs this may need to be matched or aligned with the clock rate of the transmitting or receiving device.

8.11 Chapter Highlights

1. Communication as transfer of information
   (a) Information and certainty/uncertainty
   (b) Information and knowledge, belief, and truth
2. Widely applicable concepts relating to communication
   (a) Latency, the time to get a message through the channel
   (b) Bandwidth, the maximum rate of sending messages
   (c) Reliability, knowing that the message will come through when you send it
   (d) Possible connections between each of these different concepts
3. Fundamental limitations and their sources
   (a) Effects from physics (nature)
   (b) Effects from physical component dynamics
   (c) Effects due to noise (and/or "disturbance")
   (d) Effects due to energy limitations

8.12 Study Problems

The problems in this section can be investigated individually or in groups. They are larger than in the earlier chapters, so, expect them to require more time to solve.

1. Model a system for transmitting the bits of the binary representation for 42 on the RC channel presented in this chapter. Your model should include a vector representation of the binary representation for the message at the sender and the receiver.
   (a) Find the shortest clocking period that would allow the correct transmission of the signal for this initial test message.
   (b) Once this time has been determined, find another string that would not be transmitted correctly using these settings.
   (c) Explain why the evaluation using the first string was not sufficient.
   (d) Find a way to determine the fastest clocking period.

2. Amplitude modulation (AM) transmits a signal using a carrier that is itself a fixed frequency wave. It is the basis of AM radio. In essence, the signal and the carrier are multiplied to generate the transmitted signal.
   (a) Model a source and signal generation for such a communication, assuming that the transmission channel is perfect. Assume that the carrier frequency is 100 Hz.
   (b) Assume that the target knows the exact transmission frequency but not the phase. Model the target and explain the mechanism for determining the phase for the carrier signal.
   (c) Use the channel to transmit the 4 bit representation of the numbers from 0 to 8.
   (d) Determine experimentally the fastest rate with which this transmission can be done correctly using this channel.

8.13 To Probe Further

- Forbes article entitled The Real Reasons Quantum Entanglement Doesn't Allow Faster-Than-Light Communication.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

www.dbooks.org

Chapter 9
Sensing and Actuation

The terms sensing and actuation are used to refer to getting information about the world and to affecting physical objects, respectively. In cyber-physical systems, an interesting aspect of exploring sensing and actuation is that it provides us with a natural opportunity to learn more about how computational components work today, and in particular, which ones can be realized directly using semiconductor based circuits, and which ones require other intermediate steps to realize.

9.1 Everyday Input and Output

Sensing and actuation occur in simple forms in many computational systems such as a desktop, laptop, or any device that we traditionally think of as a digital computer. Electronically, the simplest way to get an input into a computer is through a switch, such as the home button on a smartphone or a particular key on a keyboard. A switch is a device that either allows or blocks a current depending on external input, such as the physical position of a lever or button. A switch can, in principle, be used directly to send a binary signal into a digital circuit. In principle, a keyboard button or an OFF/ON switch can use such simple circuitry. In practice, and for a variety of reasons, additional circuits may be useful to improve the quality of the signal and to ensure the safety of various subsystems. The point is, having a device that provides input to a computational system, in its simplest form, can be quite straightforward. Maybe more importantly, every signal that we sense will need to go through such a step to enter into the computational system.

Once this signal has reached the digital circuit, it can be processed in one of two ways. Either the signal will be held by a latch so that changes in its value can only be observed at a clock tick or it can be read by a signal that will use its value directly.

The question now is how do we turn a digital output into a physical action of some sort. As it turns out, one of the simplest ways to observe the output physically is to use one of the technologies that is most widely used today, namely, the Light

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9_9

Emitting Diode (LED). For many microprocessors, all what would be required would be to connect an LED followed (in series) with a small resistor to the wire carrying the digital signal that we wish to observe. Then, when there is a high (voltage) signal on the wire, the LED turns on, and it turns off when the signal is low. As an aside, it should be noted that some microprocessors use HIGH to represent 0, whereas others use it to represent 1.

9.2 Symmetry: LEDs and Photo-Voltaic Cells

It would have been elegant if we could somehow observe such an output by a change in the position of a button. That would give us a nice symmetry that seems natural when we are using actuation and sensing primarily for communication. Alas, whereas an OFF/ON switch may be the easiest way to get an input into a computational component, building the mechanism needed to get a switch to move is, relatively speaking, non-trivial. LEDs, on the other hand, do give us an example of some of the simplest ways to provide an input and to observe an output from a digital system: An LED itself is photo-sensitive, and can therefore be used for sensing light as well as emitting it. Isolated in a lit environment, an LED will have a voltage across two connectors. This voltage can then be amplified to detect the presence or absence of light.

The fact that LEDs can be used for both sensing and actuation makes them particularly interesting devices, as they are highly flexible as input, output, and communication devices. At some point, certain models of one of the predecessors of the smart phone, the Personal Digital Assistant (PDA) supported communication between such devices via infrared. Remote controls for many home appliances have for a long time used infrared light for one-direction communication. More recently, there has been growing discussion of light fidelity (LiFi, in analogy to WiFi) as a communication medium. Fiber optics are currently one of the highest bandwidth mechanisms for communicating between computer systems and over long distances. In addition, fiber optic buses are used in high performance computing systems to connect CPU cores. The ease with which light can be generated and processed by semiconductor devices makes it possible, in principle, that future CPUs may use light within chips. Media reports in 2019 included news that Intel is working on a chip with optical interconnects for Neural Network applications (See To Probe Further).

Another interesting aspect of LEDs is that the presence of light creates a voltage potential that can be used to harvest energy. This is in fact what photo-voltaic cells do—they can be viewed as a minor variation on the LED. A challenge with photo-voltaic cells is that they must be grouped and connected electrically with care, so as to support higher aggregate voltages as well as to enable buildup of higher currents. In addition, being significantly exposed to an open environment, they must be packaged in a way that allows them to operate effectively over a long period of time without

www.dbooks.org

degradation in efficiency. In light of what we have just learned, the attentive reader will understand that design of LED lighting is challenging for similar reasons.

9.2.1 Diodes

To understand why connecting computational and physical components is interesting, it helps to know some basics of how modern computational components are realized. Most of us have heard of Silicon, and that computer chips are made from it. Most of us have probably also heard of semiconductors, and are also wondering why something with such a curious name could be so important.

Conductors are materials where electricity can flow easily. Metals are a classic example of a conductor. Whether or not a material can conduct electricity depends on its atomic structure, and in particular, whether it facilitates or prevents the movement of electrons. At the atomic level, conductors are characterized by having an overlap between orbits called conduction bands and orbits called valence bands. Conduction bands are where electrons can move freely. *Insulators* are materials where electricity cannot flow. Plastics are a common example of an insulator. In terms of bands, these are insulators that have a "big" gap between the energy level of conduction and valence bands.

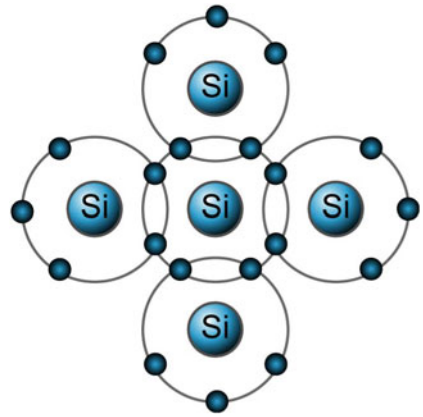


Fig. 9.1 A crystal of pure Silicon

Semiconductors are interesting not because they have a fixed conductivity somewhere between being a conductor or an insulator, but rather, because they can be used to build devices that act as either conductors or insulators depending on an external control signal. In the simplest case, this signal can be electrical. There are many ways in which such effects can be materialized. Diodes are arguably one of the simplest examples, because a diode can act as a conductor or an insulator depending on the direction of the electrical potential we apply across this device itself. Thus,

the control signal is the voltage across the device, and the effect we observe is how much current flows across the device as a result of this voltage. Unlike a resistor, the current that will flow through the device will vary dramatically depending on the direction of the voltage. To understand how this works let us take a closer look at how they are built in a semiconductor.

For simplicity, we will consider Silicon as a starting point. Silicon atoms have four electrons in their outer valence shell, and they form crystals by connecting with four other surrounding atoms (Figure 9.1). As a result, this creates a situation where in their outer valence shell each has eight electrons, which is a stable size for that shell. This keeps the electrons in place and makes Silicon an insulator at room temperature. Things become much more interesting when an impurity is added to Silicon, disrupting this stable form slightly and, in the process, giving it very attractive properties. This modification, which is made at the fabrication time, is called *doping*, and can be used to introduce either one free or one missing electron in the crystal lattice (Figure 9.2). Both types of doping, called n-type and p-type semiconductors, respectively, change the conductivity characteristics of the original crystal. But more importantly, when they are put next to each other, they create what is called a *junction*. This type of junction is the basis for creating a wide range of semiconductor devices, such as diodes, transistors, and photo-voltaic cells.

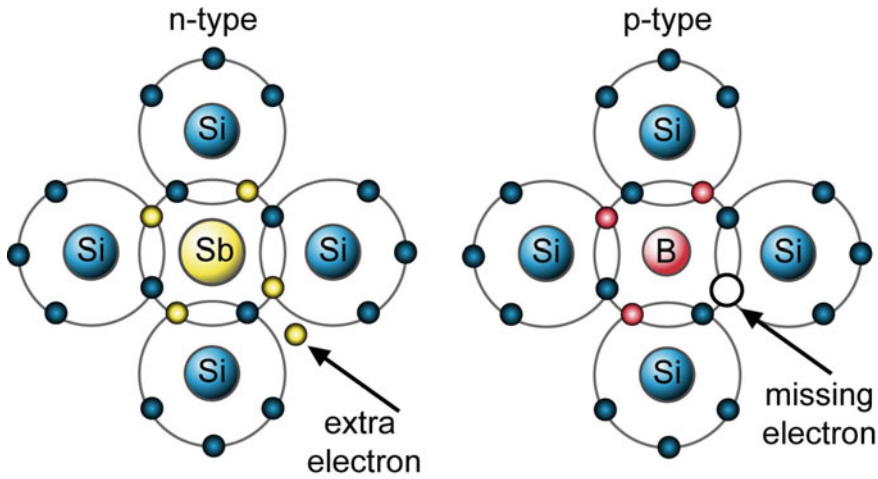


Fig. 9.2 How n- and p-type doping introduces free and missing electrons

One of the most interesting effects that arises at such a junction is the formation of what is called a thin *depletion region*, which results from the natural migration of the free electrons from one side to fill the hole created by the missing electrons on the other side. The result is that this junction lacks free electrons and is therefore not a conductor. This migration of electrons creates a voltage potential that any electron wanting to travel against needs to overcome. What is more, the size of this depletion region is sensitive to the voltage across the junction, and applied in one direction,

www.dbooks.org

this region will grow (and the voltage), but in the other, it will shrink (and the voltage buildup will be negligible). This effect is what gives junctions their ability to allow the flow of current in one direction and not the other, thereby giving us the diode as a device (Figure 9.3).

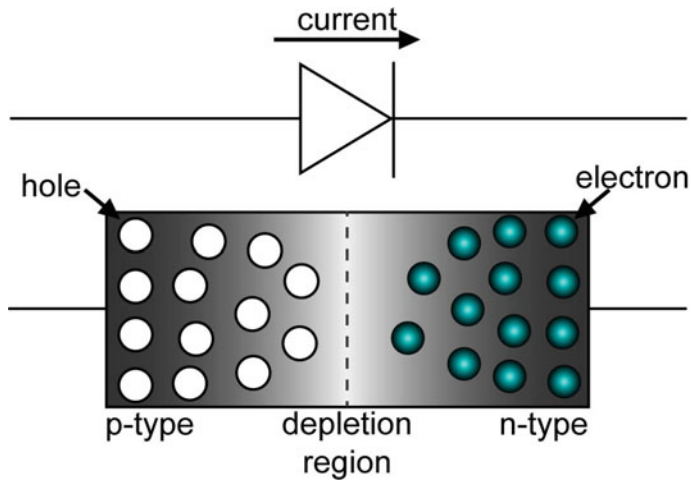


Fig. 9.3 An n/p junction creates a diode, the most elementary semiconductor device

Diodes by themselves can have a wide range of applications as electric circuit components, including in demodulation of radio signals and building digital logic circuits. For our purposes in this chapter, they help us get a basic appreciation of how semiconductor technologies work, and will help us understand why light is possibly the easiest non-electric physical media that we can connect to a digital circuit.

9.2.2 The Photo-Voltaic Effect

One of the most interesting aspects of what happens across depletion regions is the involvement of light. When an electron moves from the n-type side to the p-type, it is possible that an electron moves down from the conduction band to a valence band. This is sometimes called a recombination, as it is when an electron meets a "missing" electron in the valence band. The difference in energy between the conduction band and the valence band results in the emission of a photon (Figure 9.4). Depending on its energy, such a photon can form visible light.[1] For traditional circuit applications, such gaps are avoided for efficiency. For LEDs, the device is designed to maximize

[1] It should be noted this simpler account is more applicable for semiconductors such as Germanium. For Silicon, other physical effects play a more prominent role than photons.

the chance of the occurrence of such events, and to produce light at a particular frequency.

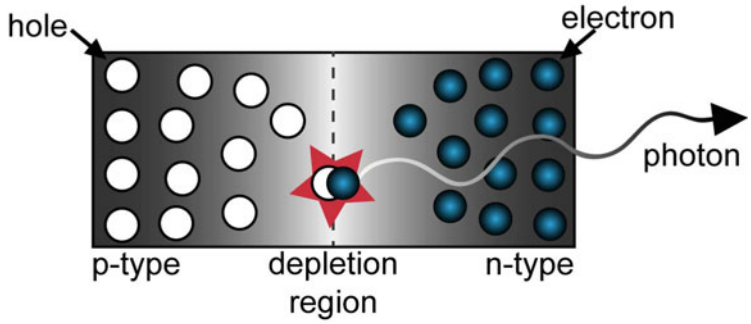


Fig. 9.4 Light Emitting Diode (LED)

An even more interesting effect is that there is also a dual dynamic: In the abundance of photons, such a junction can have electrons flowing in the opposite direction, increasing the voltage potential across the junction. The voltage potential can allow us to use this junction as a photo-voltaic cell that can be used to detect the presence of light. In the abundance of light and with appropriately configured circuitry, such cells can also be used to harvest electrical energy from this light. The basic dynamic at the level of atomic physics is called the photo-voltaic effect, and is closely related to the photo-electric effect, for which Einstein was awarded the Nobel Prize.

9.2.3 Transistors and Amplifiers

Junctions therefore allow us to build devices such as diodes, LEDs, and photo-voltaic cells. They also allow us to build another important semiconductor device, namely, the transistor. In its simplest form, a transistor can be made by juxtaposing three semiconductors segments with different doping, such as p-type followed by n-type followed by p-type. This configuration creates two junctions and a device with three terminals. Many useful effects can be realized using this device. For example, a small change in the voltage (or current) provided by the middle terminal can have a significant effect on the current that can flow across the two other terminals. This effect can be employed to realize circuits that can amplify the amplitude of a signal by several orders of magnitude. To perform this functional reliably, more than one transistor is used to build an *operational amplifier*, which functions as explained in the chapter on Control. In the context of generating and sensing light, operational amplifiers can be used to boost a digital off/on signal to drive a light emitting diode that delivers brighter light (and can therefore travel further) or amplify a low light

www.dbooks.org

signal coming in from a photo-voltaic cell to register clearly as a signal in a digital circuit (Figure 9.5). Amplifiers similarly play an important role in the accuracy with which we can sense external signals, and with which we can drive external devices. They are also used in both analog-to-digital and digital-to- analog converters.

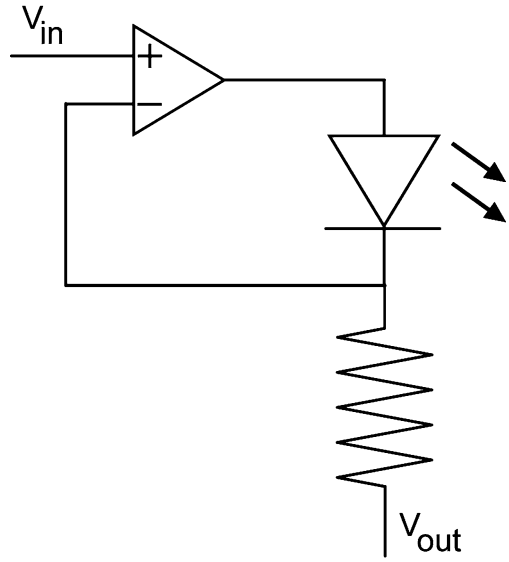


Fig. 9.5 Operational Amplifiers have numerous applications. In this circuit, one is used to drive an LED

9.3 Analog-to-Digital Conversion (ADC)

To transfer an analog signal into a digital computational component we need an analog-to-digital converter (ADC). To transfer a signal from a digital computational component we need a digital-to-analog converter (DAC). Both circuits are best understood as analog circuits and it is simplest to think of the digital value as being represented by the minimum voltage and the highest voltage (such as 0 and 15 V) and the analog signal as being able to have any value in between. For simplicity, we will also assume that we have four bits to represent the signal. This means that we can only represent 2^4 or 16 values.

A basic strategy for converting the analog signal to a digital one is to start with a simple ladder circuit made of a series of equal resistors that goes from the high voltage to the lowest voltage. In the case of our 16 level circuit, we would use 16 resistors. As long as the resistance on each is equal, the voltage drop across each of them will be equal. This will give us a source for 16 different voltages going from the lowest value of 0 V to the highest of 15 V. Starting from the 1 V point and going up we can start building 15 circuits by feeding this signal into the negative input to

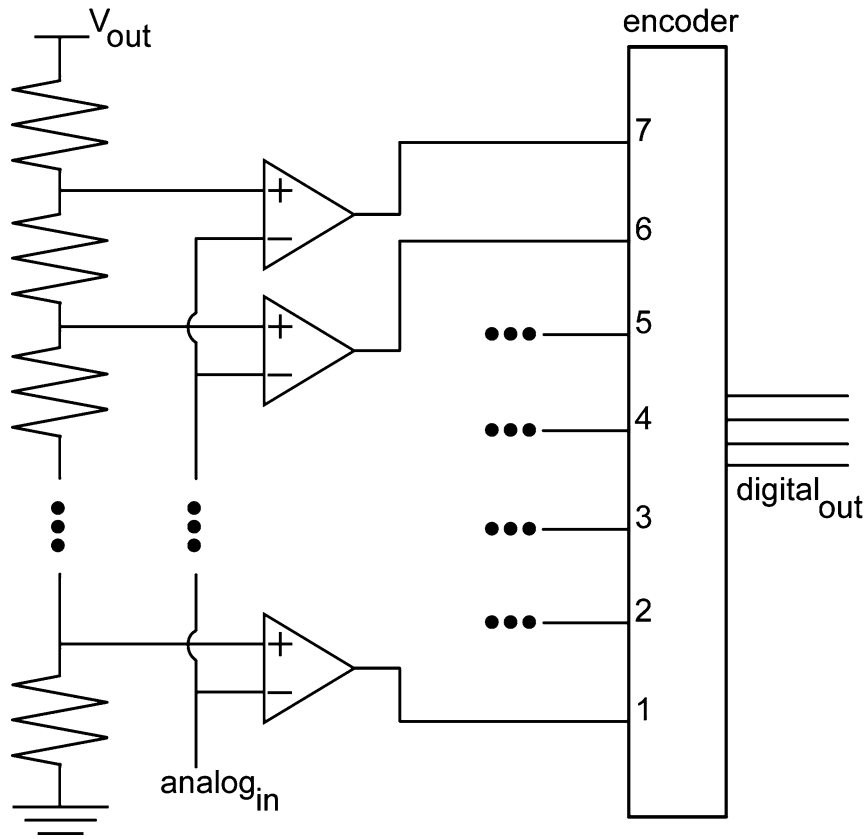


Fig. 9.6 A ladder circuit to convert from analog to digital

the operational amplifier, and the signal we want to measure to the positive one. This way, the output of each such amplifier will give us a high signal as soon as the input signal is higher that than this voltage, and will produce a very low signal otherwise. We can then treat these output signals as digital signals and collect them in one of several ways, including simply adding them or putting them through a simpler circuit called a priority encoder, which identifies the "highest" of the 15 lines and converts its number into a four-bit representation. Figure 9.6 depicts an example of such a circuit. The following model illustrates the behavior of a ladder circuit:

```
initially
Vs = 1:16,input = 0,input' = 1,output = 0
always
input' = 2,
output = sum 1 for v in Vs if input > v
```

The effect of this circuit is essentially the same as computing the `floor` of the input value, which is a more direct model of quantization. Rounding can be viewed as a

www.dbooks.org

model of basic analog-to-digital conversion. Depending on the application we can choose to build circuits that realize other rounding operations such as those that computing the ceiling or the closest integer value. Also, if we have more bits or if we have a smaller range of input values, we can let each integer represent a fraction. Again, the effect of such a circuit can be modeled more directly with a rounding function, but we would have to multiply the input first by the denominator of the fraction and then divide the resulting integer by that fraction to recover the value that we are representing.

9.4 Digital-to-Analog Conversion (DAC)

A basic strategy for the dual process, namely, digital-to-analog conversion (DAC), also makes use of an operational amplifier. In this case, a classic circuit called the *summing amplifier* is used, whereby an operational amplifier's positive end is connected to ground and the negative end is connected to an input node.The input

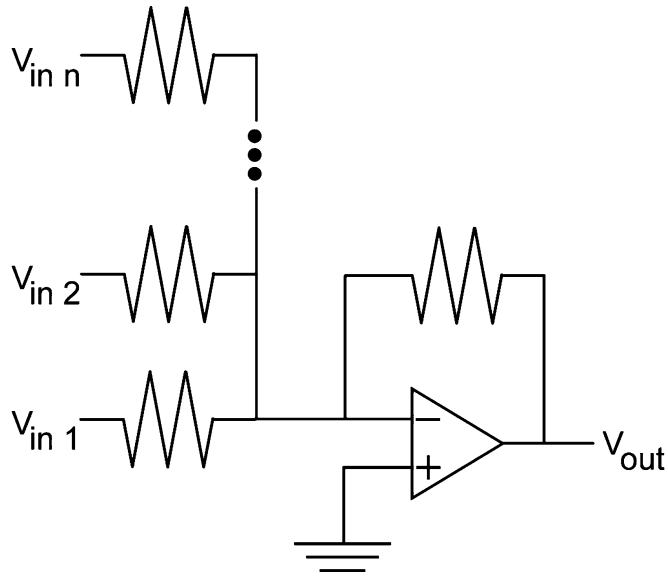


Fig. 9.7 A summing amplifier to convert from digital to analog

node is connected via a (denominator) resistance to the output of the amplifier to provide a feedback signal. In addition, the input node is connected to any number of resistors that are connected to the bits encoding analog signal we want to generate. This configuration provides a mechanism for making the output take the value of the sum of (one over) all the other resistances connected to the negative input of the amplifier for the bits set to a high voltage. Figure 9.7 depicts an example of such a circuit.

9.5 Sensing Temperature

Now that we understand the basics of how an analog signal can be mapped to a digital one, we can now turn to how the analog signals themselves can be generated. Of course, we have already considered light. One of the most commonly measured parameters is temperature. Applications include air-conditioning systems, almost every battery inside a smart device such as a smartphone or a computer, and various mechanical and chemical processes. Interestingly, it is now quite common to measure temperatures inside CPUs to avoid overheating and to respond by stabilizing temperatures by varying workload distribution.

Temperature can generate an electrical effect in a number of ways. Thermocouples are junctions of two different types of metal that produce a temperature-sensitive voltage due to what is called the thermoelectric effect (Figure 9.8). An alternative is a thermistor, which is a device that has a resistance that depends on temperature. Virtually all materials change conductivity depending on temperatures, and materials that have more variation are more suitable for this application. This contrasts to materials chosen for building traditional components, which are chosen to minimize variation with temperature.

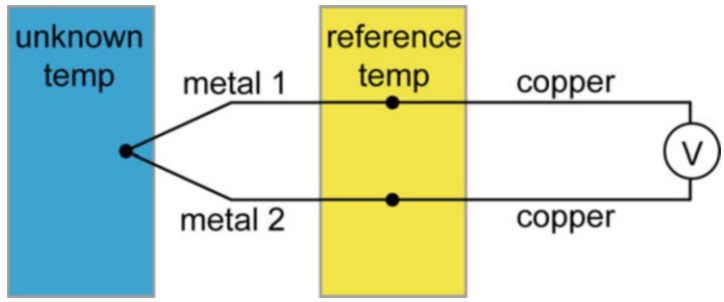


Fig. 9.8 Example of a circuit design for a thermocouple

9.6 Sensing Position

Another type of measurement that is commonly needed in a cyber-physical system is relative position. In the simplest case, a switch can be used to measure closed/open positions, as done in refrigerators and laptops. A more continuous measurement can be made using a device called a *rheostat*, which is a variable resistance device that changes resistance as one of the electrical terminals of the device moves along the resistive material, thereby changing the length that the current travels through the

www.dbooks.org

materials, and as a result, changing the total resistance (circuit element illustrated in Figure 9.9). This is a simple and reliable way to measure relative position, and can be used in both linear and angular configurations. However, it does require physical connectivity to the point which we wish to track. Internally, it also has moving parts that slide against each other, which over time can lead to significant wear and tear. For this reason, light (sometimes infrared) is used instead to detect affinity, and indirectly position. More commonly used in practice are rotary encoders that can measure either relative or absolute position using a variety of physical phenomena.

A wide range of techniques can be used for measuring position remotely, that is, without physical connectivity. Depending on the environment, one or more cameras can be used for providing positional information. In indoor environments, ultrasonic sensor can be used. In outdoor environments, systems such as the Global Positioning System (GPS) or LIDAR may be used depending on the demands of the situation.

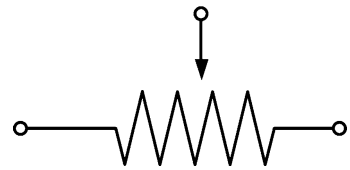


Fig. 9.9 Circuit notation for a rheostat, a basic device for sensing position

9.7 Actuating Mechanical Systems

When it comes to actuating mechanical systems, one of the most direct ways to achieve this is by powering an electric motor. Specialized operational amplifier designs may be used to generate the necessary electric power to drive a Direct Current (DC) motor. Most motors require a particular voltage level to be operated correctly. For this reason, typically, the main parameter that we control in actuating such a motor is how much power is delivered by rapidly turning the power ON and OFF according to a chosen ratio. For example, if we want to deliver no power the signal is OFF 100% of the time. If we want to send full power the signal is ON 100% of the time. If we want 50% power we mix OFF and ON signals in equal proportion. In essence, this provides us with a mechanism for controlling speed. Using feedback control and various mechanical gearing combinations, this approach can also be used to control position.

9.8 Chapter Highlights

1. Sensing and Actuation
   (a) Provide the link between computational and physical components in cyber-physical systems
   (b) Switches as one of the simplest input mechanisms
   (c) Missing symmetry
   (d) Why symmetry matters
2. Light as the medium closest to today's implementation technology for the cyber-part
   (a) Can serve as both input and output
   (b) Diodes, LEDs, and Photo-Voltaic cell
3. More on semiconductors
   (a) How semiconductors work
   (b) Transistors as the "next up" from diodes in terms of complexity
   (c) Transistors as the building block for operational amplifiers
   (d) Pervasive role of operational amplifiers in electronics
4. Building the interfaces
   (a) Temperature affects everything
   (b) Measuring relative positions
   (c) Actuating motors

9.9 Study Problems

1. Modify the example of Section 9.3 to use 5 bits and increments of 0.5 to represent the continuous input.
2. Simplify the result of the previous exercise by using the `floor` function.
3. Present a circuit for digital-to-analog conversion based on the strategy explained in Section 9.4 for the example discussed in the previous section. Derive the equation for the output as a function of the value of the input bits to show that this circuit will indeed function as a digital-to-analog converter.

9.10 To Probe Further

- Tech Lapse's article entitled Intel is working on optical chips for more efficient AI.

www.dbooks.org

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Appendix A
Acumen Reference Manual

This edition revises and updates the Acumen 2016/8/30 Manual.

A.1 Background

Acumen is an experimental modeling language and integrated development environment (IDE) for model-based design of cyber-physical systems. It is built around a small, textual hybrid-systems modeling language. This reference manual describes key features of Acumen when the "Traditional" option is selected from the "Semantics" pull-down menu. To report bugs with Acumen and/or issues with this manual, please use the online form available at http://www.acumen-language.org/p/report-bug.html. To continue to be updated about the development of Acumen, please subscribe to the announcements mailing list at http://bit.ly/Acumen-list.

A.2 The Acumen Environment and Graphical User Interface

The standard mode for using the Acumen environment is through the graphical user interface (GUI), which makes it possible to:

- Browse files in a given directory
- Load, edit, and save the text of a model
- Run models
- View a Plot, a Table, or a 3D visualization of variables over time
- Read error messages or textual outputs reported by the system

This book assumes that you are using Acumen through the GUI.

© The Author(s) 2021

W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9

www.dbooks.org

A.3 Basic Structure of An Acumen Model

A complete Acumen model consists of a series of model declarations. A complete model must contain a declaration for a model called `Main`. The declaration of the `Main` model must have exactly one parameter. By convention, that parameter is called `simulator`. For example, a typical model would have this form:

```
model Ball (mass, size) =

// Body of declaration of a model for a Ball

model Main (simulator) =

// Body of declaration of the model Main
```

The remainder of any line after the keyword `//` is ignored and treated as a comment. Similarly, any text that starts with `/*` and ends with `*/` is also a comment. Model declarations may appear in any order.

A.4 Model Parameters and the "Initially" and "Always" Sections

Model declarations start with a name for the model and a list of formal parameters, followed by the equals sign `=`. After the name and parameters, the model declaration can contain an `initially` section. An example is as follows:

```
model Ball (mass, size) =

initially
x_position = 0, y_position = 0

always
// Rest of the body of declaration of model Ball
```

The `initially` section defines the initial value for the variables local to this model. Parameter variables can be used in the definition of these initial values. Both parameter variables and model variables can be used in the rest of the body of the model. Variables introduced in this section cannot be referenced in the section itself.

The `always` section contains a collection of formulae, usually consisting of simple formulae and/or conditional formulae. It is **very important** to realize that all such formulae are executed at the same time. This also means that **the order of formulae in the text of the model does not matter**. It is still possible to model the continuous change of value denoted by a variable, let us call it `x`, through the use of **derivatives**, written `x'`. Similarly, it is possible to model the discrete change of a value through the use of the **next value**, written `x+`.

A.5 Model Instantiation

It is possible to model the creation of instances of a model. This can be done in either the `initially` or `always` sections. When done in the `initially` section, the created instance is called a *static instance, and when in the* `always` *section, it is called a* dynamic instance.

```
model Main (simulator) =

initially
b = create Ball (5, 14) // Static instance

always
// First part of model definition

create Ball (10, 42) // Dynamic instance

// Last part of model definition
```

New users will find it easier to work with static instances, since creating dynamic instances requires more care as they should be active at exactly one instant of time for each new object creation.

A.6 Expressions

Acumen expressions can be built out of variables, literals, built-in functions, vector generators, and summations.

A.6.1 Variable Names

In Acumen, a variable name is a sequence of one or more characters starting with a letter or an underscore, and thereafter possibly including digits. Examples of variable names include `a`, `A`, `red_robin`, and `marco42`. As a convention, variable names used by the language in a special way usually start with an underscore `_`. An example is the special variable `_3D`. A variable has a name followed by zero or more apostrophes `'`. Such apostrophes indicate that this variable is the *time derivative* of a variable with the apostrophe removed. Examples of such variables include `x'`, `x''`, and `x'''`.

www.dbooks.org

A.6.2 Literals

Acumen supports literal values of different types, including Booleans (`true` and `false`), integers (1, 2, 3, etc.), decimal values (`1.2`, `1.3`, etc.), floating point numbers (`1.2E-17`, `1.3E14`, etc.), strings (`"rabbit"`, `"ringo"`, etc.), and vector values (`(1,2,3)`, `(true, false, false)`, `("a", "ab", "abc")`, etc.). The special constants, `pi`, `children`, and the names of basic colors (such as `red`, `white`, and `blue`), are also literals.

A.6.3 Vector and Vector Generators

Vectors can be constructed by expressions like `(1,2,3)` and `(1,1+1,2+1)`. In addition, they can be generated by specifying a starting value, step size, and ending value. This is written as *start*`:`*step*`:`*end*. For example, `4:2:8` generates `(4,6,8)`. We can omit the *step* if it is `1`, and write *start*`:`*end*. For example, `4:8` generates `(4,5,6,7,8)`.

We can look up the first element in a vector `x` by writing `x(0)`, the second element by writing `x(1)`, and so on. The `length` function can be used to determine the length of a vector:

```
model Main(simulator) =

initially
list = (1,2,3,4,5), size = 0

always
size = length(list)
```

It is typical to use `length(list)` in a `foreach` formula.

A.6.4 Matrices

A matrix is represented as a vector of vectors. For example, the following is a two dimensional identity matrix: `((1,0),(0,1))`. The supported operations are the arithmetic operators `(+, -, *)`; the inverse `inv`, the transpose `trans`, and the determinant `det`. A sub-matrix can be extracted from an existing matrix using index *slicing*:

```
model Main(simulator) =

initially
I3 = ((1,0,0),(0,1,0),(0,0,1)),
```

```
I2 = ((0,0),(0,0))

always
I2 = I3(0:1,0:1) // First two rows and columns of I3
```

A.6.5 Summations

It is possible to iterate over collections to compute the summation of a series of values. The following example illustrates the syntax for this operation:

```
sum i*i for i in 1:10 if i%2 == 0
```

As this example illustrates, the `sum` construct allows us to indicate the iteration range and to filter the values being added based on a condition. The `if` clause can be omitted when there is not filtering, that is, when its condition is always true.

A.7 Formulae

There are five types of formulae in Acumen, namely: continuous formulae, conditional (or guarded) formulae, discrete formulae, iteration, and collections of formulae. We refer to continuous formulae and discrete formulae as simple formulae.

A.7.1 Continuous Formulae

A continuous formulae has a left-hand side that must be either a variable or the derivative of a variable, and a right-hand side that can be any expression. Examples include the following:

```
a = f/m

x'' = -9.8
```

Any such formulae in the same model are evaluated simultaneously. Thus:

```
x'' = -g, g = 9.8
```

is equivalent to:

```
x'' = -9.8
```

Continuous formulae are evaluated after all discrete formulae have been performed until they have stopped causing further change to the state of the model.

www.dbooks.org

A.7.2 If Formulae

The `if` formula is the first type of conditional formula. It allows us to express that formulae take effect under different conditions. The following code illustrates how an `if` formula is written:

```
if (x>0)
then x'' = -9.8

else x' = 0
```

In this example, as long as the value of the variable `x` is greater than zero then the first continuous formula is in effect. The result will be that the `x'` is decreasing. Since it is decreasing, whether it is already negative or starts off as positive, `x'` is guaranteed to eventually become negative. Similarly, `x` will also decrease until the condition is no longer true. Once that happens, the second equation will take effect, which will cause `x` to remain constant. By surrounding multiple comma-separated formulae in parentheses, they can be included in the else branch of an `if` formula:

```
if (x > 0)
then x'' = -9.8

else (x' = 0, stopped'=1)
```

A.7.3 Match Formulae

A `match` formula is the second type of conditional formula. It can be viewed as a generalization of an `if` formula that enables different formulae under multiple different cases depending on the value of a particular expression that we are matching on. The following example illustrates this idea:

```
match myCommand with
      [ "Fall" -> x'' = -9.8
      | "Freeze" -> x' = 0
      | "Reset" -> x = 0
]
```

Only one case can be enabled at any one point in time. Matching must be done against an explicit, constant value (like `"Fall"` and `"Freeze"`). If multiple clauses match the same value, only the first one will be enabled.

A.7.4 Discrete Formulae

A discrete formula has a left-hand side that must be the next value of either a variable or the derivative of a variable, and a right-hand side that can be any expression. Examples include the following formulae:

- `t+ = 0`
- `t'+ = 1`
- `t''+ = 0`

A discrete formula models an instantaneous change in the value of a variable. For a simulation to behave properly, any discrete formula in the body of a model definition (that is, outside the "initially" section) should generally occur inside a conditional formula (such as an `if` formula or a `match` formula) that will eventually stop being true. Otherwise, we can go into a type of infinite loop because there is an infinite chain of discrete changes in one time instance. The following example illustrates a typical use of discrete formulae:

```
if (x>=0) || (x'>0)
then x'' = -9.8

else
x'+ = -0.5 * x'
```

Here, the value of `x'` is reset to change direction (the negative sign) and reduce magnitude (multiplication by `0.5`) to model a "bounce" when a "ball" of height `x` hits the ground at level `0`. Note that as soon as the formula happens, the condition is falsified, so the discrete formula is enabled for exactly one time instant. Furthermore, because the condition requires that `x'` is negative, the new `x'` is guaranteed to be positive; therefore, we can also expect that the condition on the first line will become true, and the "ball" will again be subject to a downward acceleration (which can be seen as modeling the effect of gravity).

A.7.5 `Foreach` Formulae

A `foreach` formula allows us to perform iteration. Examples include:

```
foreach i in 1:10 do x = 2*y
```

and

```
foreach c in children do c.x + = 15
```

The second type of iteration illustrates how a model can assign a value to the `x` field of all its children.

www.dbooks.org

A.7.6 Collections of Formulae

Multiple simultaneous formulae can be expressed in a collection by simply placing a comma `,` between them. For example:

```
x'' = -9.8, y'' = 0
```

Order is irrelevant in such formulae, as they are always evaluated simultaneously.

A.8 How a Model Is Simulated: Order of Evaluation

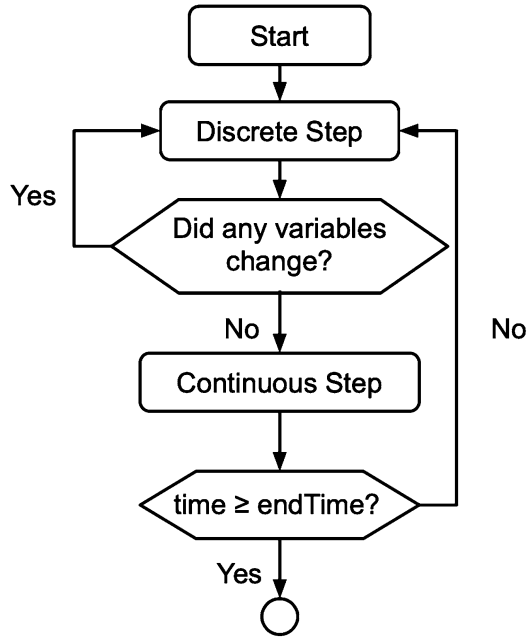


Fig. A.1 Top level evaluation loop

Initially, the simulation of an Acumen model has only one model instance, namely that of model `Main`. As model instances are created dynamically, a tree of instances is formed. The first instance of `Main` is always the root of this tree. The `children` of a model instance are, at least initially, the instances it creates. Every simulation sub-step involves a traversal of the entire tree starting from the root. Two kinds of sub-steps are performed, discrete and continuous (Figure A.1). During a *discrete step*, discrete formulae and structural actions (`create`, `terminate`, and `move`) are processed: the tree is traversed to perform active structural actions and collect active

discrete formulae. Once collected, the discrete formulae are performed in parallel. So, for example, `x+ = y, y+ = x` is a swap operation. For every model instance, first the structural actions of each parent are executed, and then the structural actions of all children are executed. If there are active actions that also change the state, we keep making discrete steps in this fashion. Otherwise, we make the continuous step. During the *continuous step*, all continuous formulae and integrations are performed.

A.9 Visualization Using the _3D Panel

Acumen has a _3D panel that can be used to produce static or dynamic visualizations in 3D. In the following we introduce the constructs needed to use this functionality.

Fig. A.2 Color panel

A.9.1 Colors

All 3D objects can have a color. Colors are described by a three-dimensional intensity representing the red-green-blue (RGB) dimensions of the colors. The color is represented by a vector of the form (r, g, b) where each of the values of r, g, and b is called an *intensity*, and is a real number be between 0 and 1. Figure A.2 illustrates some basic examples of RGB combinations. Here vectors indicate intensities, and not coordinates. That both are represented as a triple (that is, a vector of size three) is coincidental. To make it easier to put together _3D formulae, Acumen also defines

www.dbooks.org

constants for the intensities of the basic colors: `red`, `green`, `blue`, `white`, `black`, `yellow`, `cyan`, and `magenta`.

A.9.2 Transparency

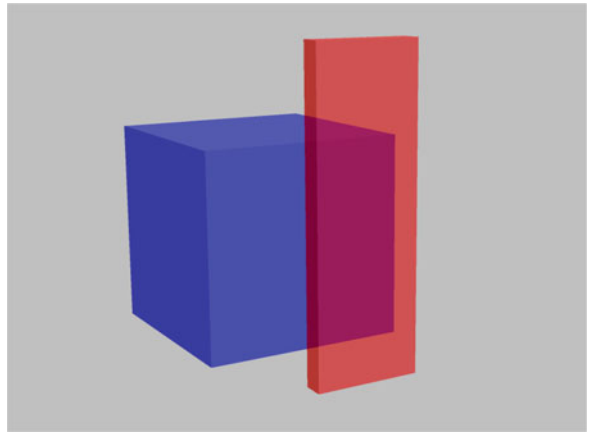


Fig. A.3 One transparent box occluding another

A 3D object can also have a degree of transparency. To support this Acumen provides a `transparency` parameter, which takes a floating point number ranging from 0 to 1. With value 0 representing opaque and 1 for the maximal transparency. The following model depicts a transparent box partially occluding another:

```
model Main(simulator) =
initially
_3D = (), _3DView = ()
always
_3D = ((Box center =(0 ,0 ,0)   size =(0.2 ,1 ,3)
           color=red  rotation=(0,0,0) transparency = 1),
       (Box center =(2 ,0,-0.5) size =(2 ,2 ,2)
           color=blue rotation=(0,0,0) transparency = 1)),

_3DView = ((-8,5,2), (0, 0, 0))
```

The 3D image resulting from this model is depicted in Figure A.3.

A.9.3 Coordinate System

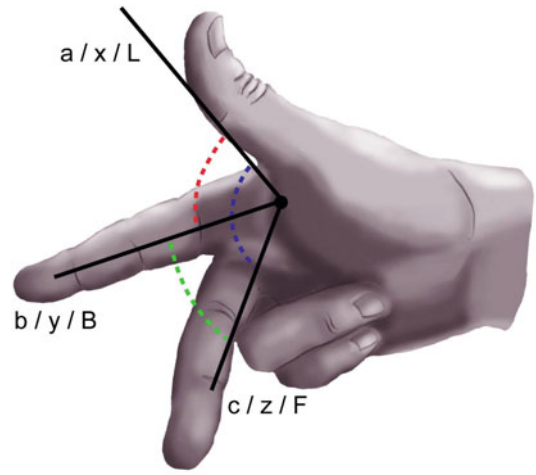


Fig. A.4 Right-hand rule

Acumen's `_3D` panel display uses a right-hand coordinate system, which is illustrated in Figure A.4.

Figure A.5 illustrates the coordinate system and some examples of points in that system. Each point is marked by a small cube, and next to it is text indicating the (x, y, z) coordinate of that point. Note that, unlike in the case of the color illustration above, the triples here are coordinates in three-dimensional space, and not color intensities. Rotations are specified as a triple of angles (in radians) about the center of the object, and are applied in the order described in Figure A.6.

A.9.4 Text

Text can be displayed in the `_3D` panel using a formula such as the following:

```
model Main(simulator) =
initially
_3D = (Text // Type of _3D object
         center=(-2.2,0,0)    // Starting point (x,y,z)
         size =0.75           // Font size
         color =(255 ,255 ,0) // Color in RGB
         rotation=(pi/2,0,0)  // Orientation (around x-axis)
         content =" Hello !") // Text to display
```

www.dbooks.org

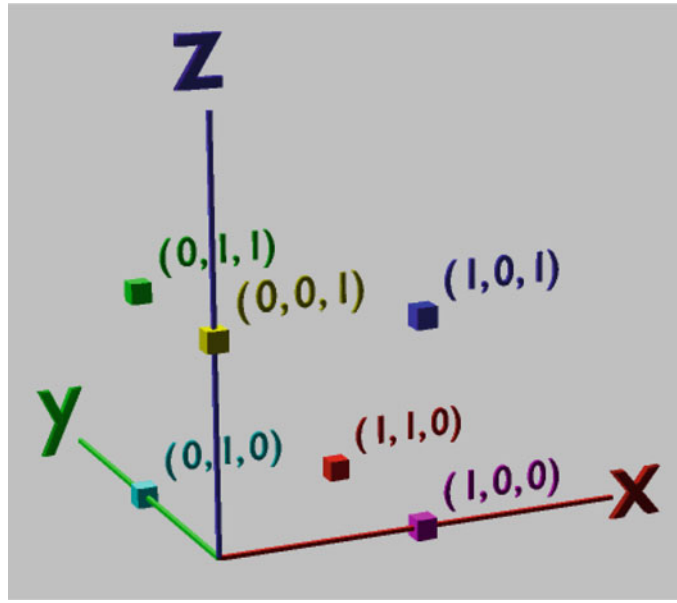


Fig. A.5 Coordinate system

The value assigned to `center` in the case of Text is actually where the text starts (the leftmost point of the displayed text) is not the true "center" of where the text is displayed. Orientations are angles that indicate how the text should be rotated around the x-, y-, and z-axes, respectively. Rotations are measured in radians, and specify an anti-clockwise rotation. Orientation rotations can be interpreted as rotations around the global frame of reference with the origin relocated to the reference point of the object that we are rotating; they can also be interpreted as having the rotation around the x-axis done first, then the y-axis, then the z-axis. Here are the characters supported by the Text primitive:

- 26 English characters, both uppercase and lowercase (a~z, A~Z)
- 10 digit characters (0~9)
- 28 symbol characters (! @ # $ % ^ & ( ) - + = | { } [ ] : ; " ' < > , . ? / *) and space " "

A.9.5 Box

A box can be displayed in the `_3D` panel using a formula such as follows:

```
_3D = (Box // Type of _3D object
           size = (0.2,1,3)) // Size in (x,y,z) form
```

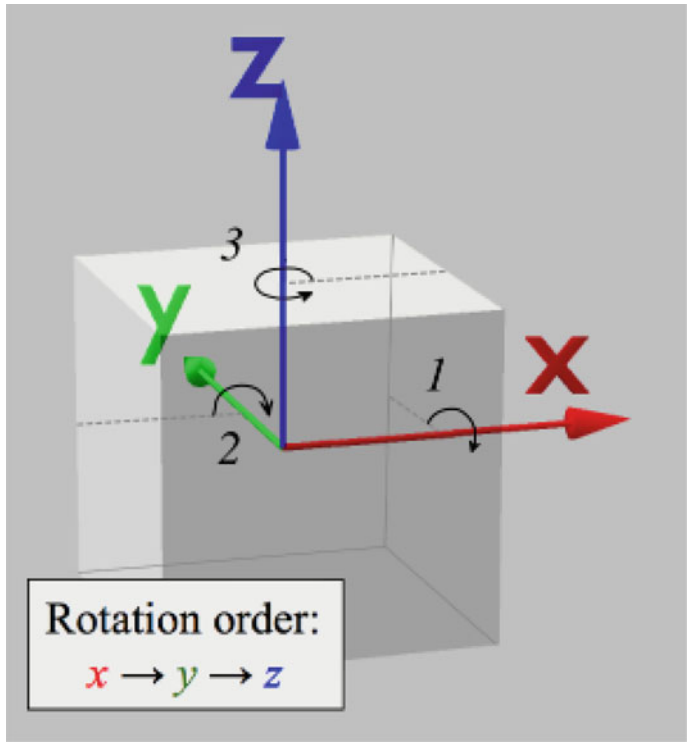


Fig. A.6 Rotation order

Note that, unlike text, boxes and the other geometric objects use the point indicated in the _3D formula to represent its center point rather than a corner point.

A.9.6 Cylinders

A cylinder can be displayed as follows:

```
_3D = (Cylinder // Type of _3D object
            radius = 0.1 // Radius
            length = 4)  // Length
```

Unlike a box, a cylinder only has two parameters to specify its dimensions, namely radius and length. The initial orientation is for its length to be along the y-axis.

www.dbooks.org

A.9.7 Cone

A cone can be drawn as follows:

```
_3D = (Cone // Type of _3D object
            radius = 0.4 // Radius
            length = 1)  // Length}
```

Note the similarity between the parameters for the cone and cylinder. The parameter types are the same, but they have a different meaning depending on the shape. The length is along the y-axis, and the pointy side is directed with the increase in the y-axis.

A.9.8 Spheres

A sphere can be drawn as follows:

```
_3D = (Sphere // Type of _3D object
            size = 0.55) // Size
```

Orientation on a sphere will not have a significant impact on the image.

A.9.9 OBJ Mesh Objects

Acumen supports loading 3D meshes saved as OBJ files as follows:

```
_3D = (Obj
color = cyan // Blended with texture of OBJ file
content = "Car.obj") // OBJ file name
```

A.9.10 Default Values

To make things easier, a default value is provided for any missing parameters for the various object types. Therefore, any of the following examples are acceptable ways to produce a sphere:

```
_3D = (Sphere // Type of _3D object
            center = (0,0,0) // Center point in (x,y,z) form
            color = cyan // Color
            rotation = (0,0,0)) // Orientation
```

or

```
_3D = (Sphere // Type of _3D object
            center = (0,0,0) // Center point in (x,y,z) form
            rotation = (0,0,0) ) // Orientation
```

or

```
_3D = (Sphere // Type of _3D object
           center = (0,0,0)) // Center point in (x,y,z) form
```

or even

```
_3D = (Sphere) // Type of _3D object
```

Similarly, defaults are provided for all other types of 3D objects.

A.9.11 Composites

All the display formulae illustrated above can be combined by adding a comma separator and inserting multiple formulae inside the outer parentheses. For example, the following formula illustrates the effect of the rotation parameter:

```
_3D =

(Text center=(-2,0,0) size=1 color=(1,0,0)
      rotation=(-pi/2,0,0) content="X",

 Text center=(-2,0,0) size=1 color=(0,1,0)
      rotation=(-pi/4,0,0) content="2",

 Text center=(-2,0,0) size=1 color=(0,0,1)
      rotation=(0+t,0,0) content="3",

 Text center=(0,0,0)
      size=1 color=(1,0,0)
      rotation=(0,0,0) content="Y",

 Text center=(0,0,0) size=1 color=(0,1,0)
      rotation=(0,pi/4,0) content="2",

 Text center=(0,0,0) size=1 color=(0,0,1)
      rotation=(0,pi/2+t,0) content="3",

 Text center=(2,0,0) size=1 color=(1,0,0)
      rotation=(0,pi/2,0) content="Z",
```

www.dbooks.org

```
 Text center=(2,0,0) size=1 color=(0,1,0)
      rotation=(0,pi/2,pi/4) content="2",

 Text center=(2,0,0) size=1 color=(0,0,1)
      rotation=(0,pi/2,pi/2+t) content="3"
)
```

Naturally, while this example contains only Text objects, composites can contain multiple different object types.

A.9.12 Shapes, Their Parameters, and Their Default Values

The following table lists the _3D shapes recognized by Acumen, together with supported parameters and the corresponding default values.

Shape	Center	Rotation	Color	Coordinate	Transparency	Content
Box	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	None
Cone	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	None
Cylinder	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	None
Sphere	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	None
Triangle	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	None
OBJ	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	String
Text	(0,0,0)	(0,0,0)	(0,0,0)	"global"	0	String

To specify the size of a 3D object, the following table lists the default values supported for different shapes:

Shape	Size parameter and its default value
Box	Size =(0.4,0.4,0.4) or length = 0.4 width = 0.4 height = 0.4
Cone	Size =(0.4,0.2) or length = 0.4 radius = 0.2
Cylinder	Size =(0.4,0.2) or length = 0.4 radius = 0.2
Triangle	Points = ((0,0,0),(1,0,0),(0,1,0)) height = 0.4
OBJ	Size = 0.2
Sphere	Size = 0.2 or radius = 0.2
Text	Size = 0.2

A.9.13 Animation = Dynamic _3D Values

In the examples above we simply assigned an initial value to the `_3D` field in the `initially` section of a model. In fact, it is also possible to continuously assign a changing value to the `_3D` parameter by assigning it a value in the "always" section. When this is done, the `_3D` panel animates the progress of this three-dimensional scene, as observed through the simulation time.

A.9.14 Manual Control of the View of the _3D Scene

To rotate the view around the center of the current `_3D` view, click and hold the left mouse button and move the mouse. To change the center of the `_3D` view, click and hold the right mouse button (on Mac OS, tap touch-pad with two fingers) and move the mouse. The mouse wheel can be used to zoom the view in and out (on Mac OS you swipe up and down with two fingers).

A.9.15 In-model Control of the View of the _3D Scene

The camera that defines the `_3D` view can also be manipulated directly in the model. This is done by adding the special variable `_3DView` to the model. Note that, as with the `_3D` variable, to use the `_3DView` variable in the `always` section, it is necessary to first declare it in the `initially` section. In the following model, the camera is situated at `(10,10,10)` and is rotated by `0.5` radians along each axis, so that it looks at the origin:

```
initially
_3D = (), _3DView = ()

always
_3D = (Box center=(0,0,0) size=(1,1,1)
color=red rotation=(0,0,0)),

_3DView = ((10,10,10), (0.5,0.5,0.5))
```

This configuration of the `_3DView` variable will yield the the `_3D` scene illustrated in Figure A.7. Animations are creating simply by using arbitrary expressions instead of literals in place of constants that set the various parameters of the 3D objects.

www.dbooks.org

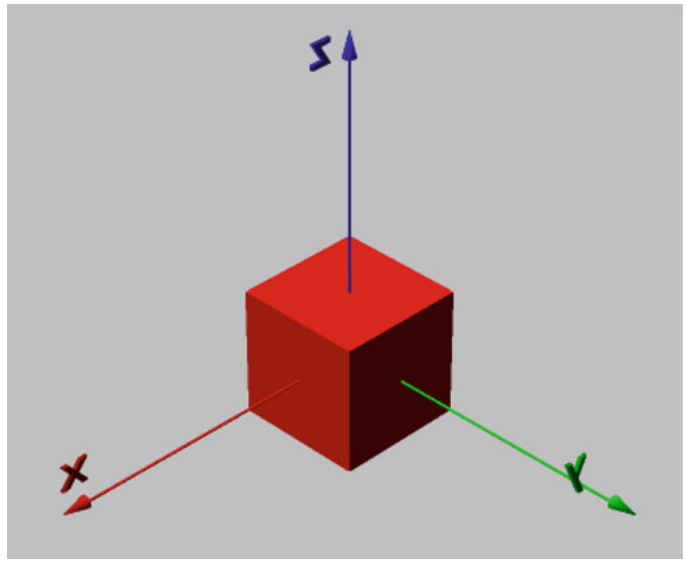


Fig. A.7 Scene with modified view point

A.9.16 Camera View

To make an object appear static with respect to the viewer, that is, to prevent it from being affected by the position of camera or manual view rotation or zoom in/out, the user can set the parameter `coordinates` to `"camera."` For example:

```
_3D = (Text
center=(-2.2,0,0)
color=(1,1,0)
coordinates = "camera"
content="Hello Acumen!")
```

is a model that results in the visualization shown in Figure A.8. Note that although the view has been manually rotated, as indicated by the axes, the text is still facing the screen. This concept is a bit hard to visualize in a picture, so we suggest that the reader runs the model above and modifies the view to see what this concept achieves.

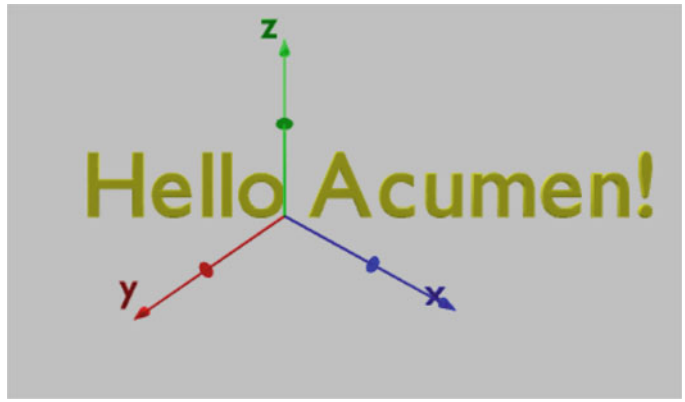


Fig. A.8 Static object view

A.10 Built-In Functions

Acumen provides the following built-in functions:

- Unary operators on Booleans and Integers: `not`, `abs`, -
- Binary operators on Integers: +, -, *, <<, >>, &, |, %, `xor`
- Unary operators on Floats: `sin`, `cos`, `tan`, `acos`, `asin`, `atan`, `toRadians`, `toDegrees`, `exp`, `log`, `log10`, `sqrt`, `cbrt`, `ceil`, `floor`, `rint`, `round`, `sinh`, `cosh`, `tanh`, `signum`, `abs`, -
- Binary operators on Floats: +, -, *, ^, /, `atan2`
- Relational operators on Integers and Floats: <, >, <=, >=
- Binary operators on Vectors: .*, ./, .^, +, -, `dot`, `cross`
- Unary operators on Vectors: `norm`, `length`
- Binary operators taking a Scalar and a Vector: +, *
- Binary operators taking a Vector and a Scalar: +, *, /, .^

In most cases, operators that start with a letter are prefix operators that take explicit arguments, such as the case with `sin(x)`, while operators that start with a symbol are infix operators, such as `x+y`. The only exceptions to this rule are `xor`, which is an infix operator, and unary -, which is a prefix operator that has no parentheses.

A.11 Function Declarations

To define custom functions beyond the built-in ones, Acumen provides a top level `function` definition construct. The structure of a typical function declaration is as follows:

```
function f(x,y) = x + 2*y
```

Inlining, which substitutes the call to a function in the code with a copy of the function body using the actually input arguments, is performed for every function call before running the simulation. For example, the function call `f(1,2)` is replaced with the expression `1 + 2 * 2`.

www.dbooks.org

A.12 Operator Precedence

The precedence ordering for built-in functions in Acumen is as follows:

1. Boolean disjunction ||
2. Boolean conjunction &&
3. Boolean equality == and then Boolean inequality ~=
4. Integer less than <, greater than >, less than or equal to <=, and then greater than or equal to >=
5. Vector generation $i:j:k$
6. Numeric addition +, subtraction -, vector addition .+, vector subtraction .-
7. Numeric multiplication *, division /, vector multiplication .*, vector division ./, and integer modulus %
8. Numeric exponentiation ^ and then vector exponentiation .^
9. Numeric unary negation -
10. Field access .
11. Built-in prefix function applications and vector lookup
12. Field check ?

A.13 Simulator Settings

The `simulator` parameter passed to the declaration of model `Main` provides the user with a mechanism to specify how the model should be simulated as part of the model itself. There are two basic settings that the user can specify:

- The time when the simulation should terminate is `endTime`. The default value for this setting is 10 s.
- The numerical integration step size is `timeStep`. The default value for this setting is $2^{-6} = 0.015625$.

It is generally recommended that any adjustments to these values are made using a discrete formula at the very start the simulation time.

A.14 Command Line Parameters

For batch processing, it is often useful to pass parameters to a simulation from the command line. For this purpose, Acumen provides a mechanism for passing such information. It consists of two parts:

- From the command line, the directive `--parameters` can be used to specify the name and value of parameters (each name followed by its value) and
- In the model, the creating model instance of type `simulator.parameters ()` provides access to these variables.

The following example illustrates how the parameters can be passed from the command line, and how they can be used within the model. To start a simulation in offline mode from the command line with certain parameter values, write:

```
java -jar acumen.jar --parameters abc 11
```

To use these values, the model itself can use the `simulator.parameters` type as follows:

```
model myModel (a,b) =
initially
x = a, x' = b, x'' = 0

always
x'' = -x

model Main (simulator) =
initially
p = create simulator.parameters(),
// Get command line parameters and put in an model instance

init_phase = true

always
if init_phase

then if (p?a && p?b)
// Check to see if parameters were provided in the
// command line

then create myModel (p.a,p.b)
// Use command line parameters a and b

else create myModel (17, 42),
// Default values, useful for interactive mode

init_phase+ = false

noelse
```

It is good practice to use the ? test on parameter names (as illustrated by the example above) to make sure that they have been provided by the command line call, and then to use default values in the case that they are not there. A side benefit of doing so is that models can be easily run in both command line and interactive modes.

www.dbooks.org

A.15 Print to Standard Output (stdout) or Console

For batch processing and also for the interactive mode, Acumen provides support for printing textual outputs. In batch mode, printing sends output to `stdout`. In interactive mode, it goes to the console. To see how the `print` annotation works, consider the following model:

```
model Main (simulator) =

initially
x = 5
// print out the value x-1 and x
always
if x > 0 then

x+ = print("x-1=", print("x=",x)-1)

noelse
```

Running this model results in the value of `x` being repeatedly decremented by one until it reaches zero. If we wish to observe this process on the console (or `stdout`), then we would insert a `print` annotation around the value being assigned to `x` in the `if` formula. The annotated version is as follows:

```
model Main (simulator) =

initially
x = 5

always
if x>0
then x+ = print(x-1)

noelse
```

Note that on the console the values are printed backwards (as are all messages).

A.16 BNF of Acumen

The following context free grammar defines the current syntax of the Acumen language. The notations ? and * are used, respectively, for optional and repetition, and text after // is a comment.

Set		Possible values
digit	::=	[0-9]
letter	::=	[A-Za-z]
symbol	::=	! | @ | # | $ | % | ^ | & | (|) | - | + | =| | {| } | [|] | : | ; "| ' | < | >| , | . | ? | / | * |
id_character	::=	letter | "_"
ident	::=	id_character (int | id_character)*
int	::=	"-" ? digit+
float	::=	"-" ? digit+ "." digit+
boolean	::=	`true` | `false`
string	::=	(symbol | id_character)*
gvalue	::=	nlit | string | boolean | **[** nlit **..** nlit **]** // interval
nlit	::=	int | float
name	::=	ident | name'

expr	::=	| gvalue | name
		| `type` (modelName)
		| f(expr*) // function application
		| expr op expr // binary operation
		| expr'[expr] // partial derivative
		| (expr)' | (expr)" . . . // time derivative
		| **(**expr*) **//** expression vector
		| name(expr) // vector indexing
		| **(**expr**)**
		| expr . name // field access
		| expr ? name // field check
		| `let` name = expr * `in` expr // let notation
		| `sum` expr `in` name = expr `if` expr // summation

op	::=	| + | - | * | / | % | < | > | <= | >= | && | || | | == | ~= | .* | ./ | // point-wise operator
action	::=	| name+ **=** expr // discrete formula | name = expr // continuous formula | `create` modelName (name*) | name`:=` `create` modelName (name*) | `terminate` expr // eliminate model instance | `move` expr expr // move model instance to new parent | `if` expr `then` action* `else` action* // conditionals | `foreach` name `in` expr `do` action* // foreach | `match` expr `with` [clause | clause . . .] | `claim` expr // claim predicate expr is true | `hypothesis` expr | `hypothesis` string expr
clause	::=	gvalue `claim` expr `->` action* | `case` gvalue `->` action*
modelDef	::=	`model` modelName (name*) inits action*
inits	::=	`initially` (name := expr | create modelName (expr*))* `always`

www.dbooks.org

Set		Possible values
model	::=	modelDef *
include	::=	`#include` string
interpreter	::=	“reference2015” | “optimized2015” | “reference2014” | “optimized2014” | “reference2013” | “optimized2013” | “reference2012” | “optimized2012” | “enclosure2015”
semantics	::=	`#semantics` interpreter
function	:=	`function` ident (ident +) = expr
fullMod	::=	semantics? include? function* modelDef*

Index

© The Author(s) 2021
W. M. Taha et al., *Cyber-Physical Systems: A Model-Based Approach*,
https://doi.org/10.1007/978-3-030-36071-9

www.dbooks.org

www.dbooks.org

www.dbooks.org