Verification and Validation of a Pressure Control Unit for Hydraulic Systems

P. Boström*, M. Heikkilä+, M. Huova+, M. Waldén* and M. Linjama+

*Åbo Akademi University, Finland
+Tampere University of Technology, Finland

October 16, 2014

Introduction

- Verification and validation of a pressure relief function for a hydraulics system
  - Demonstrates the techniques we have used to verify and validate a complex cyber-physical system
- Verification of safety properties of the control software
  - Automated formal verification
  - Challenge: matrix calculations
  - Tested two verification tools: Our tool VerSAA and Simulink Design Verifier
- Model-based validation that the complete system fulfills safety properties
  - The system is too complex for formal verification (with reasonable effort)
  - Applied search-based testing where the search for bad behaviour is formulated as an optimisation problem

Introduction

Case study

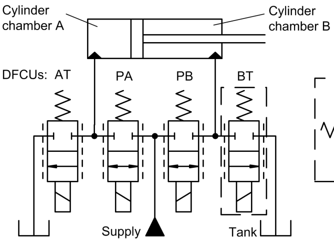
Verification of the control software

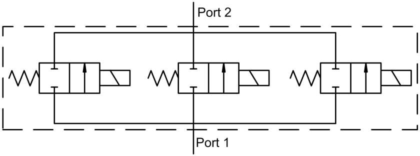
Model-based system validation

Conclusions

Digital hydraulics

- A pressure relief function is implemented as an add-on to a main controller for a digital hydraulics system
- In a digital hydraulics system complex servo- or proportional valves are replaced by simple on/off-valves connected in parallell
- Valves are grouped into Digital Control Flow Units (DFCU:s)





The pressure relief function

- Here we only consider the A-chamber of the cylinder
- Idea: When the chamber pressure p_A approaches the maximum allowed p_{max} then more valves are opened on the tank side until the pressure drops or all valves are open
  - The flow Q_A through the DFCU is increased
- A valve configuration in a DFCU is represented by a vector u containing 0:s and 1:s
- The goal of the controller is to choose the u that gives the smallest flow rate $u * Q_{max}^T$ above a limit Q

The pressure relief algorithm

The limit Q for the flow rate that the pressure relief function should provide is given as

$$Q = \frac{p_A - p_c}{p_{max} - p_c} [\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array}] * Q_{max}^T$$

with zero point at $p_A = p_c$ and $p_A = p_{max}$ requiring opening of all valves

Pressure control algorithm

1. Determine a valve configuration u_{temp} which is the possible valve combination with minimal flow above the limit Q
2. Choose the output u_{out} such that $u_{out} = \max(u_{in} * Q_{max}^T, u_{temp} * Q_{max}^T)$, where u_{in} is the input valve configuration to the pressure controller.

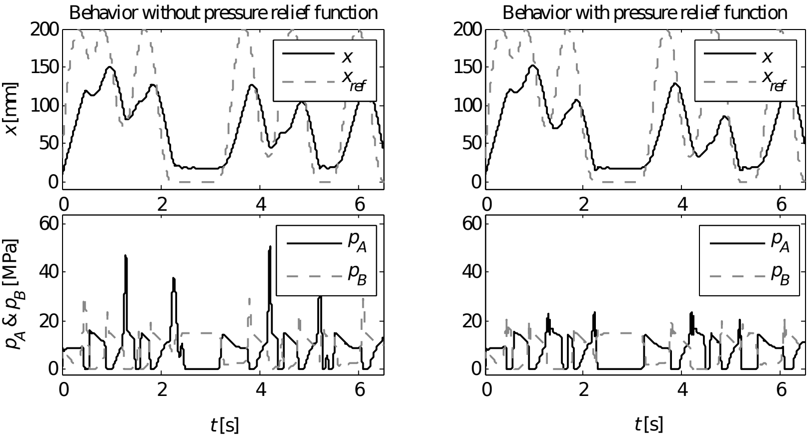
The possible valve combinations

The possible combinations the controller can choose from are given by the rows in:

$$PossibleCombinations = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

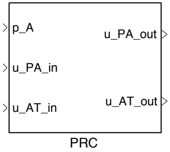
Simulation of the system

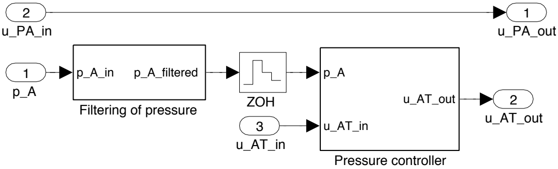
System behaviour without and with pressure relief functionality.
Maximum pressure p_{max} is set to 20 MPa



Verification of the control software

- The control software was developed in Simulink together with a simulation model of the plant.
- The pressure relief function is a subsystem in the complete model





Verification of the control software

Safety properties for the control software were identified based on the pressure relief concept.

The conditions for the subsystem PRC

- $u\_PA\_out = u\_PA\_in$
- If the filtered A-pressure is smaller than p_c, then $u\_AT\_out = uAT\_in$
- If the filtered A-pressure is greater than p_{max}, then $u\_AT\_out = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]$ (i.e. all valves open)
- If the filtered A-pressure is between p_c and p_{max}, the flow rate of the output valves is at least the flow rate of the input valves $u\_AT\_in * Q_{max}^T \leq u\_AT\_out * Q_{max}^T$

Decomposition of properties

The correctness conditions can be decomposed as correctness conditions for the internal subsystems

The conditions for the subsystem Pressure controller

- If p_A is smaller than p_c, then $u\_AT\_out = u\_AT\_in$
- If p_A is greater than p_{max}, then $u\_AT\_out = [\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array}]$ (i.e. all valves open)
- If p_A is between p_c and p_{max}, the flow rate over the output valves is at least the flow rate of the input valves $u\_AT\_in * Q_{max}^T \leq u\_AT\_out * Q_{max}^T$

The conditions for the subsystem Filtering of pressure

- ...

Verification tools

We compared two tools to check the properties

VerSAA

- Developed at Åbo Akademi University
- Contracts suitable for assume-guarantee reasoning used for specification
- Generates verification conditions that are discharged by the SMT-solver Z3

Simulink Design Verifier

- Provided as a Simulink toolbox by Mathworks
- Properties to verify given as special verification blocks or statements
- Based on k-induction and a SAT-solver provided by Prover Inc.

Contracts in VerSAA

The contract for the subsystem Pressure controller is given as:

contract :
inports :
p_A : double;
u_AT_in : matrix(double, 1, 5)
outports :
u_AT_out : matrix(double, 1, 5)
requires : all($u\_AT\_in = 0 || u\_AT\_in = 1$)
ensures : all($u\_AT\_out = 0 || u\_AT\_out = 1$)
ensures : $p\_A \geq p_{max} \Rightarrow$ all($u\_AT\_out = 1$)
ensures : $p\_A < p_c \Rightarrow$ all($u\_AT\_out = u\_AT\_in$)
ensures : $(p\_A \geq p_c \&\& p\_A < p_{max}) \Rightarrow$
$u\_AT\_in * \text{transpose}(Q_{max}) \leq u\_AT\_out * \text{transpose}(Q_{max})$
end

Verification results

- Multi-rate subsystem with two sampling periods that consists of 69 blocks
- Both tools could verify all 10 properties. Additionally, absence of runtime errors such integer over and underflow, index out of bounds and division by zero was proved
- VerSAA used 30 seconds while Simulink Design verifier used 11 minutes
- Simulink Design Verifier needs less user annotations due to k-induction
- Both tools approximated floating-point numbers by infinite precision rational numbers

Model-based system validation

After the software has been verified to satisfy its requirements, we need to show that it actually serves its purpose

- The system model is an extremely complex hybrid system (contains non-linear differential equations that do not even have analytical solutions)
- Even if we manage to prove that the model is correct with much effort, this does not necessarily hold for the real system

We have opted for using simulation-based testing to validate system correctness

- Automatic search-based test generation approach to automatically find test cases that expose flaws in the system

Search-based testing

- The idea is to formulate the problem of finding undesirable behaviour as an optimisation problem
- Optimum of the cost function is the undesirable behaviour
- Typically the problems are non-convex and there are no algorithms that are guaranteed to find the optimal solution
- Here we have applied genetic search algorithms, which have been shown to find good solutions to hard optimisation problems in practise

Search-based testing

- We are interested in testing quantitative aspects, i.e., how high can the pressure become in the system
- The system is an open system with one input signal: the piston reference position x_{ref}.
- The system has internal state - not sufficient to check instantaneous input-output
- Hence, to create a test case we need to define x_{ref} over a time interval.
- The reference position trajectory x_{ref} needs to be realistic, i.e. a signal that can be encountered in the real system.

Input signal requirements

We have the requirements for all times t in a test

$$\begin{aligned} x_{min} &\leq x_{ref}(t) \leq x_{max} \\ v_{min} &\leq \frac{dx_{ref}(t)}{dt} \leq v_{max} \\ a_{min} &\leq \frac{d^2 x_{ref}(t)}{dt^2} \leq a_{max} \end{aligned} \tag{1}$$

or in a discrete form with sampling time T_s

$$\begin{aligned} x_{min} &\leq x_{ref}(T_s i) \leq x_{max} \\ v_{min} &\leq \frac{\Delta x_{ref}(T_s i)}{T_s} \leq v_{max} \\ a_{min} &\leq \frac{\Delta^2 x_{ref}(T_s i)}{T_s^2} \leq a_{max} \end{aligned} \tag{2}$$

Test generation algorithm

To get high pressures we need to have high speeds and accelerations

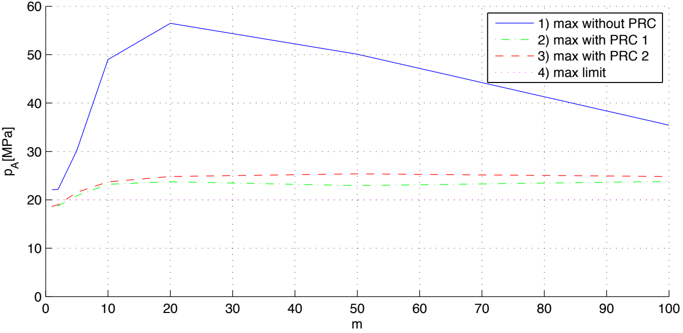
Test generation algorithm

1. Pick k pivot elements where the x_{ref} has the value x_{max}
2. Solve the constraint system for vector x_{ref} so that each element i satisfies the constraints in (2) and so that $\Sigma_i x_{ref}(i)$ is minimised. This is a linear programming problem that maximises the velocity and acceleration in x_{ref} within limits.
3. Simulate the complete system using the generated x_{ref}.

The positions of pivots are optimised by a genetic algorithm

Test generation results

Below are the maximum pressures in the A-side of the cylinder found by testing using different acceleration and velocity limits.



The maximum pressure found without PRC is an unacceptable 56MPa, while the maximum pressure with PRC is an acceptable 25MPa

Conclusions

- Presented an approach to verification and model-based validation of a pressure relief system
  - A complex cyber-physical system
- Formal automated verification proved useful for checking that the software fulfills certain correctness properties
- Search-based testing proved successful to find high pressure peaks in the original system, and to show the improvement obtained with the pressure relief function
  - This does not prove absence of pressure peaks, but the correctness proof of the software ensures that they will not be caused by faulty software