ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inspire
iwf
Institut für
Werkzeugmaschinen & Fertigung

Automatic extraction of POMDPs from Simulink models

Oliver Stollmann

Bachelor Thesis
2011

Bastian Migge
PD Dr. habil. Andreas Kunz

Prof. Dr. Konrad Wegener

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

Your task description pdf file here!

Acknowledgment

This is the acknowledgment

Contents

List of Figures

List of Tables

Introduction

This is the introduction

Motivation

This is the "Motivation" chapter!

Background

This is the "Background" chapter!

3.1 Dynamic Systems

This section covers deterministic and randomized dynamic systems in theory and gives a few intuitive examples of dynamic systems.

Dynamic systems are systems whose development over time can be described by a single or a set of mathematical equations. Although these equations may not always be symbolically solvable they describe the system's dynamics perfectly. Examples include the time-varying temperature of an object as it is placed in the oven or the voltage across an inductor.

A common example of dynamic systems is the mathematical pendulum see in Figure 3.1. By considering the forces on the pendulum or it's energy it is quite simple to deduce the following differential equation to describe the system:

$$0 = \frac{g}{l} \cdot sin(\phi(t)) + \ddot{\phi}(t)$$

With the small-angles assumption

$$sin(\phi(t)) \approx \phi(t)$$

the solution is of the differential equation is

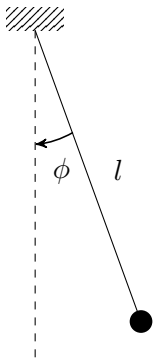

Figure 3.1: *1-Dimensional Mathematical Pendulum*

$$\phi(t) = \hat{\phi} \cdot sin(\sqrt{\frac{g}{l}} \cdot t + \phi_0)$$

where $\hat{\phi}$ is the semi-amplitude and ϕ_0 the phase at time $t = 0$. This equation describes the dynamic system perfectly for all times.

An interesting property of the pendulum system is that given the same initial state and excitation (eg. initial angle at $20°$ and speed 0), the system will behave the same way at as time progresses. This property is called determinism and guarantees that given the same excitation and initial state the system will always develop identically over time. Systems that posess this property are called *deterministic dynamic systems* and differ greatly from the opposed *randomized dynamic systems*.

Randomized dynamic systems are dynamic systems with an element of *randomness*. The consequence of this is that the same initial conditions and excitation do not guarantee an identicial system response. A trivial example of a discrete randomized dynamic system is the following:

$$\begin{aligned} q[n+1] &= q[n] + x[n] + e[n] \\ y[n] &= q[n] + x[n] \end{aligned}$$

where $x[n]$ is the input, $e[n]$ is white noise and $y[n]$ the output. If this system is provided with the same input signal twice, the resulting output signal is likely to be slightly different. This is caused by the inherint randomness of a white noise input.

Deterministic dynamic systems and *randomized dynamic systems* are two extremely useful mathematical constructs and serve as the basis of the more advanced theory of the next few sections.

3.2 Deterministic Processes

Deterministic Processes

3.3 Stochastic Processes

Stochastic processes describe systems that change probabilistically. Unlike deterministic processes, where the same initial conditions and excitation will always lead to the same result, stochastic processes develop randomly. This means that two dynamic systems that start at the same state and are given the exact same inputs may differ in their output.

Stochastic processes are useful in two cases. Firstly to model systems that are physically random (quantum physics) and secondly to model abstractions of reality. The latter case occurs when we choose to model a system probabilistically because we either don't understand the underlying dynamics or because we choose to model them probabilistically instead of deterministically.

The next three sections build upon each other, starting with Markov Chains, then extending these to Markov Decision Processes and finally to Partially Observable Markov Chains.

3.3.1 Markov Chains

A Markov Chain is a certain type of stochastic process that describes the dynamics of a probabilistic system by defining the conditional transition probabilities $Pr(X_{n+1} = x|X_n = x_n)$ between members of a finite or infinite state-space.

3.3.1.1 Markov Property

Markov Chains possess the Markov Property,

$$Pr(X_{n+1} = x|X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = Pr(X_{n+1} = x|X_n = x_n),$$

which states that the current conditional transition probabilities are independent of the systems past states.

3.3.1.2 Transition Probability Matrix for Finite-State Markov Chains

If a Markov Chain's state-space is finite the transition probabilities between states can be represented in a transition probability matrix where the probability of going from state i to state j, $Pr(X_{n+1} = j|X_n = i)$, is equal to the (i, j) element of the matrix:

$$Pr = \begin{pmatrix} Pr(X_{n+1} = 1|X_n = 1) & Pr(X_{n+1} = 2|X_n = 1) & \cdots & Pr(X_{n+1} = N|X_n = 1) \\ Pr(X_{n+1} = 1|X_n = 2) & Pr(X_{n+1} = 2|X_n = 2) & \cdots & Pr(X_{n+1} = N|X_n = 2) \\ \vdots & \vdots & \ddots & \vdots \\ Pr(X_{n+1} = 1|X_n = N) & Pr(X_{n+1} = 2|X_n = N) & \cdots & Pr(X_{n+1} = N|X_n = N) \end{pmatrix}$$

3.3.1.3 Markov Chain Example

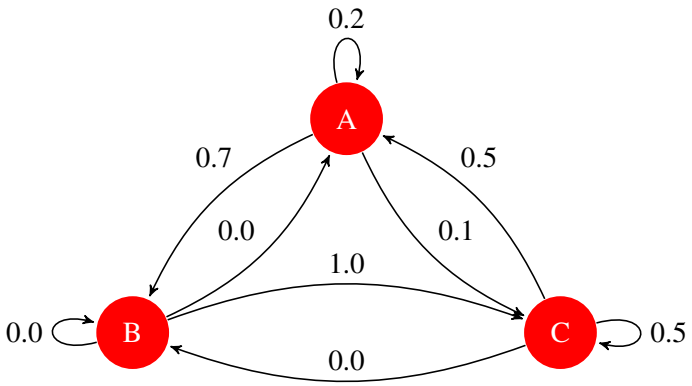

Figure 3.2: *Three-State Markov Chain*

Figure 3.2 shows an example of a three state Markov Chain. The transition probabilities matrix is as follows:

$$\begin{pmatrix} 0.2 & 0.7 & 0.1 \\ 0 & 0 & 1 \\ 0.5 & 0 & 0.5 \end{pmatrix}$$

Looking at this transition probability matrix or Figure 3.2, we can, for example, deduce that if the system is currently in state A the probability of transitioning to state B in the next step is 0.7, the probability of transitioning to state C is 0.1 and the probability of remaining in state A is 0.2. Interestingly, because the second row only contains a single non-zero value, which must thus be equal to 1, the transition from state B is entirely deterministic, meaning that it will always be the same.

3.3.2 Markov Decision Processes

Markov Decision Processes (MDPs) are an extension of Markov Chains and describe *controllable* probabilistic dynamic systems. They are defined as a four touple (S, A, P, R) with:

- S: set of states,
- A: set of actions,
- P: conditional transition probabilities,
- R: rewards.

3.3.2.1 Actions: Decisions

MDPs are used to model probabilistic systems that can be influenced through decision-taking. These decisions are represented as actions and have a direct influence on the transition probabilities of the system. Transition probabilities are now three-dimensional and depend not only on the current state, but also on the action being taken; $Pr(X_{n+1} = x_n | X_n = x, a_n = a)$ is the probability of going to state x_n in the next step given that the system is currently in state x and action a has been chosen.

3.3.2.2 Reward Model

MDPs also add the idea of rewards to Markov Chains. Although rewards are not necessary to describe *controllable* dynamic systems, they are the key to optimization. Every transition probability is paired with a reward, or cost (negative reward), value; $R(s, s', a)$ is the reward (scalar) that the system receives when it transitions from state s' to state s given that action a was chosen.

Note: It is important to understand that rewards do not have a probabilistic nature, but rather that the combination of conditional transition probabilities and rewards provide the basis for stochastic optimization in MDPs (see next section).

3.3.2.3 Optimization

MDPs can be used to optimize decision making. The combination of a system description and an associated reward model allow us to calculate the optimal decision to take in a given situation. The aim of this optimization is to produce a *policy*, $\pi(s)$, that defines exactly which action should be taken if the system finds itself in a certain state.

The computation of this policy requires us to define what we are trying to optimize. In most cases optimization simply aims for the maximization of rewards (or minimization of costs) over a certain decision span. This optimization goal is defined in a so called reward function, the most common of which is the *expected discounted total reward* (infinite horizon),

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}),$$

with:

- γ: discount factor, where $\gamma \in (0, 1]$,
- $R_{a_t}(s_t, s_{t+1})$: reward received in $t + 1$ for taking action a from state s_t at time t.

The *expected discounted total reward* seen above is only one of many possible reward functions. It represents the idea that we value the total sum of rewards, but prefer rewards received earlier to rewards received later; *rewards are discounted over time*.

Now that we have both, the set of rewards received and the reward function - definition of how we *value* these rewards - we can compute an optimal policy for the decision maker. The computation of this policy can be done with either *linear programming* or, more commonly, with *dynamic programming*. An in-depth analysis of these algorithms is not a part of this text, but a reader may wish to consult the literature [Put94] that deals with *value-* and *policy-iteration*. The result of the optimization is, as described above, a policy $\pi(s)$ that maps states to actions.

3.3.3 Partially Observable Markov Decision Processes

A Partially Observable Markov Decision Process (POMDP) is a further extension of a Markov Decision Processe, the difference being that the decision maker can no longer observe the entire system state, but must instead deal with partial observations when making decisions. It is formally defined as a six-tuple (S, A, O, T, Ω, R) with:

- S: set of states,
- A: set of actions,
- O: set of observations,
- T: conditional transition probabilities,
- Ω: conditional observation probabilities,
- R: rewards.

3.3.3.1 Observations

An observation is any system output that the decision maker can *observe*. MDPs assume that the decision maker has the ability to *see* what state the system is currently in, whereas POMDPs make no such assumption, relying instead on partial observations.

3.3.3.2 Optimization

Although the reward function for MDPs and POMDPs remains the same, optimization differs in that the output is no longer a policy that maps *states* to *actions*, but rather a policy that maps *observations* to *actions*. This more realistically models reality where decision makers often do not have the ability to observe the complete state of a system, but rather receive only partial information (example: sensor data). The result of this added observation dimension is that the decision agent must now maintain a constantly changing belief state, that is initially defined and then continuously updated as actions are taken and observations received.

The computation of optimal policies for POMDPs is computationally extremely costly. The traditional policy computation approaches used for MDPs are often intractable for POMDPs and it is thus often necessary to solve approximately. A number of interesting approaches have been studied [Han98].

3.4 Simulink

Simulink is a commercial modelling and simulation tool developed by MathWorks. It is an industry standard in the field of control engineering. Models are created graphically in a block-based user interface and simulation results can be easily be analysis with plots or mode advanced tools. Additionally a large number of internal and third-party libraries further simplify modelling, especially in specialized fields such a music or aerospace.

Simulink is also tightly integrated in MATLAB, MathWorks' commercial numerical computation tool, which is widely used in academia and industry in many different fields. Simulink runs inside the MATLAB environment and can thus easily be controlled, tested or extended using the MATLAB programming language.

Simulink is especially useful for engineers because no programming knowledge is required even for modelling complex dynamic systems (example: power plants). Default configuration options and an entirely graphical interface hide most implementation details (simulation details, solver types, et). A graphical modelling interface is also useful because it allows a more intuitive understanding of the model, unlike the large mathematical matrices used in Markov Chains, Markov Decision Processes or Partially Observable Markov Decision Processes.

Figure 3.3 shows an example of the MATLAB environment, a Simulink model and plots of simulation results.

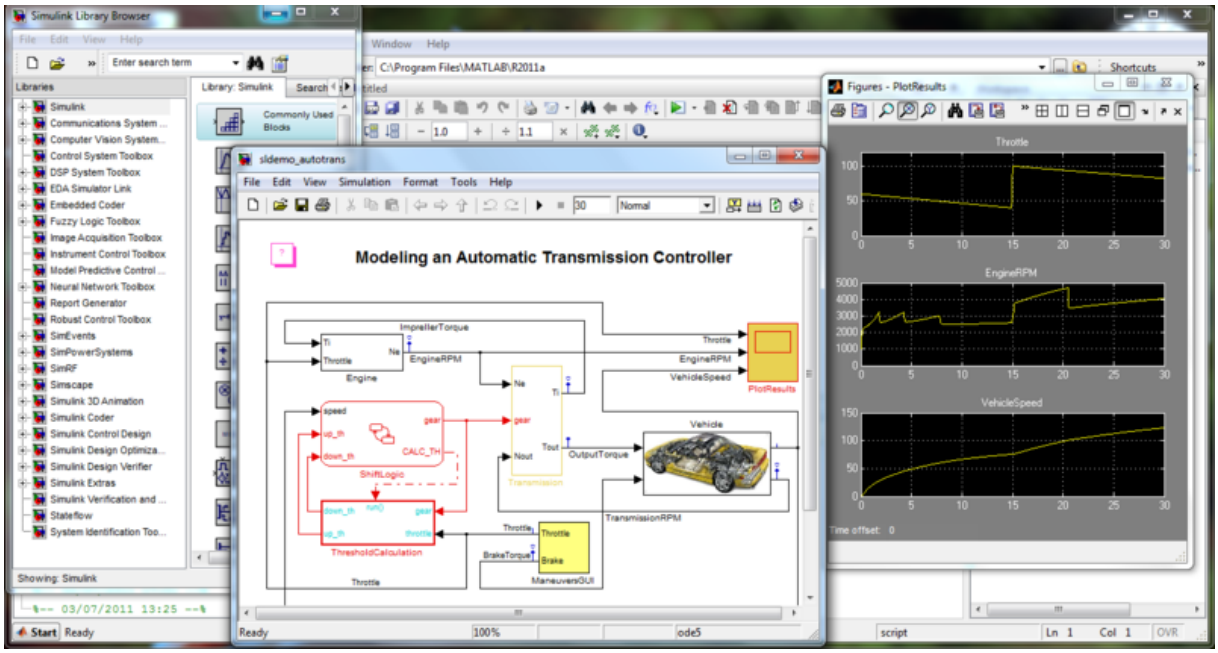

Figure 3.3: *Screenshot of Simulink*

Methodology

Methodology

4.1 Extraction

Talking about Extraction of MDPs by brute force simulation approach.

4.2 Validation

Talk about necessity to validate extracted MDP. Step response approach.

Implementation

Implementation

5.1 Extractor

This is about the implementation of the extractor.

5.2 Validator

This is about the implementation of the validator.

Results

This is the "Results" chapter!

6.1 Extractor

Talk about extractor output.

6.2 Validator

Talk about the output of the validator.

Conclusion

This is the "Conclusion" chapter!

Outlook

This is the "Outlook" chapter!

Appendix

A.1 Item 1

Previously, this and that has been done and so on.

Bibliography

[Han98] Eric A. Hansen. Solving POMDPs by Searching in Policy Space. In *UAI*, pages 211–219, 1998.

[Put94] Matrin L. Putterman. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. Wiley-Interscience, Hoboken, New Jersey, 1994.