A method for evaluating the robustness of a high-speed solder paste printer

ALEXANDAR VUJADINOVIC

KTH Industrial Engineering and Management

Master of Science Thesis
Stockholm, Sweden, 2014

En metod för att utvärdera robustheten av en höghastighetsskrivare för lodpasta

ALEXANDAR VUJADINOVIC

KTH Industriell teknik och management

Examensarbete
Stockholm, Sverige, 2014

En metod för att utvärdera robustheten av en höghastighetsskrivare för lodpasta

Alexandar Vujadinović

Examensarbete MMK 2008:86 MDA 479
KTH Industriell teknik och management
Maskinkonstruktion
SE-100 44 STOCKHOLM

A method for evaluating the robustness of a high-speed solder paste printer

Alexandar Vujadinović

Master of Science Thesis MMK 2014:86 MDA 479

KTH Industrial Engineering and Management
Machine Design
SE-100 44 STOCKHOLM

Examensarbete MMK 2014:86 MDA 479

En metod för att utvärdera robustheten av en höghastighetsskrivare för lodpasta

KTH Industriell teknik och management

Alexandar Vujadinović

Godkänt 2014-11-14	Examinator Hans Johansson	Handledare Carl During
	Uppdragsgivare Mycronic AB	Kontaktperson Henrik Linde

Sammanfattning

Jetstråleskrivaren MY500 är en del av produktutbudet för PCB-tillverkning som utvecklats och tillverkas av Mycronic AB, tidigare känt som Micronic Mydata AB. Som för många liknande företag, finns det ett tydligt intresse i att minska utvecklingskostnader och göra det möjligt att ta nya produkter och uppgraderingar till marknaden snabbt. Hardware-in-the-loop-simulering är ett sätt att testa delar av ett system innan prototyper konstruerats, och det ger stora fördelar till ett utvecklingsprojekt eftersom olika idéer och systemparametrar kan undersökas till en mycket lägre kostnad jämfört med tester som använder faktisk hårdvara.

Syftet med detta magistersexamensprojekt är att skapa en plattform för hardware-in-the-loop-simulering och utvärdering för positioneringssystemet som förflyttar skrivarmunstycket i X- och Y-led. MY500-maskinen skjuter ut punkter bestående av lodpasta längs en förutbestämd bana som även innehåller information om målpositionerna för de olika punkterna. Detta hanteras av en styrenhet, CMOT3, också utvecklad av Mycronic AB, som kör ANSI-C-kod och har olika gränssnitt för interaktion med resten av maskinen. I detta specifika fall användes dess SPI-gränssnitt för att kommunicera med den simulerade systemmodellen, som kördes på en Speedgoat-dator som har en systemarkitektur specifikt framtagen för att köra Simulink-modeller i realtid.

Innan detta projekt påbörjades så har redan två olika examensprojekt genomförts kring modelleringen av positioneringssystemet för MY500. Ett av dessa hade målet att implementera den i en HIL-plattform, och även om det ej gjordes på grund av tekniska problem så var slutsatsen att det var rimligt att fortsätta arbetet. Detta projekt använde ett annorlunda tillvägagångssätt för systemstrukturen och resulterade i den första lyckade HIL-plattformen för X- och Y-aktuatorerna för MY500, och den har framgångsrikt producerat realistiska testresultat. En analysmetod baserad på Multivariate Process Capability har tagits fram och använts för att poängsätta data från simuleringarna.

I den avslutande delen har testdata som analyserats med den föreslagna utvärderingsmetoden diskuterats, och förslag har tagits fram kring fortsatt arbete på både plattformen och utvärderingsmetoden.

KTH Industrial Engineering and Management

Master of Science Thesis MMK 2014:86 MDA 479

A method for evaluating the robustness of a high-speed solder paste printer

Alexandar Vujadinović

Approved	Examiner	Supervisor
2014-11-14	Hans Johansson	Carl During
	Commissioner	Contact person
	Mycronic AB	Henrik Linde

Abstract

The MY500 solder paste jet printer is a part of the PCB production product range developed and produced by Mycronic AB, formerly known as Micronic Mydata AB. As with many other similar companies, there's a clear interest in reducing development costs and making it possible to bring new products and product upgrades to the market quickly. Hardware-in-the-loop simulations is a way to test parts of a system before any prototypes are constructed, and it brings great advantages to a development project since various ideas and system parameters can be examined at a much lower expense when compared to tests that utilize actual hardware.

The purpose of this master's thesis project was to create a hardware-in-the-loop simulation and evaluation platform for the MY500 machine's X and Y axis nozzle positioning system. The MY500 machine ejects dots of solder paste on a PCB prior to the placement of electronic components, and it does so along a pre-selected trajectory which also contains information about the target positions for the dots. This is handled by a control unit, also developed by Mycronic AB, called CMOT3, which runs ANSI-C code and has various interfaces for interaction with the rest of the machine. In this particular case, its SPI interface was used to communicate with the simulated system model, which ran on a Speedgoat computer which has a system architecture specifically designed for running Simulink models in real-time.

Before the project was started, there have already been two thesis projects that have dealt with modeling of the MY500 machine's positioning system. One of these had a goal of implementing it in a HIL platform, and although it wasn't done due to technical issues, the conclusion was that it's reasonable to continue the work. This project used a different approach for the structure and resulted in the first successful HIL platform for the MY500 X and Y actuators, and it has been shown to produce realistic simulation outputs. An analysis method based on multivariate process capability evaluation has been developed and used to score the data from the simulations.

In the concluding part, the test data that has been analyzed with the suggested evaluation method is also discussed, along with some suggestions of further work on both the platform and the method itself.

ACKNOWLEDGEMENTS

I want to begin by thanking the following people for their support throughout my thesis project:

Carl During, Mycronic AB

Henrik Linde, Mycronic AB

Robert Axelsson, Mycronic AB

Bobbi Ferm, Mycronic AB

Joel Rundgren, Mycronic AB

Simon Eriksson, Mathworks Sweden

It has been very interesting and educating to spend time on Mycronic doing this project, and I am grateful for having been given the opportunity to work with you.

Alexandar Vujadinović

Täby, September 2014

NOMENCLATURE

The purpose of this section is to provide a reference of the notations and abbreviations that appear in the report.

Notations

Symbol	*Description*
c_p	*(traditional process capability indices)*
c_{pk}	´´
c_g	*(process capability indices proposed as part of this project)*
c_{gk}	´´
c	*sensitivity for c_g and c_{gk}*
Δ	*normalized position error (to specification limits from -1 to +1)*
μ	*normalized mean error*

Abbreviations

CAN	*Controller Area Network*
CS	*Chip Select*
HIL	*Hardware-in-the-loop*
I/O	*Input/Output*
LSL	*Lower Specification Limit*
SPI	*Serial Peripheral Interface*
SSH	*Secure Shell*
SUT	*System Under Test*
USL	*Upper Specification Limit*

TABLE OF CONTENTS

1. INTRODUCTION

This chapter aims to give a brief overview of what this master's thesis project was about, the technical context in which it was done, and what the goals were. Note that the company Micronic Mydata AB changed its name to Mycronic AB during the course of this project. The newer name will consistently be used in this report.

1.1. Background

Mycronic AB develops and manufactures precision machinery for the electronics industry. Among their products for printed circuit board production is the MY500 machine, a jet printer for solder paste which is used on circuit boards prior to the placement of components, which is done by a separate pick-and-place machine. This thesis project was centered on developing a virtual test platform for the robustness of the MY500 unit's X and Y axis positioning system which moves the solder paste nozzle over the board that's being printed.

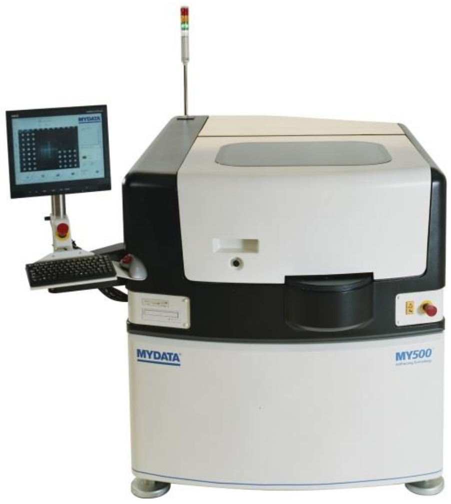


Figure 1: MY500 jet printer (image courtesy of Mycronic).

This project was started because Mycronic had an interest in having a platform that performs tests of the MY500 positioning system using a simulation of the X and Y positioning dynamics which interacts with the control algorithm that's developed for the real-world MY500 jet printer. Such a platform would offer some advantages due to the fact that using a real machine would be more time-consuming and cause greater expenses. Another benefit with a simulation-based platform is the repeatability – large sets of tests can be done independently of complicated factors such as component wear.

The MY500 machine (see Figure 1) contains a number of CAN nodes, one of which bears the designation CMOT3 and controls a number of actuators that includes the X and Y motors. A CMOT3 controller had been made available for use throughout the project. The CMOT3 was interfaced with an SPI cable to a Speedgoat real-time computer, to which the Simulink model of the positioning system was uploaded.

1.2. Project goals

The purpose of the project was phrased in the following way by Mycronic:

"This thesis project should investigate and propose a method for how simulated hardware can be used to evaluate the robustness and find the limitations of an implemented embedded controller (implemented in ANSI-C). The method should also be used for regression tests of production code."

The goals of the project:

1. Investigate the state of the art that can be applied to robustness testing of a two-dimensional planar servo-actuated high-speed gantry.
2. Propose a method, based on the information gathered in the investigation, which hypothetically can be used for robustness tests and regression tests of legacy code for the MY500 jet printer's X and Y positioning system.
3. In particular, analyze if the proposed method could be used to evaluate and analyze hardware changes, i.e. if a specific motor can be changed without lowering the performance below the requirements specified for the MY500 unit.
4. Further, verify the proposed method by using an existing HIL platform including a virtual model of the MY500 mechanics connected to the real controller boards.

As this is a master's thesis project, an important additional goal was to identify suitable research questions as soon as possible (as there were none in the beginning) and undertake efforts to answer those. They are presented with a more detailed discussion in their own section of this chapter.

1.3. Stakeholders

The performer of this master thesis project, and the company Mycronic AB which had an interest in having a simulation-based test platform for the positioning system in one of its products, could be seen as the primary stakeholders here. Secondary stakeholders were the university KTH, specifically the Department of Machine Design which publishes the thesis reports in Mechatronics, Mathworks which had to provide some support with the used xPC Target platform, and also Mycronic's customer chain which benefits from technical advances in printed circuit board assembly.

1.4. Project outline

The following approximate approach has been suggested in the beginning of the project:

1. Search for information about the concept of robustness and robust performance in a control theory context and how it can be defined and assessed.
2. Meet up with a Mathworks representative to set up the Speedgoat unit and get familiarized with the xPC Target real-time software environment.
3. Modify the plan and extend it with more details, based on the information that has been gathered and the relevance that the found publications have to the project.
4. Study the MY500 positioning system and the earlier mentioned modeling work, and prepare an x-y model with an interface that can be used for the HIL simulations.
5. Make an outline on how the robustness shall be tested and assessed, and specify the tests that shall be done as well as how the data should be processed.
6. Design a prototype which performs HIL tests according to the selected specifications in and presents data to the user in a way that gives a good overview.
7. Using the prototype, perform repeated HIL tests and gather data on the robustness and the possibilities to change the value of some of the constants in the modeled system.
8. Discuss the gathered data, and assess the possible benefits of gathering data in this way, as well as the limitations of the used method.

Since this was not a group project, a group-focused methodology would be redundant, and the work was thus done by making plans and setting up milestones related to the different technical issues as they appeared during the development of the test platform.

1.5. Delimitations

This project was started with a clear focus on the development of a robustness testing platform based on HIL simulations of the X and Y actuators that are used in the MY500 machine. While the selected system model needed some attention before it could be successfully implemented in the platform, there project scope was limited to the test platform only; there were no studies on new approaches for the modeling of the system and how accurate those would be.

1.6. Research questions

At the early stages of the project, there were no clearly defined research questions. The research questions arose in a later stage, when some information searching had been done and some progress had been made with the system.

An approach based on the evaluation of uncertainty regions for the position and velocity of the jet printer nozzle was considered for a short time, but it was abandoned afterwards, and efforts were instead made to create a method inspired by process capability evaluation. The goal of the plant itself is to place solder paste dots as close as possible to a set of targets, and a method based on these discrete events was thought to be better than adding unnecessary complexity (that doesn't relate much to the performance of the plant) by using some sort of continuous evaluation of the printer nozzle's trajectory.

The research questions were:

1. *Can the process capability ratios c_p and c_{pk} be used to assess the performance and robustness of a planar positioning system with an elliptic specification limit?*
2. *Can the limitations of using c_{pk} for the robustness and performance analysis of a planar positioning system with an elliptic boundary region and a not fully normally distributed output be overcome by introducing an alternative index that uses Gaussian functions to score the error data, and produces the same results as c_{pk} in the one-dimensional ideal case where the output is normal?*

The background behind the first question is the finding of an example of a robustness study that used process capability analysis to evaluate the robustness of a plant for hydroforming of aluminum components. Further literature studies showed that there are limitations in using a method designed for one output variable in this case, and the second question was a natural consequence of that.

The second research question was phrased after it was found that there are no commonly accepted standards in evaluating multivariate process capability, and that the methods used very different approaches and had their own limitations (some assumed perfect normality, some were for rectangular specifications in the two-dimensional case, and so on). Proposing a new method that's simple and suitable for this context seemed to be the most reasonable choice in this case, since the goal was to make a testing platform where different test cases can be compared with each other, and not to design an index that competes with the specific features of the already proposed multivariate methods.

This question was approached through analysis of data sets that were generated through repeated HIL simulations by the platform that was developed, also as part of this project. The analysis method itself was also examined by testing it with sets of randomly generated one-dimensional normal distributions to find out if it would behave similarly to the common process capability indices as the mean error and standard deviation varies. In the end of the project, the suggested methods were discussed more closely in the context of the test platform that had been developed.

1.6. Disposition

The project was started with a research phase, in which a search for information was done in order to establish a useful frame of reference for the remainder of the project. It consisted of both literature studies as well as studying two previous projects that related to the MY500 machine's positioning system. The literature search was done on the Internet, partially using the KTH article search service, and partially through Google. The research phase is presented in its own chapter following this one. During the research phase the aim was to achieve better understanding and insights on how the concepts of robustness testing and performance can be approached, so that a new method which fits in to the characteristics and limitations of this project's hardware platform could be developed and evaluated.

After that, the implementation stage is presented, where both the approach to the evaluation of the system and the work with the technical aspects are presented. In this stage, many problems were encountered and had to be dealt with as they arrived, in order to move on with the tasks.

Finally, the last chapters present what has been achieved. In the Results chapter, examples of simulation and evaluation data are shown together with explanations on their usefulness and how they should be interpreted. A discussion and a critical analysis follows in the next chapter, and in the end there are some recommendations on the further work that could be done based on this project, as well as recommendations on how to relate this project to the real-world machines that the simulations represent.

2. FRAME OF REFERENCE

2.1. Literature studies

2.1.1. Hardware-in-the-loop (HIL) simulation

A HIL setup can be defined as a platform which uses a hardware prototype that's connected to emulated parts of a system, with a bidirectional data flow. There is however some variation in how a HIL setup is defined and the circumstances may vary greatly between different contexts – some tests may even deal with open-loop control instead of closed. One common aspect is that a HIL setup often focuses on certain parts of a system, and very peripheral influences on the dynamics of a system may be neglected, such as a simulation of how a building behaves during an earthquake, where the building's influence on the actual earthquake can be considered small. (Fathy, Filipi, Hagena, & Stein, 2006)

HIL simulation is used for performing system-level tests of embedded systems that are repeatable, comprehensive and cost-effective. The response of sensors and actuators is electronically emulated, and the actual I/O interface is used for communication. Having a real-time model that simulates parts of the system under test (SUT) and the relevant interactions with its environment is necessary for successful HIL simulation. Together with synthetically generated operator commands, it becomes possible to automate test sequences and have precise repeatability of the tests. However, as with any simulation, it's critical that it's possible to demonstrate that the simulated environment adequately represents reality. (Ledin, 1999)

There are multiple benefits associated with using HIL simulation, and it is applied to a wide variety of technological fields as a development tool. Prototyping with HIL is usually quicker and more cost-effective than using physical hardware, and can reduce the time it takes to bring a complete product to the market. It is a safe, repeatable and non-destructive way of testing systems where parameters have been changed far from their nominal values. Further, a HIL simulation is often a more controlled environment than its equivalent physical test, and this leads to better repeatability. (Fathy, Filipi, Hagena, & Stein, 2006)

Common alternatives to HIL include Model-in-the-loop, Software-in-the-loop and Processor-in-the-loop, although these simulation techniques are not real-time. (Ágústsson, 2013)

A quick overview of the HIL platform created in this project can be seen in Figure 2.

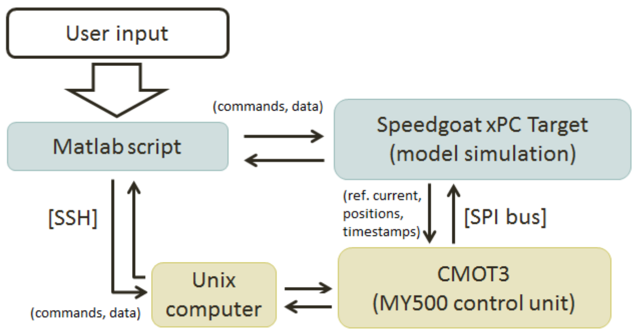


Figure 2: HIL platform, in the context of this project. The CMOT3 controller is the SUT here. It's interfaced to a simulation instead of a real MY500 machine's X and Y motors.

The xPC Target computer is where the X and Y axis simulation is done, and the CMOT3 is a Mycronic-developed control board which calculates reference currents, handles the trajectories and target position data for the various print tasks, etc. The Matlab script which interacts with the user and sends the different commands to the xPC Target computer as well as the CMOT3 control board is in its entirety a product of this project. These are all explained in more detail further on in this thesis.

2.1.2. The concept of robustness in control theory

During the information gathering phase of this project, most of the encountered literature that touched upon the concept of robustness of systems was focused on the control aspect of this topic – that is, how to achieve robust control. Even though the control aspect is already done, the related literature that was found gave nevertheless some good initial insights on how to proceed with the task of creating an evaluation method for the test platform.

In classical control theory, analysis of stability margins is used. There are two points of interest here – the *gain margin*, which can be defined as the inverse of the (open loop) system's gain where its phase response is 180°, and the *phase margin*, which can be defined as the difference between the system's phase response at 0 dB gain and 180°. On the Nyquist plot, the margins can be seen by looking at where the loop transfer function intersects the real-axis and the unit circle, respectively. However, the stability margins of classical control theory do not represent robustness well. It is possible to have an infinitely large gain margin and 90° phase margin, and at the same time be close to the critical –1 point. Simultaneous variation in both margins is ignored when analyzing them individually. Further, in systems with multiple inputs and outputs, analyzing one loop at a time doesn't give information about how the cross-coupling of simultaneous variations in multiple loops affects the system as a whole. (Chiang & Safonov, 1998)

In the book "Multivariable Feedback Control" from 2001, Skogestad and Postlethwaite state their approach to the robustness problem as follows:

1. Determine the uncertainty set by finding a mathematical representation that describes it.
2. Check robust stability. Determine if all of the plants in the uncertainty set result in a stable system.
3. Check robust performance. If the robust stability is satisfied, determine if specifications are met for all of the plants in the uncertainty set.

The uncertainty of the model is not the only aspect that should be considered when dealing with robustness. Other examples include failures of sensors and actuators, physical constraints, changes in objectives, etc. Also, if the design is based on optimization, problems can occur if the mathematical objective function doesn't describe the control problem properly. (Skogestad & Postlethwaite, 2001)

The most important goal for a control system (in addition to providing stability) is to ensure that the plant meets its performance specifications. A common way of describing these goals is to specify the sizes or size limits for certain signals of interest, such as the errors in position or velocity. There are several definitions that are used to describe signal sizes (norms). The most appropriate method depends on the situation. (Zhou, 1995)

L_2 and H_2

For signals, the norm L_2 is simply the root-mean-square of the signal. For systems, the equivalent norm is known as the H_2 norm, which can be interpreted as the RMS gain of the system throughout its frequency range:

$$\|x(t)\|_2 = \sqrt{\int_{-\infty}^{\infty} x(t)^2\, dt}\ ,\ \|G(s)\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} trace\big(G(j\omega)^H G(j\omega)\big)\, d\omega} \qquad (2.1)$$

L_∞ and H_∞

The infinity norm L_∞ is the maximum value of the signal. For systems, the H_∞ norm is the peak gain of the system throughout all frequencies (Skogestad & Postlethwaite, 2001):

$$\|x(t)\|_\infty = \max\big(x(t)\big)\ ,\ \|G(s)\|_\infty = \max\big(G(j\omega)\big) \qquad (2.2)$$

From the robustness testing perspective, the most interesting aspect here is to find a good way to evaluate performance, so that system tests with different parameters can be compared against each other. While it would in some cases be relevant to look closer into the mentioned signal and system norms, and make an evaluation method based on them, the significantly discrete character of the MY500 machine's dot placement task led to the conclusion that the field of process capability analysis might give better insights on how to evaluate the plant's performance.

2.1.3. Univariate process capability analysis

Process capability indices (PCIs) can be seen as summary statistics for the output of a production process. (Kaya & Kahraman, 2010)

The central concepts of process capability are the uniformity of the analyzed process output and the variability of quality-critical characteristics. PCIs (sometimes called process capability *ratios*) are quantitative evaluation methods of those aspects, and the most commonly used ones are denoted as c_p and c_{pk}. Variations and the specification limits around the process target value are usually normalized in terms of multiples of the standard deviation, represented with the Greek letter sigma (σ).

The process capability ratio c_p represents the (six-sigma) spread of the process output relative to the specification boundaries. For example, if the distribution of the output spans across half the size of the range within the specification boundaries, the c_p value is 2 (as it would need to be twice as wide to reach the specification limits). A common practice is to make sure that the process capability is 1.5 for new processes, and expect that it will over time decrease to 1.33.

It is however important to note that c_p is not affected by how well the output is centered. Because of this, another process capability ratio designated c_{pk} has been introduced, which represents the actual capability of the system, as opposed to the potential one. The c_{pk} ratio is obtained by taking the distance between the mean output and the closest specification limit, and dividing it by three times the standard deviation. If the process output is centered, c_{pk} will be equal to c_p. A c_{pk} value below 0 corresponds to a situation where the mean falls outside the boundaries.

$$c_p = \frac{USL-LSL}{6\sigma} \quad , \; c_{pk} = \min\left[\frac{USL-\mu}{3\sigma} \; , \; \frac{\mu-LSL}{3\sigma}\right] \tag{2.3}$$

USL stands for the upper specification limit, LSL for the lower limit, σ for the variability of the process output, and μ for its mean. (Montgomery, 2009)

While the mentioned indices offer a useful and simple way of quickly getting an overview on how a production process performs, traditional univariate process capability analysis can sometimes render misleading results if the analyzed output variables from a process have a correlation. There are often correlations in industrial processes and it is beneficial to analyze them in order to reduce output variability. (Kaya & Kahraman, 2010)

A 2011 study utilized the c_p and c_{pk} indices as part of a robustness evaluation of a machine that uses warm hydroforming to produce certain aluminum components, for which the studied quality characteristic was the long-term uniformity in thickness. (Koç, Agcayazi, & Carsley, 2011)

2.1.4. Multivariate process capability

Since the plant in this project has a bivariate output (X and Y positions), multivariate process capability analysis would be relevant here.

Products created by industrial processes have generally more than one quality characteristic, and there has been some activity in developing multivariate process capability indices (MPCIs) that can assess performance without being limited to analyzing one output at a time. An issue that makes MPCIs fundamentally different is that the tolerance region can't be represented by simply having two points along a line. (Pan & Lee, 2009)

There are no commonly accepted and established standards in this field of study. MPCIs were first published in the late 1980s, but activity in this field has remained relatively low until around 2005.

In the licentiate thesis report "Contributions to Multivariate Process Capability Indices" by Ingrid Tano, there is a list that mentions 16 different methods that have been discussed in literature from 2000 to 2011.

They can be divided into four groups (some methods belong in multiple groups), based on:

1. the ratio of process output region and its tolerance region.
2. the probability of non-conforming process output.
3. usage of Principal Component Analysis.
4. other characteristics.

The proposed methods in the list are of varying usefulness. Confidence intervals have been derived for four of them. Six of the different indices don't require the output to be multivariate.

Tano made a comparison of the four methods that had their derived confidence intervals. All of them were in the first category, and two of them also make use of principal component analysis additionally. They also require normal distributions.

Another method devised by Tano & Vännman is presented in the same publication. It is based on principal component analysis of data that has been normalized to all of the specification ranges, so that the range of conforming values stretches from -1 to +1. Just like the four methods that were discussed more closely by Tano, this one was also based on "…the assumption of multivariate normality". (Tano, 2012)

In 2010, so-called Robust Process Capability Indices were proposed by Kaya & Kahraman. The context was however not robustness testing or robustness evaluation, but rather to introduce a modification to traditional PCIs so that they would be able to observe autocorrelation, and then be used as a basis for new capability indices that make use of fuzzy sets. Overall, the computational complexity here was deemed to be unnecessarily high for this project. (Kaya & Kahraman, 2010)

The observation that there were no commonly established standards was also confirmed through the multitude of published MPCI method suggestions that were found while searching for information during the research phase.

2.2. Existing models of the MY500 positioning system

When this project was started, there existed two models of the positioning system for the jet printer nozzle. The models are of different complexity, but both have been made with the sole purpose of representing the movements of the printer head in the X and Y directions. The test platform developed in this project uses an adaptation of the later model, without any extensions such as the Z-axis.

2.2.1. Rundgren's master thesis[1]

In 2011, modeling of the MY500 positioning system was done by Joel Rundgren, using the SimMechanics extension of Simulink. It resulted in a realistic model of the system which included most of the vital parameters, although some work remained to be done at the time of the completion of the thesis. (Rundgren, 2011)

The model was subsequently extended to include the dynamics from the support structure as well as the Y axis, before the next thesis project on the MY500 started. (Ágústsson, 2013)

[1] *"Modelling, Control and Simulation of a Jet Printer Robot", see reference list for further information.*

2.2.2. Ágústsson's master thesis[2]

Eiður Ágústsson's thesis from 2013 involved simplification of the x-y model so that it would not need the SimMechanics toolbox, along with some comparative work to compare how well the purely Simulink-based model followed the original version. The reason behind the simplification was that it was not possible to run the SimMechanics model in a HIL context due to the significantly longer computation times. The simplification itself was largely successful, although other technical complications remained, which prevented fully successful tests back then.

Ágústsson mentioned problems with lowering the sample time as processes interfered, as well as glitches during certain sample times and situations, which caused the system to go into different error modes and fail. There was also no successful simulation of the Y axis. The problem appeared after the controller code was extended to send data using SPI. (Ágústsson, 2013)

2.2.3. Comparison of the two existing models

The simplified model in the latest MY500 thesis work by Ágústsson consisted of two single-input single-output systems in parallel, without any interaction between them. It was considered a reasonably accurate simplification of the previous model, although one needs to be aware of that it neglects the high-order dynamics from the vibrations in the structure. Some comparisons between them have been made in that thesis to illustrate the general character of the differences.

An example of how it looks like can be seen in Figure 3, which is a graph of the acceleration step responses for the two different models.

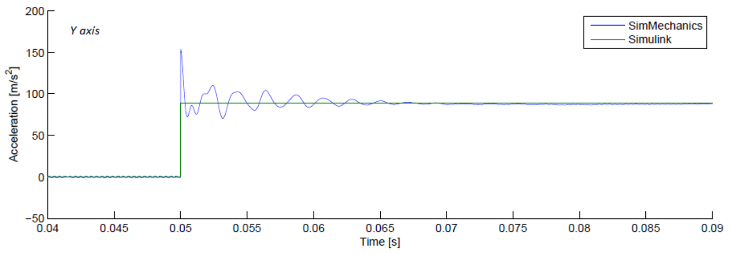


Figure 3: 26A step response for the Y axis of the SimMechanics model. Note the higher-order dynamics as well as the short time in which they have any significant effects. (Ágústsson, 2013)

Another notable aspect is that there's a significantly longer computation time for Simulink files that use blocks from the SimMechanics extension package. An average computation time of 50 ms per step for Rundgren's SimMechanics model was observed, compared to 8.5 µs for the simplified model.

2.3. Hardware

2.3.1. MY500/CMOT3 positioning system

A locally developed controller unit called CMOT3 (see Figure 3) is used in the MY500 machine. It features digital signal processing capability as well as a field-programmable gate array for I/O. At the time of the previous thesis project involving the MY500, the control loop was set to run with its nominal 200 µs cycle. (Ágústsson, 2013)

The MY500 system places solder paste by performing continuous straight movements of the printer head while ejecting paste with regular intervals. The resolution in the placement of the dots is much narrower than the distance covered during one control cycle (nominally 200 µs). Performance, here defined as how close the dots are placed compared to the target positions, depends on the ability to follow the trajectories closely without too much variation in either speed or position while a strip of dots is being ejected.

[2] *"Hardware-in-the-loop Simulation of a High Speed Servo System", see reference list for further information.*

After startup, a command named `msetdyn` is needed to activate the actuators properly. For both the X and Y motors, it starts by moving the printer head until it collides with the walls, so that the operation ranges can be determined. Then, a series of pre-set movements and measurements are performed in order to calibrate the control algorithm. A shorter variant of the command can be called by appending the number 1 (example: `q_ msetdyn x 1`). This may be preferred in situations where it will be used repeatedly – the full length startup is significantly longer (30 vs. 15 seconds, approximately).

Before a print job can be done, an on-board camera is used to measure the locations of certain calibration points on the PCB, so that the chosen trajectory can get the proper offsets and be aligned with the copper surfaces. This step is only necessary in real print jobs because the PCB can't be placed at exactly the same location consistently – in simulations it can be ignored in favor of consistency and shorter testing times.

The source code to the CMOT3 is written in C code. While it is complex and handles many tasks that depend on each other, it offers some flexibility and possibilities of modifying its operations. After a modification of the program has been compiled, the resulting binary file can be loaded to the CMOT3 with the command `loadmot`. This command also doubles as a quick way of performing a pseudo-restart of the system to be used as a known common state from which all individual test runs can be started.

Only the X and Y control code is used in this project. Trying to initialize the other actuators such as `z` and `fi` would in this case result in error messages as they are not connected and don't have any models that simulate their behavior.

The trajectory files are in the form of lists where 5 commands appear repeatedly in groups, but with different parameters. This is an example of one such segment:

```
movtp x 0 1244024 117306 1249104 119936 62 6 1
movtp y 0 1003799 117306 1003799 119936 62 6 1
movtp z 0 132299 117306 132299 119936 62 6 1
pumptp fi 0 25000 117306 106327 119936 62 6 0 200 1
jett e 117306 119936 62 6 0 0
```

As the `z` and `fi` actuators aren't initiated, the commands regarding those are ignored without any complications regarding functionality – fortunately the CMOT3 code is written to handle this situation without going into an error mode.

Table 1 gives an example on how the performance is specified for the customer side, and the magnitude of the MY500 unit's precision.

JET PRINTING SPEED AND ACCURACY	
Rated speed and accuracy (components per hour eq.)	30000 cph
Reference board throughput	28000 cph
Single board repeatability 3σ (X, Y)	54 µm
Single dot accuracy @ Cpk=1.33 (X, Y)	80 µm
Deposit accuracy @ Cpk=1.33 (X, Y) QFP100C	33 µm
Deposit accuracy @ Cpk=1.33 (X, Y) 0603	40 µm
Deposit repeatability 3σ (X, Y) QFP100C	19 µm
Deposit repeatability 3σ (X, Y) 0603	24 µm

Table 1: MY500 performance specification example. (Mydata, 2010)

The preliminary requirements for the simulated tests are therefore roughly based upon this information.

2.3.2. xPC Target

Mathworks xPC Target is a real-time testing environment that enables execution of Simulink and Stateflow models on a target computer for HIL tests and other applications. Together with the associated hardware product range, it offers compatibility with a wide variety of I/O methods. (Mathworks, 2003)

The system that runs the 32 or 64 bit xPC Target kernel is called the Target PC. The Host PC is the platform where the applications are created and developed. The kernel on the Target PC runs the real-time application and communicates with the I/O boards. It also receives a timer interrupt signal, either from its own clock or from an external source. The communication between the Target and the Host PC is either done with a TCP/IP Ethernet connection or with RS-232. This connection is needed for uploading and running Simulink models, exchanging data, tuning the parameters, etc. (Mathworks Training Services, 2013)

In this project, the FPGA-based IO module IO313 was used in the Speedgoat computer. The CMOT3 control card was connected to the module’s SCSI-III port to facilitate the SPI communication link.

The CMOT3-Speedgoat setup can be seen in Figure 4.

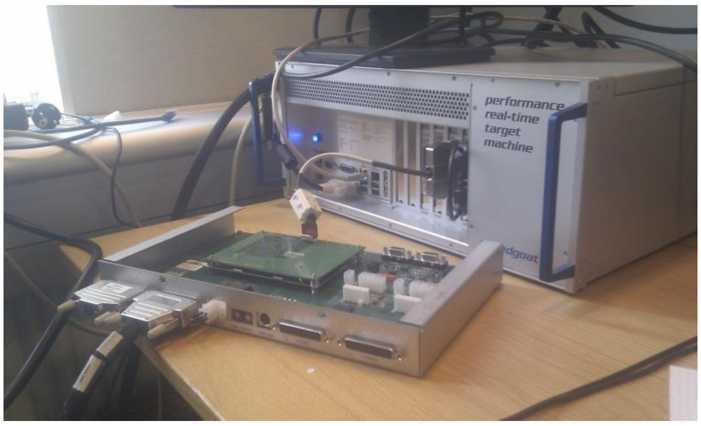


Figure 4: CMOT3 unit, here connected to a Speedgoat real-time computer with an SPI cable. The UNIX computer, which handles commands and other related tasks, is not in the picture.

3. IMPLEMENTATION

3.1. Specifications and goals

3.1.1. Overview

Some delimitation had to be done when it came to the robustness aspects. As mentioned in the framework, there are many possible causes for uncertainty in a model. The purpose of this project is to propose a set of methods to examine robustness with the focus on identifying possibilities of hardware changes in the MY500.

Three main aspects of robustness were presented earlier in the framework. However, only two of them are relevant here, since a fully functioning (and stable) system is examined in this case. The focus is on the system's robust performance. Specifically, it is of interest to the company to be able to investigate how certain changes in the parameters affect the system and how critical they may be to the overall quality of the printing.

Process capability and robust performance

As explained in the framework, the analysis of gain and phase margins is inadequate for robustness assessment, especially in this case where there are two systems that affect the performance. Since this basically is a production process where the quality characteristic is how close the solder paste dots are placed compared to their reference positions, a relevant way of assessing performance would be analysis of process capability.

There are however two output variables here, since the dots are placed on a two-dimensional surface. Traditional process capability analysis is based on examining one output variable.

Analysis of each axis separately would in effect make the tolerance region square (or more generally described, rectangular). This would make the analysis more tolerant to diagonal errors, which is something that's undesired. It would also ignore certain problematic correlations, for example the situation where disproportionate amounts of errors are located along a circle around the origin of a graph where both the X and Y errors are plotted against each other.

Non-normal positional error distribution

During the printing of paste onto a PCB, strips of solder paste dots (see Figure 5 for an example of target data) are placed with various velocities, using various interval lengths, and in various angles, depending on which component footprint they are for and how they're turned. Unless special attention is given to ensure that all segments of the print job use the same type of movement and have the same number of paste dots in each strip, an error distribution with high normality cannot be expected.

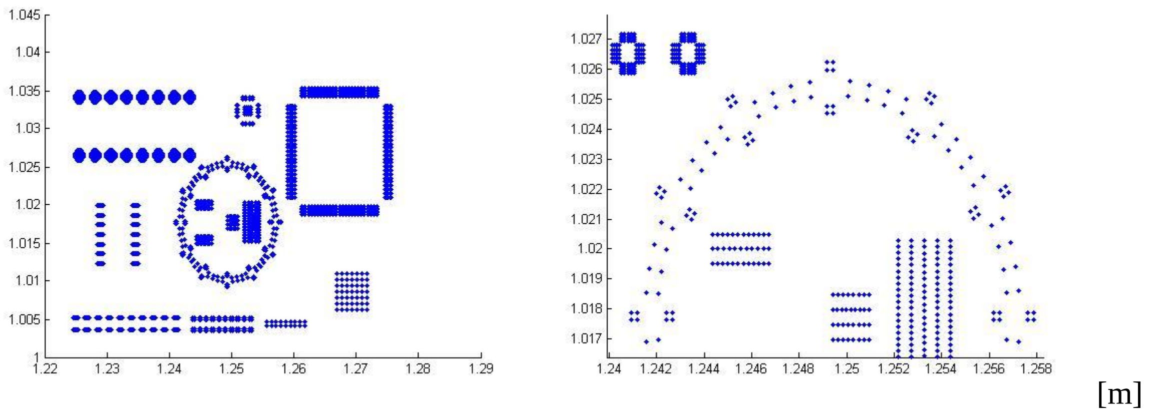


Figure 5: Overview and zoomed-in view of the target positions for the trajectory file that was

used for the tests in this project (`servoMovTPLog_Demo20_Senju_LF.txt`). There are 3008 individual solder paste targets along this file's trajectory.

3.1.2. Input and test delimitation

Early on in the project, it was decided that the implementation would be centered on a Matlab script, which interacts with both the user and the systems that are part of the testing platform. In the beginning of the program, a list of all system constants should be presented to the user. After that, the user should be able to choose if the test should involve one or two constants, which constants that are to be tested, and their respective range and resolution.

Desired characteristics of an analysis method for the output data may include:

- Compatible with any number of output dimensions
- Doesn't need to be estimated by numerical means
- Doesn't require a perfectly normal distribution (and reacts in a proper way)
- Reacts to both distribution and mean-value accuracy (as c_{pk} does)
- Reacts to correlations that affect the output
- Produces results that are similar to those from traditional capability analysis (especially when the process output data is normally distributed)

Since there apparently is no multivariate method that has reached widespread acceptance, proposing a new method would be a good choice in this case. Another aspect here is that the method in this project does not need to compete with the existing MPCIs that have been suggested – the important thing is that it's able to evaluate the process output in a way that's easy to understand as well as quick to calculate and implement in an automated robustness testing platform that compares different test cases.

3.2. Testing platform and technical issues

Before any work was done with developing an analysis method, a simulation platform had to be created and brought to a functional state, so that data could be generated and evaluated.

3.2.1. Modifications of the byte handling in Simulink

The previous model by Ágústsson published in 2013 included a rather complicated system of blocks for switching, type conversions, and routing, in order to handle the incoming and outgoing bytes. Any attempts to do any adjustments and modifications to it proved to be hard and time-consuming due to the large amounts of inter-dependent blocks. Therefore, it was simplified considerably by modifying the old project file to make use of Simulink's "Matlab Function" blocks, in which Matlab code can be used for direct interaction with the other Simulink blocks.

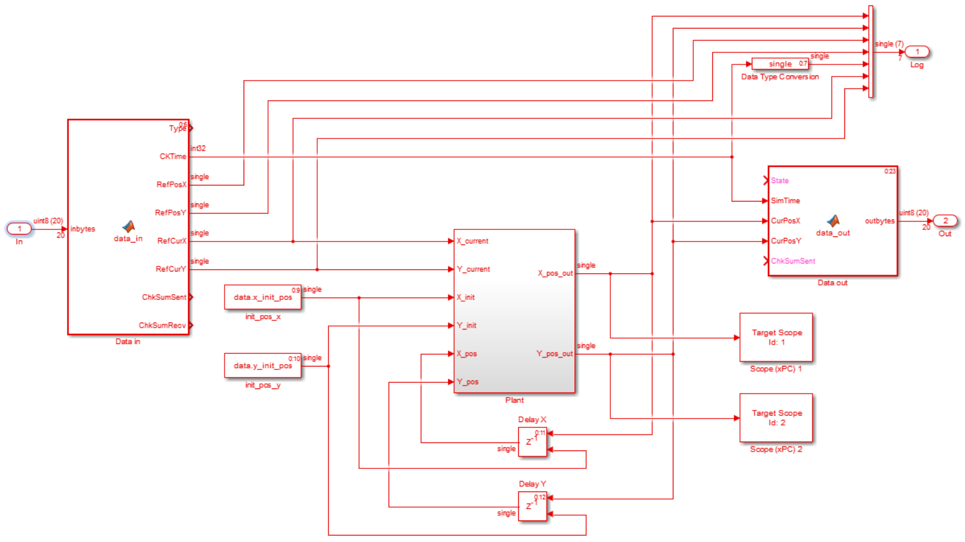


Figure 6: The new data handling system (code given in Appendix C).

The incoming data from the SPI buffer is in the form of a vector containing 8-bit words, with the last position corresponding to the first byte. These are processed in the “Data In” block, where they are arranged together correctly and cast into their respective data types. The “Data Out” block has the opposite behavior – it receives data which is split into 8-bit words and arranged right-to-left before feeding it to the SPI output buffer. Unused ports can be seen on these blocks in the figure – they represent fields that were removed during the course of the work because they were not needed, and to keep the message shorter so that more computation time would be available on the CMOT3 side.

3.2.2. Positioning system model

Ágústsson’s lower-order system model was selected as the basis for the simulations in this project. It was then extended with a collision model to simulate the scenario where the printer head reaches any of the four internal walls in the machine. The reason why it had to be done is that the initialization commands (`msetdyn`) for both axes involve movements where the printer head collides with the walls; the purpose is to measure the extent of the operation range so that collisions don’t occur while a board is being printed. There were already reset inputs for the velocities and the positions, and these were reused for this purpose, as the simplest way to simulate a collision was to reset the velocity and let the positions saturate at specified constant values.

A non-perpendicular collision would of course need a more complex model to be simulated realistically – although this complexity is not needed in this case because the ranging is done one axis at a time. After the range has been measured, it is saved with a safety margin so that these types of collisions never occur (and if they happened, the print job would have been halted anyway).

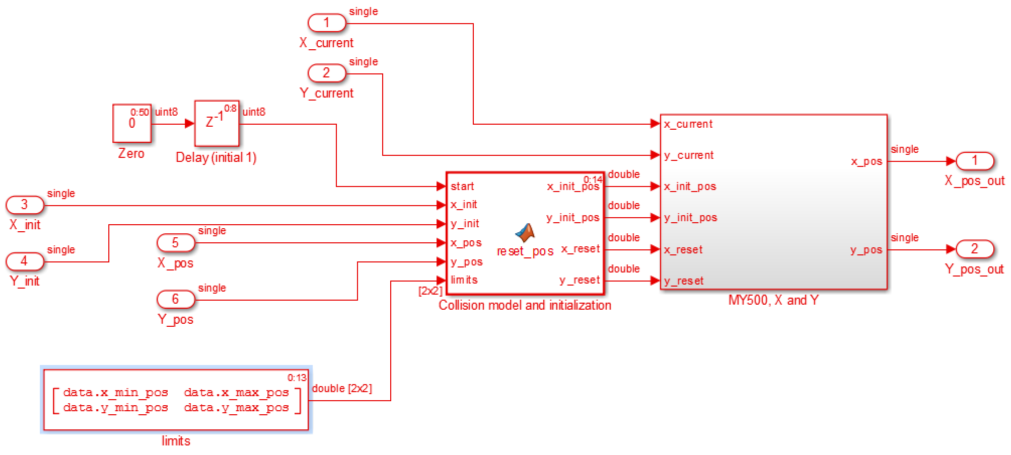


Figure 7: Contents of the "Plant" block seen in Figure 6. The MY500 block is based on the model from the previous thesis project.

3.2.3. Communication problems

A CMOT3 control unit had been set aside by Mycronic, to be used in the tests. Transfer of data between the CMOT3 and the xPC Target unit is carried out in a way that's equivalent to how the CMOT3 communicates in a real MY500 machine.

Some modifications in the CMOT3 communication protocol had to be done. Previously, bytes were sent one-by-one through SPI, with a Chip Select (CS) signal that only covered eight bits. In the xPC Target environment, CS is supposed to span across an entire data frame, regardless of how it is segmented internally. Variable frame length is not supported, and the segmentation exists only as a pre-selected definition – all bits are sent as one word (for instance, if measured with an oscilloscope, the transfer of twelve 8-bit words will appear as identical to three 32-bit words).

The SPI data buffer can be accessed with its specific Simulink block, and each function call needs about 30 µs to be completed, regardless of the chosen frame size. This situation created a major problem: having 23 of these events during every cycle (200 µs) led to a CPU overload as soon as communications were started.

Further, even if it had worked, it would be impossible to disambiguate the single-byte frames without counting them after a monitored start-up, or adding redundant data for synchronization. This would unnecessarily increase complexity and possibly be detrimental to the overall robustness of the testing platform itself.

It also proved to be impossible to keep the original 10 MHz communication speed when running the CMOT3 together with the Speedgoat unit, because of data corruption. Another obstacle was the long access times to the Speedgoat SPI buffer as well as unreliable functionality of the interrupt routine. It should be noted that the problems did not involve the CMOT3 controller, and various strategies were tried such as introduction of start/stop delays between the Chip Select signals and the data stream, but the problems remained. After multiple conversations with Speedgoat representatives, the decision to lower the communication frequency was taken.

Bobbi Ferm at Mycronic helped with the frequency adjustments, as it is not controlled by a function in the main CMOT3 source code. The frequency for the clock signal that delimits the individual bits in a message is created through division of a 100 MHz signal. Through repeated tests it was found that reliable transmission of messages could not be carried out at frequencies too far above 5MHz.

The cable was shortened and a ferrite core was added as an attempt to solve this problem, but it had no effect at all on the communications. Oscilloscope measurements also showed no apparent problems with the signal at 10 MHz. This was a recurring topic on the telephone conferences with the Speedgoat representatives in Switzerland. Later, it was found that the voltage levels in the Speedgoat unit's IO313 card were incompatible with the 3.3V system that the CMOT3 uses.

In a Speedgoat datasheet, it is stated that communications are carried out through a TTL transceiver. However, in the same section, "V-out-high – 3.8 V minimum" is found as well. (Speedgoat GmbH) As the IO313 card was described as a TTL-capable unit, it card had been bought by Mycronic with the expectation that the transceiver should detect a high logic level at 2.0 V, because that level is commonly found in TTL contexts. At 5 MHz, the signals denoting individual bits apparently seem to last long enough to be registered without any noticeable problems, even though 3.3 V is below the guaranteed threshold.

3.2.4. CMOT3 modifications

A modification was done to one of the CMOT3 drivers so that it would handle request-based switching of the CS signal, so that it wouldn't be reset after each byte. In the beginning, a more automatic approach was suggested, with a "CS latching" SPI command variant that would be used for all bytes except the final one. However, after consultation with Mathwoks, the manual method was chosen, since the Speedgoat used start/stop delays when configured as an SPI master, and it was conjectured that it would be safer to imitate this behavior in the CMOT3, to reduce possible glitches.

The message itself was reduced to 20 bytes per cycle. The previously leading "status" byte was never assigned to any use, and the different commands were sent through SSH using a set of macros in the main Matlab script, so it was removed. The last two bytes for checksum handling were also left out, because through repeated testing it was found that significant errors would make the controller halt. Thus, the appropriate approach here is "either ensure a fully functioning two-way communication, or don't perform the test at all"; which makes checksums redundant. See Figure 8 for a photograph of a measurement that illustrates the communication cycle clearly.

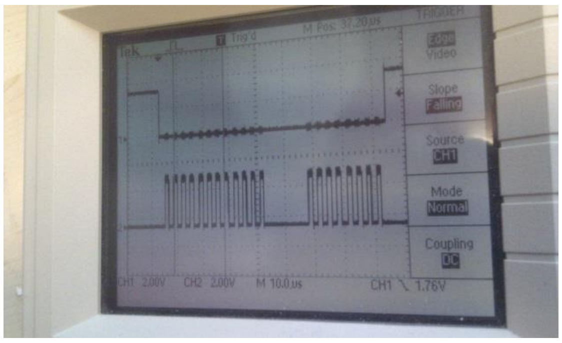


Figure 8: Oscilloscope snapshot of a communication cycle. The top graph is the CS signal, and the bottom graph shows the messages (all have the value 255 in this case). Note the delay in the middle that coincides with the calculation step of the CMOT3 unit (see appendix B).

Due to the longer messaging time caused by the reduction in communication frequency, the control cycle period had to be doubled to 400 µs. To avoid seriously compromising the reliability of the test results, other steps were needed to compensate for the effects of this modification. The goal here was to make sure that the simulations would run at half speed while having the same behavior and producing the same results as if the system wasn't modified in this way.

At first, all of the integration blocks in the Simulink model had to be set with a gain of 2, because the sample time became twice as large as before. Some minor modifications had to be done to the CMOT3 as well, so that it wouldn't trigger errors and halt the program because of the unexpectedly long time between the samples. An overview of the control/communication cycle can be seen in Appendix A.

3.2.5. Command macros

Acquiring data from the CMOT3, and interacting with it in general, proved to be a challenge as well. While there is a dedicated Linux computer for the CMOT3, where commands easily can be entered in a terminal, doing the same things automatically (and using a Matlab script) from another computer is not as straight-forward, and acquiring the responses in a useful way is not as simple as it may appear.

SSH was chosen as the method of communication because the terminal on the Linux side is compatible with it, and Matlab allows the sending of text strings to the Windows terminal. The next step was to find a terminal-compatible SSH client, and the choice fell on Plink (the terminal variant of Putty), as it proved to be easy to use, especially together with custom parameters.

All messages had to be initiated with login data, and that was prepared.

```
host     = '[redacted]';
username = '[redacted]';
password = '[redacted]';
login    = ['plink ' host ' -l ' username ' -pw ' password ' '];
```

After that, the message normally follows. However, it wasn't enough, due to a local setting that treated external commands differently. Therefore, it had to be unlocked by another command, which was found after some searching.

```
msg_head = 'TERM=xterm; export TERM; source .profile; export
IS_LOCAL_TTY=yes; sleep 0; ';
send     = @(msg) system([login, msg_head, msg]);
```

The macros can then be relatively easily defined, for example as:

```
initctrl = @() send('q_ setpar canxy 12 38 1');
```

Some other macros were also done in order to handle the returned data from certain commands, such as error codes and timestamps of placed dots, which are received by Matlab in text format.

All macros were saved in a separate file (see Appendix D), so that an updated list would be available for both the simulation and analysis scripts, as well as when they're needed for work on another Matlab-based CMOT3 project.

3.2.6. Other issues

There were still some problems after the slowed-down system was brought to a functional state.

One notable issue was a problem with the axis ranges. Both the X and Y axes couldn't function much beyond 0.95m and 1.45m, which is somewhat narrower than the ranges found in the MY500 machine. The failures happened at the axis initializations (`msetdyn`) in both cases. A possible reason could be a process timeout that happens earlier than it should because the system runs at half speed. However, the true cause behind this behavior was not found, and the chosen action to move past this issue was to simply avoid using trajectory files that go beyond these more narrow boundaries. See Figure 9 for an example plot.

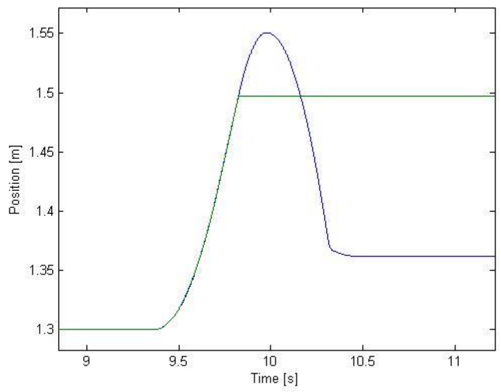


Figure 9: Example of a failed X axis initiation (limit set to 1.6 m). The reference trajectory becomes stuck at one value and the inputs to the X motor cease shortly thereafter.

The `acc_eval.pl` script was needed to get the timestamps for the individual dots, which are placed using much shorter timescales than the control cycle. The script saves the log in the CMOT3 unit's memory as `canxy.dcl`, and then the file has to be called with the `cat` command so that the script receives its contents and parses it with the function `strread` together with a regexp-based search criterion. As mentioned before, the terminal output data from the CMOT3 arrives in text format to Matlab, and they had to be parsed in various ways so that it would be possible to process and analyze the received information.

The dot placement data couldn't be recorded properly after the CMOT3 modifications had been done – only the target positions, timestamps and the dot indices within their strips, were correct. Therefore, the Simulink trajectory log from the Speedgoat unit's output log needed to be used. By using the timestamps from the dot log together with the timestamps received through SPI, the positions for each dot could be interpolated. Before the interpolations, a processing step was implemented for the trajectory data to remove repeated entries, as well as mitigating a time offset that was noticed through empirical observation of output data from multiple simulations. After that was implemented, dots could be recorded along with the trajectories. See Figure 10 for a zoomed-in plot that shows both the dots and their printer head trajectories together with target data.

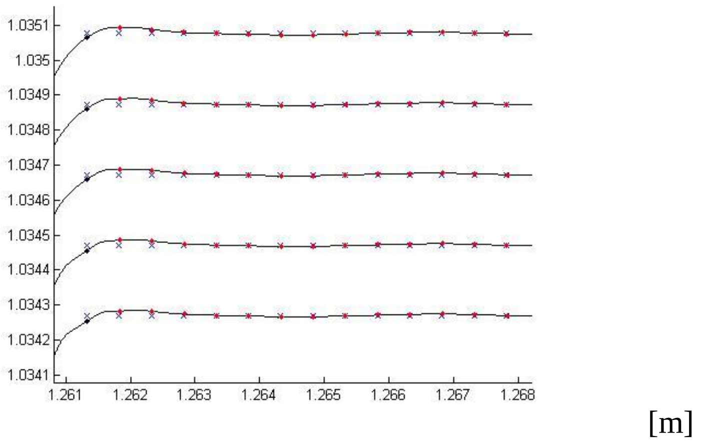


Figure 10: Example plot of dot placements compared with their respective target positions. Left-to-right movement. Note that the X axis is magnified considerably more than the Y axis.

The testing of system parameters is also expected to involve situations where the examined range stretches beyond what the controller can handle without going into an error mode and halting its operation. Therefore, care had to be taken to make the testing and evaluation platforms flexible enough so that they wouldn't crash when some data sets are incomplete.

3.3. Simulation script

As explained earlier, a simulation script written in Matlab code was chosen to be the nexus of the implementation, as it could be used (after some preparatory work) for automatic sending of commands that trigger the different steps of the simulations, as well as saving relevant data in a pre-specified manner. In the set of files associated with this project, the simulation script is named `user_run.m`.

3.3.1. Overview

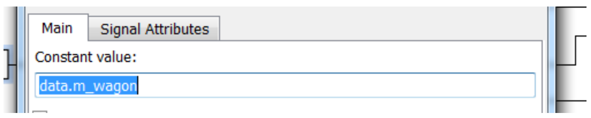
The following list provides an approximate overview of what's done on every test run.

- Update the model data struct so it's run with the correct parameters
- [SSH] Reset the CMOT3 control unit (`loadmot`)
- Compile and upload Simulink program
- Run Simulink program
- [SSH] Initialize periodic communication
- [SSH] Initialize control mode
- [SSH] Start X axis (short version, ca. 15 seconds)
- [SSH] Start Y axis (short version, ca. 15 seconds)
- [SSH] Run print job
- Stop Simulink program
- [SSH] Get dot placement data (`acc_eval.pl`)
- Save simulation logs, dot location data, indices, etc.

3.3.2. Model parameters

On the host computer, an "xPC Target Object" is automatically generated upon compilation to facilitate the interactions with the uploaded Simulink program. The start and stop commands can be used as part of a normal Matlab script, and some information can be requested from the target computer even while its program is running. Data can be extracted from the Speedgoat target computer in two ways – either by directly making requests to read certain variables, or by placing top-level output blocks in order to feed data into the output buffer. (Mathworks Training Services, 2013)

All of the system constants are held in a struct, which is updated for each simulation index, and all the references to the system parameters in the Simulink file were changed to refer to the corresponding fields in the struct instead (Figure 11).



Constant value:

data.m_wagon

Figure 11: All Simulink system constants were redefined as references to a struct.

A timestamp is generated in the beginning of the script, which is used to generate filenames that identify which test a specific simulation is associated with. For example, if a test is started March 1st 2014 07:05:00, the filename would become "HILtest_20140301T070500", followed with test indices or "meta" for data that is common to the whole set of tests.

Three test modes were created: "Nominal system test", which performs one test run of the system with its specified constants, "Test one variable", which allows the user to examine performance with one varying system constant, and "Test two variables", which does the same, but for two parameters.

The first option is not suitable here, since the readouts of multiple variables can't be made simultaneously, which introduces errors that don't exist in the simulation itself. These problems can be avoided by choosing the second option and feed the relevant data to the output buffer, where all rows are associated with a specific place on the time vector (which also can be requested). The only possible issue here is the limited size of the buffer, although it proved to be large enough for the simulations in this project.

3.3.3. Data fields in results log

As the simulations may take considerable amounts of time, Matlab's ability to store selected data on the hard drive as `.mat` files was used, so that analysis can be done on separate occasions. The testing script saves them to the folder `\Results\` with the previously mentioned nomenclature that the analysis script later can recognize.

Field	Contents	Other info
`completed`	Info on whether the test was completed	First saved as 0 and then set to 1 at end of script
`datestamp`	Exact time at the start of a simulation	`datestr(now,30)`
`data`	Nominal parameters for the model	
`varnames`	Model parameter names	
`testvars`	Number of model parameters to examine	
`HWI_dir`	Trajectory file location	In the CMOT3 memory
`HWI_file`	Trajectory filename	In the CMOT3 memory
`var1`	Index of tested parameter 1	Only for simulations with multiple runs
`var2`	Index of tested parameter 2	Only for simulations with multiple runs
`values1`	All tested values for parameter 1	Only for simulations with multiple runs
`values2`	All tested values for parameter 2	Only for simulations with multiple runs

Table 2: Contents of the `.mat` file for metadata.

Field	Contents	Other info
`slrtdata`	Copy of all the data in the xPC target object	
`dotdata`	Copy of `canxy.dcl` as acquired from `acc_eval.pl`	Saved after formatting

Table 3: Contents of the `.mat` files for simulation results. If a test consists of multiple simulations, then there are several of these files in the same folder.

3.4. Output evaluation

3.4.1. Selection of evaluation method

A simple and naïve way to evaluate system performance would be to examine the norms of the error signals as presented in one of the paragraphs in the framework chapter. However, the purpose of this system is to place dots of solder paste, and the intermediate positions between the placement events do not affect the performance directly. Therefore, a more direct focus on the dot placement itself is the obvious choice here.

As explained earlier, some method related to process capability would create a good overview about how well the system performs as a certain parameter is changed. An example of a robustness study that used the c_p and c_{pk} indices was found during the research phase of the project, although it was an analysis of one process output variable. The discrete character of the dot placement task makes it easy to view the printing as a production process rather than a control system that simply strives to reduce the size of the positional error at all times throughout the whole reference trajectory. Thus, a decision was taken to use an evaluation method based on process capability, but for multiple variables.

The method used for selecting an appropriate evaluation scheme was to generate a random output with a normal distribution (for an example, see the histogram in Figure 12), subject it to variations in mean value and size, and then compare different suggested methods with traditional process analysis by graphic visualizations.

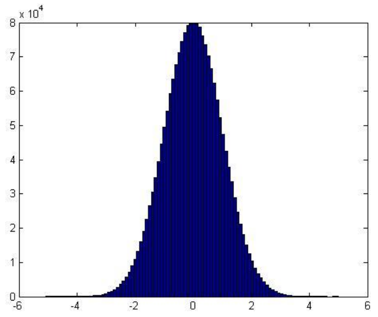


Figure 12: A histogram showing an example of a normal distribution created with the function `randn` *in Matlab.*

3.4.2. Evaluation index c_g

At first, the position error data is normalized using offsets and linear scaling, so that the targets are located at the origin and the distance to the specification limit for every variable is 1.

The evaluation of the dot placements is done by scoring the dots individually based on their distance from the target, followed by a calculation of the average. The score is referred to as c_g hereafter, because of the Gaussian function that is used to score the individual dots.

$$c_g^* = \frac{1}{N}\sum_{n=1}^{N} e^{\frac{-(\Delta_n-\mu)^2}{2c^2}} \tag{3.1}$$

Here, e represents the base of the natural logarithm, Δ_n represents the (normalized) error for each dot, and μ represents the offset. The constant c in the denominator represents the root-mean-square width of the scoring function's peak (a Gaussian curve), and represents thus the sensitivity against anomalous distributions that affect performance negatively.

Finally, the result is scaled by a constant, k, to ensure some similarity to the result obtained through traditional process capability analysis (although it should be noted that significant differences remain).

$$c_g = k \cdot \frac{1}{N}\sum_{n=1}^{N} e^{\frac{-\Delta_n^2}{2c^2}} \tag{3.2}$$

A suggested scaling method here is to let c_{pk}=1 and c_g=1 be equal in the case where there is no mean error. This was however only selected for simplicity and closeness to the values commonly encountered when dealing with c_p and c_{pk}, so that they would diverge the least in that approximate region. For c=1/36, k thus needs to be about 12.1. For c=1/6, k should be about 2.27.

3.4.3. Comparison with c_{pk}

In the one-dimensional case, the suggested method has both similarities and differences when compared with c_{pk}. Assuming that the process can keep the mean value centered on the target, the two methods behave similarly for high sensitivities (low c values). However, c_g reacts more to mean value shifts, decreasing more sharply with increasing mean errors.

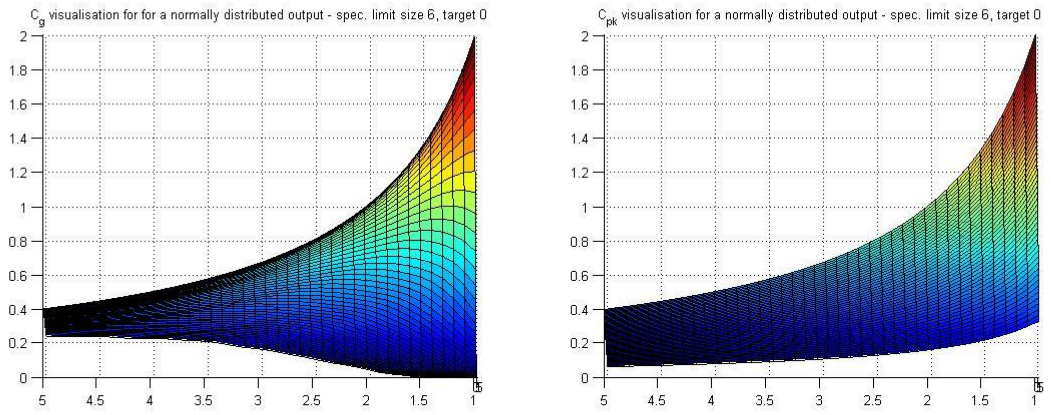


Figure 13: Comparison between c_g and c_{pk} for a randomly generated normal distribution. Here, c=1/36 has been used, and the results have been scaled properly.
(X axis: standard deviation of distribution)

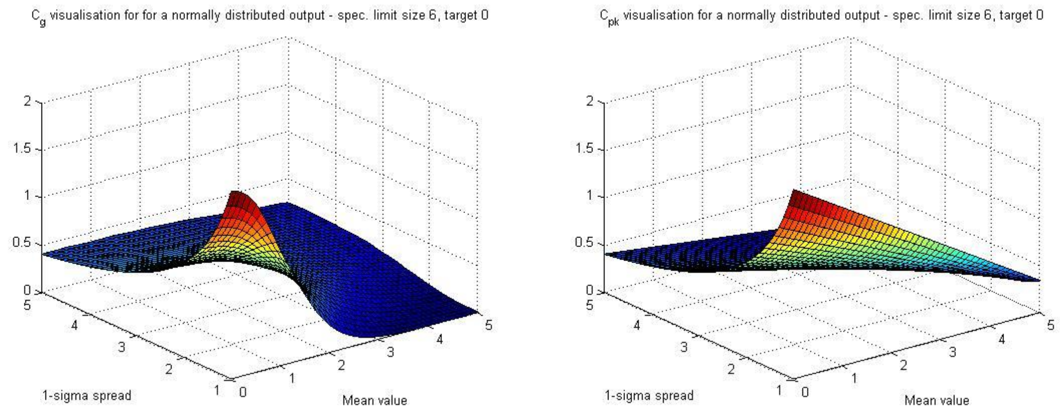


Figure 14: Three-dimensional view of the graphs in Figure 13.

Note that this is a one-dimensional analysis of ideal process output data with a normal distribution, made just to illustrate the differences between c_g and c_{pk}. While the motivation behind this method may not be apparent in that specific case, it does have certain features when the output variables are more than one. For example, if the errors correlate in an anomalous way, this method could detect it while the c_p and c_{pk} values aren't affected considerably (a hypothetical extreme case illustrating this is presented in section 4.1.2).

Traditional process capability analysis does not exhibit this feature, as it's based on analyzing one output variable at a time, and assumes that there's a normal distribution.

3.4.4. Evaluation index c_{gk}

Since the similarity to process capability analysis was found to be high for low mean value errors (and high sensitivity), there's an opportunity to modify the suggested method and compensate for the differences throughout the region that's shown in the previous graphs.

$$c_{gk} = |1 - \mu| \cdot k \cdot \frac{1}{N}\sum_{n=1}^{N} e^{\frac{-(\Delta_n - \mu)^2}{2c^2}}$$

Modifying the formula in this way creates an index that's similar to c_{pk} both when the standard deviation and the mean value of the process output vary. Note how the first factor is similar to the calculations done for c_{pk}, in which the output mean is compared to each of the two specification limits.

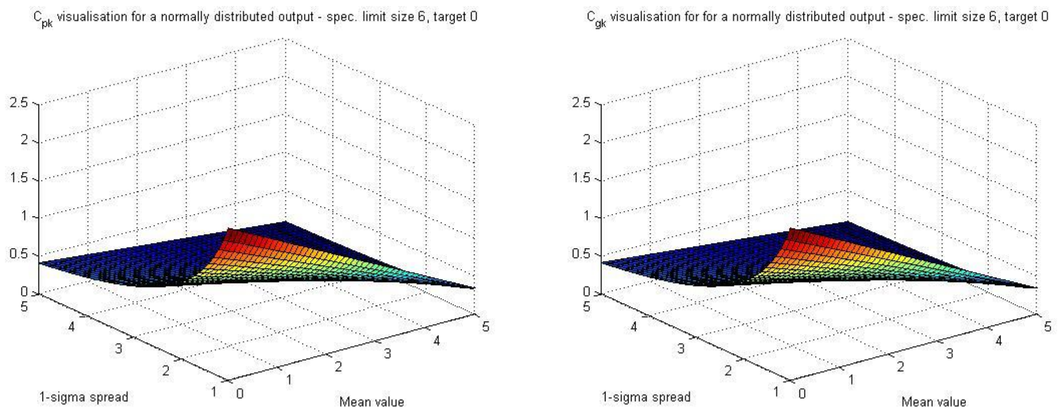


Figure 15: Side-by-side overview comparison of c_{pk} and c_{gk}.

The difference between c_{gk} and c_{pk} was found to be negligibly small (magnitude 10^{-2} to 10^{-3}) within the previously used range, for c=1/36. See Figure 16.

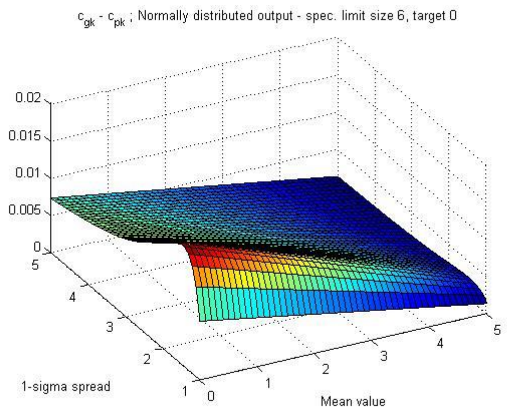


Figure 16: General appearance of the difference between the two indices. Some small variation was encountered here due to noise from the randomly generated distributions that were tested.

Since there are no significant mean value errors in the data that has been encountered in this project, the index c_g has been used instead of c_{gk} when performing the evaluations. Another contributing factor was the fact that the idea to modify c_g into c_{gk} appeared quite late in the project, when some calculations already had been made with c_g instead.

3.5. Interfaces

3.5.1. User interface for simulation

When the simulation script `user_run.m` (Appendix E) is started, the contents of the data struct with the model parameters are presented to the user. Thereafter, the user can select three different test modes.

```
0: Nominal system test
1: Test one variable
2: Test two variables
```

```
Select test mode:
```

If a nominal system test is selected, the script will immediately try to contact the CMOT3, start the Speedgoat model, and perform one test run with the nominal parameters, as presented to the user.

In the case where one of the two other options are chosen, the user will be prompted to specify which parameter(s) to vary between the test, their specific range(s), and the number of increments. The last option can therefore take much time to complete, as the number of test runs is the product of the number of increments for both variables.

After a simulation is completed, the user can turn on the simulation and use the Matlab terminal window to try macros and other commands manually if it’s needed.

3.5.2. User interface for evaluation

A separate script was created for the evaluation of the data rendered by the simulations. In the project files, it can be found under the filename `analyze.m` (Appendix F). Depending on the type of test that's picked by the user for analysis, different selections of data are presented.

In the beginning of the script, the contents of the results folder are presented to the user. When the user selects a folder, the metadata is loaded to identify the test mode that was used. After that, the associated simulation data is processed and presented. Appendix B shows an example of how it appears from the user’s perspective. See also the example graphs in Figure 17.

If the selected log is from a nominal test (that is, a single test run with the nominal parameters), the trajectory is presented along with an X-Y plot of all the errors. In the terminal window, statistical data on the dot placement is also displayed to the user.

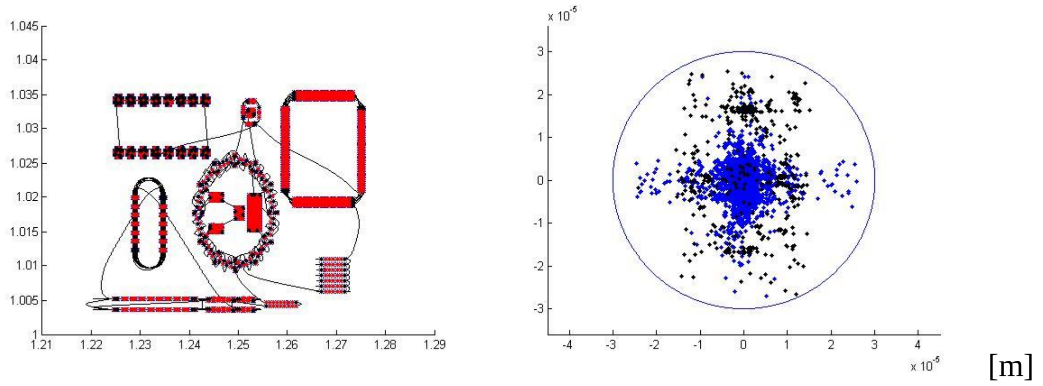


Figure 17: A set of graphs from a simulation with nominal parameters. The 30 µm tolerance radius was arbitrarily selected for the purpose of illustrating this example. The black dots represent the leading dots of the printed strips (which are prone to larger errors).

```
Standard deviation X : 5.0415µm (First dots: 7.5038µm, Ten starting dots:
5.3167µm)
Standard deviation Y : 6.8281µm (First dots: 12.4584µm, Ten starting dots:
7.2643µm)
Cp value X :  1.9836 (First dots: 1.3327, Ten starting dots: 1.8809 )
Cp value Y :  1.4645 (First dots: 0.80267, Ten starting dots: 1.3766 )
Cpk value X : 1.9737 (First dots: 1.2709, Ten starting dots: 1.8701 )
Cpk value Y : 1.4694 (First dots: 0.71981, Ten starting dots: 1.3824 )
```

Here calculation of the constant c_g renders the result 1.27 with c=1/6, and 0.92 with c=1/36 (in both cases, k has been set to scale the index for similarity with c_p, as described earlier).

Note that specific attention is given to the first dots, as well as the first ten dots in a series. This is due to the local best-practice that has been established through experiences with jet printing. Dots are placed in sets of dots along straight lines, called “strips”, and the system tends to have the largest errors in the beginning. Therefore, they are highlighted in the presentation of output data. Analysis of the first 10 dots (all, if the strip is shorter) is done in the machines to give a good overview with more data, and therefore it's also included here.

When a test consisting of multiple simulations is started, the info presented to the user will instead be graphs showing standard deviations of the errors, as well as the c_g index.

In the case where the simulation was started with parameters beyond the operation range of the CMOT3 controller (causing the simulation to halt prematurely), the user is warned of that there are some incomplete sets of data in the selected folder, so that it can be taken into consideration. An example of its appearance in the Matlab terminal window:

```
Processing 20 simulation logs. Please wait..............
Warning: Inconsistent dot count (24 vs. 3008).
```

It should also be noted that the structure of the scripts allows for easy modifications of both the model and the data saving and processing if there are other tasks and evaluations that need to be performed.

4. RESULTS AND ANALYSIS

In this chapter, some selected results from different simulations are discussed. The aim is not to give complete and final answers on which exact values the different system parameters in the Simulink model should have, but rather to present a good overview on how the analysis method works and can be used in this context.

4.1. Analysis of data from a set of simulations

4.1.1. X axis motor constant example

A test series consisting of 70 tests was done on the model's motor constant for the X axis (Mc_x). Its nominal value, as found in Ágústsson's old project files, was 15.6 Nm/A. It was tested in the range of 9 to 50, with equally sized increments between each test. See Figure 18.

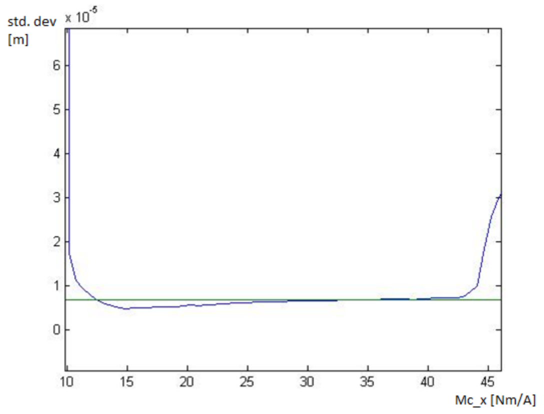
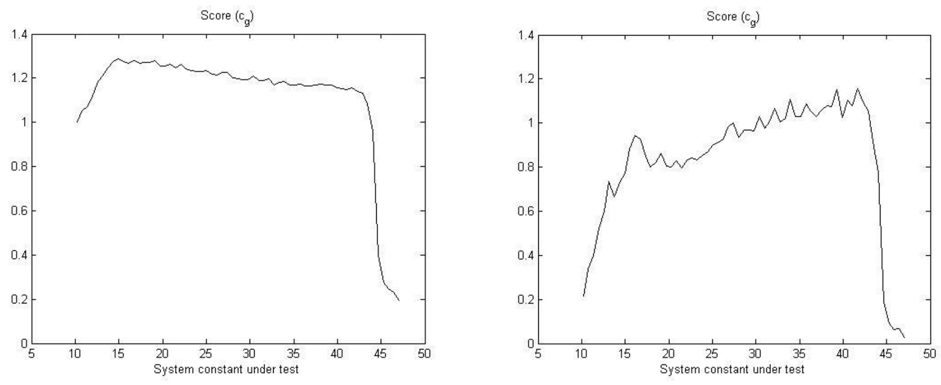


Figure 18: X and Y standard deviations of process output from a test run with varying Mc_x (the constant curve represents the Y axis).

As with traditional process capability indices, c_g indicates higher performance with a higher number. On both sides, the curve ends at the points where the CMOT3 went into an error mode. Since the output does not exhibit perfect normality, this is a good opportunity to show how the sensitivity of the c_g formula (3.1) can be tuned, and how it may appear like.



Mc_x [Nm/A]

Figure 19: The formula that rendered the left curve has had its sensitivity decreased by using c=1/6, while the right one has a higher sensitivity with c=1/36.

Note how in Figure 19 the general appearance of the graph radically changes as the sensitivity constant is changed. While a higher sensitivity was empirically found to make c_g (and especially c_{gk}) appear more similar to c_{pk}, lowering it can be useful in cases like this, where the distribution is considerably non-normal and prone to noise.

4.1.2. A hypothetical extreme case

Unfortunately, the model on which the HIL simulations are run isn't complex enough to simulate anomalies in the output that come from the disturbances that one axis causes to the other, or other complex behaviors. However, some comparative work can be done anyway, through post-processing of output data so that, for example, a hypothetical correlation is created in order to examine a specific type of anomaly.

To serve as an example, a hypothetical extreme case scenario, as shown in Figure 20, can be created simply by sorting the X and Y outputs by size, and then shuffling one of them at the middle of the vector.

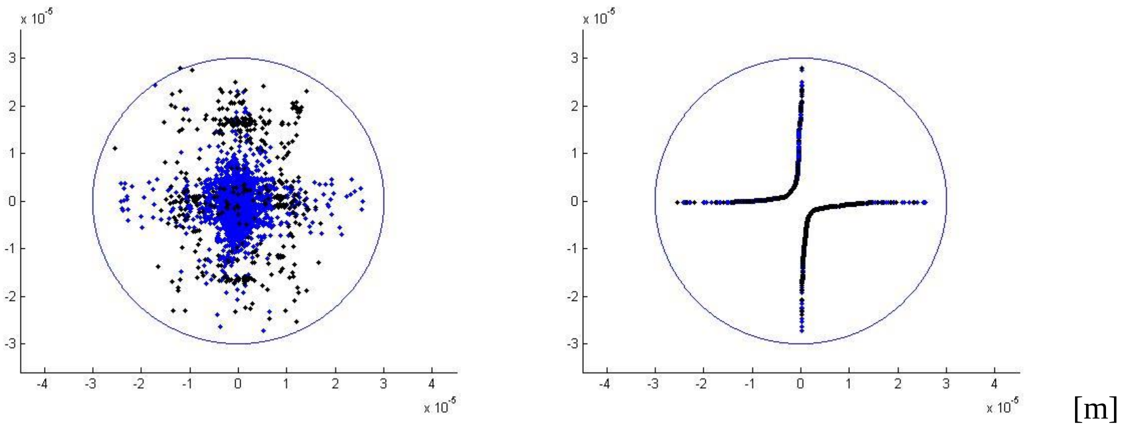


Figure 20: Output error data, plotted before and after the sort/shuffle operation, to illustrate a theoretical extreme scenario. Note that all individual X and Y values are the same in both cases.

For c=1/6, the example to the left has $c_g = 1.27$ and the one to the right has $c_g = 1.13$.

For c=1/36, the example to the left has $c_g = 0.92$ and the one to the right has $c_g = 0.00378$.

Both of these render exactly the same result in traditional process capability analysis (c_{pk}=1.96 for X, and c_{pk}=1.45 for Y), and this thought experiment highlights the suggested method's functionality.

4.1.3. Mass constant test with variation in both X and Y directions

A described characteristic of the suggested method was its suitability to evaluate multivariate process output in a way that makes it easy to react to possible issues that univariate process analysis can't.

As there was no interaction between the two sub-models for each axis, the tests with the current model was sufficient but still left some things to be desired. Therefore, as explained earlier, a hypothetical extreme case scenario was constructed using the same individual position data as found in one of the test runs. The results were as expected; the anomalous correlation created by the sort/shuffle operations caused the score to be lower, which shows that the method is able to produce a reasonable assessment for two outputs whose minima aren't at the same place.

Analysis of another test case (for graphs see Figure 21 and Figure 22) shows that the suggested method can generate reliable results when the accuracy varies in both directions. Here, the mass of the nozzle has been tested with different values. The mass of the Y-axis beam, taken from a version of the previous MY500 thesis project's files, was apparently slightly lower than optimal.

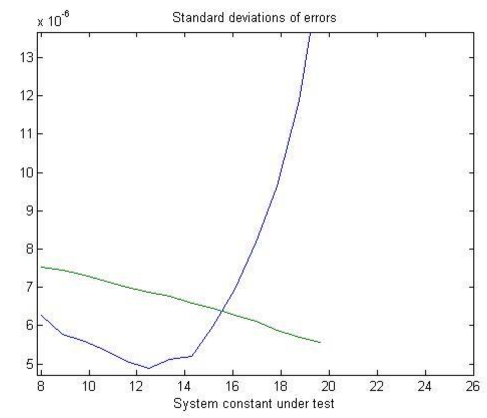


Figure 21: Standard deviations of X and Y errors. The X axis is the graph with a local minimum, while the Y axis standard deviation decreases only slightly throughout the tests.

Since the mass of the nozzle affects both X and Y axis actuation, changing it causes the standard deviations of the errors to vary in both directions.

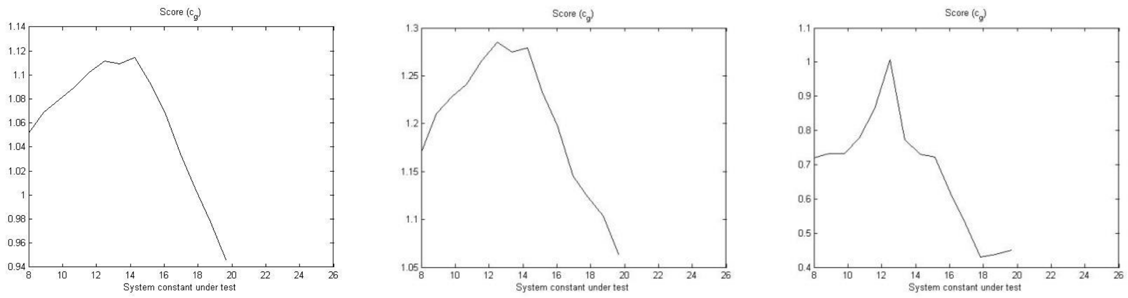


Figure 22: Three different score graphs for the test, with c=1/3, c=1/6, and c=1/36.

Here, it can be seen that the c_g index is able to assess the performance when both output parameters vary, and that it can do so for various sensitivities. The X-axis precision is more significantly affected by the parameter shift. Note how the calculation with the lowest sensitivity (c=1/3) results in a lower peak than the middle one (c=1/6). This is due to the scaling explained earlier, where the index was chosen to converge with c_p at c_p =1, with no mean error. Therefore, the maximum possible score approaches 1 as sensitivity is lowered infinitely.

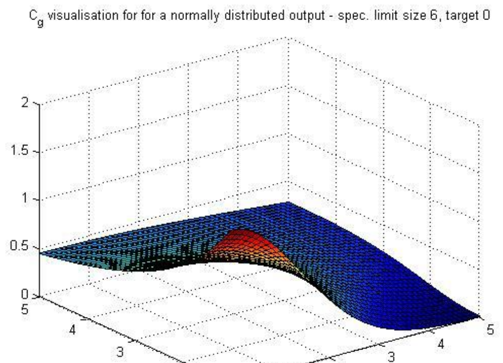


Figure 23: Visualization (equivalent to Figure 14 presented earlier) of c_g with c=1/6 instead of 1/36. Note how there's somewhat less similarity with c_p at this lower sensitivity.

It's apparent that sensitivity shouldn't be lowered indefinitely (because then there wouldn't be any differences to analyze), although no deeper analysis with the goal of finding a recommended limit has been done yet. There may also be a better way to scale the index than to simply multiply it with a constant to get similarity with c_{pk}. For example, some scaling method that uses exponentiation may perhaps be better, although it wasn't examined in detail here.

5. DISCUSSION AND CONCLUSIONS

A discussion of the results and the conclusions that the authors have drawn during the Master of Science thesis are presented in this chapter. The conclusions are based from the analysis with the intention to answer the formulation of questions that is presented in Chapter 1.

5.1 Discussion

In the beginning, there were no delimitations and no set of requirements about the method that was to be suggested and implemented in the testing platform for the purpose of evaluating the robustness of the modeled positioning system. Therefore, the basic principle for the analysis method, as well as the method itself, had to be decided during the project. As explained, the main inspiration was process capability analysis, and the efforts that have been made to make an equivalent multivariate method. Some notable aspects regarding this, is that the suggested indices in this project were of much lower complexity than the ones encountered during the research phase, and that it wasn't made with any prior experience in this field.

As with the previous MY500 jet printer thesis project, much effort was spent on issues relating to the communication between the CMOT3 control unit and the target computer that ran the system model simulation. The success of producing results was dependent on the various workarounds that were made to compensate for the fact that full communication speed couldn't be used. A goal of this project was a HIL simulation platform, which should operate in real-time. However, due to the modifications that were done, there is some room for interpretation regarding whether the platform should be seen as a true HIL test or not. Since the modifications slowed down the CMOT3 control cycle, while still allowing it to behave as if it was running on normal speed, it could be considered "as close as it gets" to a true HIL test, given the difficult technical circumstances and limitations involved with the equipment.

In one of the e-mails received by Speedgoat, after the voltage level problem had been identified, it was suggested that the IO313 could be replaced through the purchase of an IO331 card, which is made specifically to handle low voltages. Also, in a very late stage of this project, Speedgoat suggested modifications of the IO313 card's SPI system that would mitigate some of the delays. Due to the timescales involved in making decisions about whether resources should be used on these alternatives and bringing them to a functional state, only the originally available equipment was used throughout the project.

5.2 Conclusions

5.2.1. Research questions

To begin this part, a re-cap of the research questions is necessary:

1. *Can the process capability ratios c_p and c_{pk} be used to assess the performance and robustness of a planar positioning system with an elliptic specification limit?*

During the research phase, a study (referenced to in section 2.1.3) was found where PCIs were used successfully to analyze the robustness of a process that produced hydroformed aluminum components. However, since the plant in this project has a bivariate process output and traditional PCIs are univariate, the main consequence of using them would mean that the specification limit has a rectangular shape. Another aspect here is that correlations in the errors would be ignored. Therefore, it can be stated that the usage of traditional PCIs would provide an inadequate summary of the process output.

2. *Can the limitations of using c_{pk} for the robustness and performance analysis of a planar positioning system with an elliptic boundary region and a not fully normal output be overcome by introducing an alternative index that uses Gaussian functions to score the error data, and produces the same results as c_{pk} in the one-dimensional ideal case where the output is normal?*

The suggested index c_g (and thus also its related variant c_{gk}) showed clear indications on that it was possible to evaluate robustness and performance by using an index that's modeled after the standard process capability indices, which also can handle the situation where is non-normal and the boundary region is elliptic, and respond to it properly.

Also note that even though this project's tolerance region was circular from the beginning, the method is compatible with the more general elliptic case. A circle is a special case of an ellipse, and the scaling of the tolerance ranges results in a normalized circular tolerance region no matter if the original one was elliptic or not.

5.2.2. Performance evaluation and uncertainty

Some stochastic variation could be observed in the test sequences, which makes it apparent that this testing platform can only be used to get a somewhat approximate idea of where the optimum of a certain parameter is, and an estimate of how much it can vary, rather than exact answers.

Further, there are some robustness-related aspects of the system (however, these are a bit outside the project scope) which aren't tested by the platform that was developed. A theoretical scenario could be if a faulty or poorly designed system repeatedly sends requests to the CMOT3 controller, making it waste computation time and skip an important task. Performing a complete and reliable HIL test that localizes all possible failures of this type would be a near-impossible endeavor, due to the large amount of processes and commands. This project is based on the assumption that commands are properly transmitted and received; and repeatable results were obtainable at the final stages of the project, using the test script that was developed.

5.2.3. Usage of the test platform

Even if there are a few unsolved issues remaining, the work in this project has resulted in the first test platform capable of producing repeatable results from interaction between the CMOT3 control unit and a simulated model of the positioning system in the MY500 machine.

Whether the platform should be used at all given its current state, or if further improvements have to be made no matter what, remains to be discussed by those that work with developing the features of the MY500 printer and other related machines in the Mycronic product family.

The Simulink model could be extended with a more detailed voltage-to-current model that's closer to how the real actuators behave. Currently, it treats this aspect in a very simple way – the reference current is fed to the actuator model directly. Also, it might be desirable to extend it so that it's more similar to the SimMechanics model, but in a way that doesn't slow down the calculation time so greatly.

5.2.4. Verification

As for the method, it can be considered adequately verified to be suitable for the task it was made for. It was also shown to be able to handle an artificial extreme-case scenario that the traditional process capability indices don't react to in any manner, and did it in a proper way.

In the framework of this project, there was no opportunity to do extensive real-world comparisons of entire sets of simulations with one or two varying system parameters. The main reason is that it would be almost impossible and very time-consuming to, for example, find a fitting set of motors with different motor constants, and dismantle one of the jet printers multiple times to try them all.

However, there are still some things that can be mentioned in this section.

The simplified Simulink model is based on a high-order model that was done by with the more resource-demanding extension package SimMechanics. As explained in the previous master's two theses, the SimMechanics system was deemed to model the real-world system well, and the later model, which used only basic Simulink features, was deemed to be a reasonably acceptable approximation of the former. Tests of system variables that are done in the vicinity of the nominal ones can therefore be seen as a very good hint of the behavior of the real-world system; however it might be better to use tests of a real machine in more critical contexts.

A test run with a real MY500 printer was also made to compare its output with the general appearance and magnitude of a simulation that used the same template file. While the positional error data for the dot placement didn't have identical appearance as in the equivalent simulation, they were similar, and there magnitudes of the errors were generally close to each other. For confidentiality reasons, plots and detailed error data from a real MY500 machine cannot be included in this report.

6. RECOMMENDATIONS AND FUTURE WORK

In this chapter, recommendations and suggestions for future related work are presented.

6.1 Recommendations

First of all, the issue with the communication overhead times on the IO313 unit needs to be dealt with, so that the simulation can be done properly without running it at half-speed and having various compensations and workarounds to mitigate the effects from the changed cycle time.

Another thing that might be interesting to try is to adapt the test script so that it sends commands to a real MY500 unit (preferably with an empty "dummy" paste cartridge), and compare its feasibility for gathering data sets that can be analyzed quickly and with much more flexibility in the Matlab environment.

Regarding the suggested performance index c_g and the way it has been used in this thesis, it's important to understand that it is in no way a replacement for process capability analysis. It was created as a tool to overview how the performance of the (simulated) MY500 system varies as the system parameters are changed.

Any usage of c_g in a more significant context should be preceded by studies of how it behaves in different contexts (especially with different sensitivities). The choice to scale c_g so that it would be equal to c_{pk} in a specific case could arguably be seen as relatively arbitrary and mostly aesthetic, and a different approach would perhaps be useful.

As for the later modification to c_g, designated c_{gk}, the same recommendations should apply.

6.2 Future work

6.2.1. Theoretical work

The suggested evaluation method has several opportunities for further research.

So far, it has only been tested with two output variables at most. Both the requirements and the outputs haven't involved any correlations (save from a theoretical example to illustrate an advantage with the new evaluation method).

Therefore, future points of interests may include:

Correlations in the output

Here, one could examine the abilities for early detection of anomalies, such as those caused by different mechanical failure modes, that make the output drift in different ways. Comparisons could be done with other evaluation methods to examine which ones are optimal for detecting the different types of error modes that may occur within a given production process.

Correlations in the tolerance area

Another interesting perspective would be the case where a correlation in the tolerance area actually is desired – for example in the manufacture of a folding product where it's important that certain parts have dimensions that fit with each other.

6.2.2. Practical issues

There are some technical problems that remain at the finalization of the project.

Half-speed simulation issue

Due to the multiple technical issues that were presented earlier, both the communication and the simulation had to be downgraded to half their normal speed and the system had to be tweaked so that equivalent results would be produced despite this modification.

An obvious goal of a future project dealing with this platform would be to resolve this issue and use a 200 µs control cycle with normal timing.

SPI buffer access delays

The SPI buffer access delays create a significant drop in precision. If they were shorter, or if it was possible to begin retrieving data from the buffer without interfering with other calculations, the precision of the test platform would be significantly improved.

IRQ problems

Another problem is that there were glitches associated with using interrupts at received messages, which sometimes made the simulations fail unexpectedly. Therefore, interrupt triggered model steps weren't used. As the asynchronous model stepping is a source of uncertainty

Data log from acc_eval.pl

As mentioned earlier in this report, the position logs from `acc_eval.pl` were faulty, and an improvised solution had to be implemented by using saved trajectory data from the Speedgoat and interpolating the coordinates with the help of timestamps (which were recorded on both sides without problems).

If it's not feasible to find a solution where the modifications of the CMOT3 can be undone to run everything at the nominal speed, then it would be good to find the cause of this problem, and see if similar complications have arisen on other places, for example with code that handles other actuators.

TTL high logic level inconsistency

The fact that the Speedgoat IO313 unit was operated slightly outside of its specifications is likely the cause behind several of the presented problems. The datasheet stated that 3.8 V is the lowest voltage for communicating a high logic level, while the CMOT3 unit used 3.3 V logic. The reason why SPI communication worked despite this may have been the fact that the transmitted bits at 5 MHz last twice as long, and thus have longer time to overcome the input capacitance of the IO313.

An alternative to purchasing new LVTTL-compatible equipment could be to try implementing a line driver that converts 3.3 V levels to 5 V reliably at 10 MHz. If the voltage level issue had been discovered in an earlier stage of the project, there would have been time to find and purchase a suitable device to examine if it could be the solution to the communications-related problems.

No activation of the actuators z and fi

This is not an issue that affected the quality of the tests in this particular case, but there are still some things that are worth mentioning. Since the operations associated with calculating the appropriate reference currents and sending them to their respective recipient take time, a more extensive HIL test would take this into consideration. Therefore, a future project may need to extend the platform so that the calculations for the actuators denoted as z and fi would be active during the tests, even if they aren't directly influencing the test results in those particular cases. For example, finding a problem early on with deadline overruns during the control cycle may be valuable, since resources can be directed earlier to address that problem and move on with the development.

6.2.3. Issues mentioned in the previous thesis project

Some remaining issues were mentioned in the previous MY500 thesis project. These are presented and discussed here.

Performance

Ágústsson mentioned that the platform was not stable at the finalization stage of his project, due to the sporadic failures that caused the CMOT3 controller to enter different error modes and halt its operation. That problem can be considered partially solved, because the current platform has been observed to run simulations reliably as long as some care is taken to ensure that some of its limitations don’t interfere, such as the problem with the narrower axis range.

Problems with Y axis

It is now possible to run Y axis simulations, and also with both the X and Y motors active simultaneously.

Change Simulink model so it’s easier to change parameters

The parameters are now defined in a struct, which the simulation script updates on each test run before the Simulink file is re-compiled prior to its upload to the Speedgoat computer. Ágústsson mentioned however more specific aspects of the system, such as direction-dependent friction constants, sensor noise, etc. The only modification on such a fundamental level was the collision model extension, although other things may easily be added if it’s desirable to test them. If less time had been spent on technical problems, a modification as suggested by Ágústsson may have been attempted. However, since the project scope was the testing platform itself, model-related details had a relatively low priority, and it was undesirable to let time be spent on, for example, CMOT3 error modes that relate to simulated sensor noise, etc.

7. REFERENCES

Ágústsson, E. (2013). *Hardware-in-the-loop Simulation of a High Speed Servo System.* Stockholm: KTH Industrial Engineering and Management.

Bertsekas, D. P., & Tsitsiklis, J. N. (2002). *Introduction to Probability, first edition.* Massachusetts: Athena Scientific, MIT.

Chiang, R. Y., & Safonov, M. G. (1998). *Robust Control Toolbox.* Natick: Mathworks.

Fathy, H. K., Filipi, Z. S., Hagena, J., & Stein, J. L. (2006). Review of Hardware-in-the-Loop Simulation and Its Prospects in the Automotive Area. *Modeling and Simulation for Military Applications*.

Kaya, İ., & Kahraman, C. (2010). A new perspective on fuzzy process capability indices: Robustness. *Expert Systems with Applications 37*, 4593–4600.

Koç, M., Agcayazi, A., & Carsley, J. (2011). An Experimental Study on Robustness and Process Capability of the Warm Hydroforming Process. *Journal of Manufacturing Science and Engineering*.

Ledin, J. A. (1999). Hardware-in-the-loop simulation. *Embedded Systems Programming*, 42-62.

Mathworks. (2003). *xPC Target For Use with Real-Time Workshop.*

Mathworks Training Services. (2013). *Fundamentals of Code Generation for Real-Time Design and Testing.* Mathworks, Inc.

Montgomery, D. C. (2009). *Introduction to Statistical Quality Control.* Arizona: John Wiley & Sons, Inc.

Motorola. (2003). *SPI Block Guide V03.06.*

Mydata. (2010). MY500 Jet Printer Specification. Täby.

Pan, J.-N., & Lee, C.-Y. (2009). New Capability Indices for Evaluating the Performance of Multivariate Manufacturing Processes. *Wiley InterScience, Quality and Reliability Engineering International*.

Rundgren, J. (2011). *Modelling, Control and Simulation of a Jet Printer Robot.* Stockholm: KTH Electrical Engineering.

Skogestad, S., & Postlethwaite, I. (2001). *Multivariable Feedback Control - Analysis and Design.* New York: John Wiley & Sons, Inc.

Speedgoat GmbH. (n.d.). *speedgoat.ch*. Retrieved August 27, 2014, from IO313/Datasheet: http://www.speedgoat.ch/Products/IOmodules-ConfigurableFPGAs-IO313.aspx

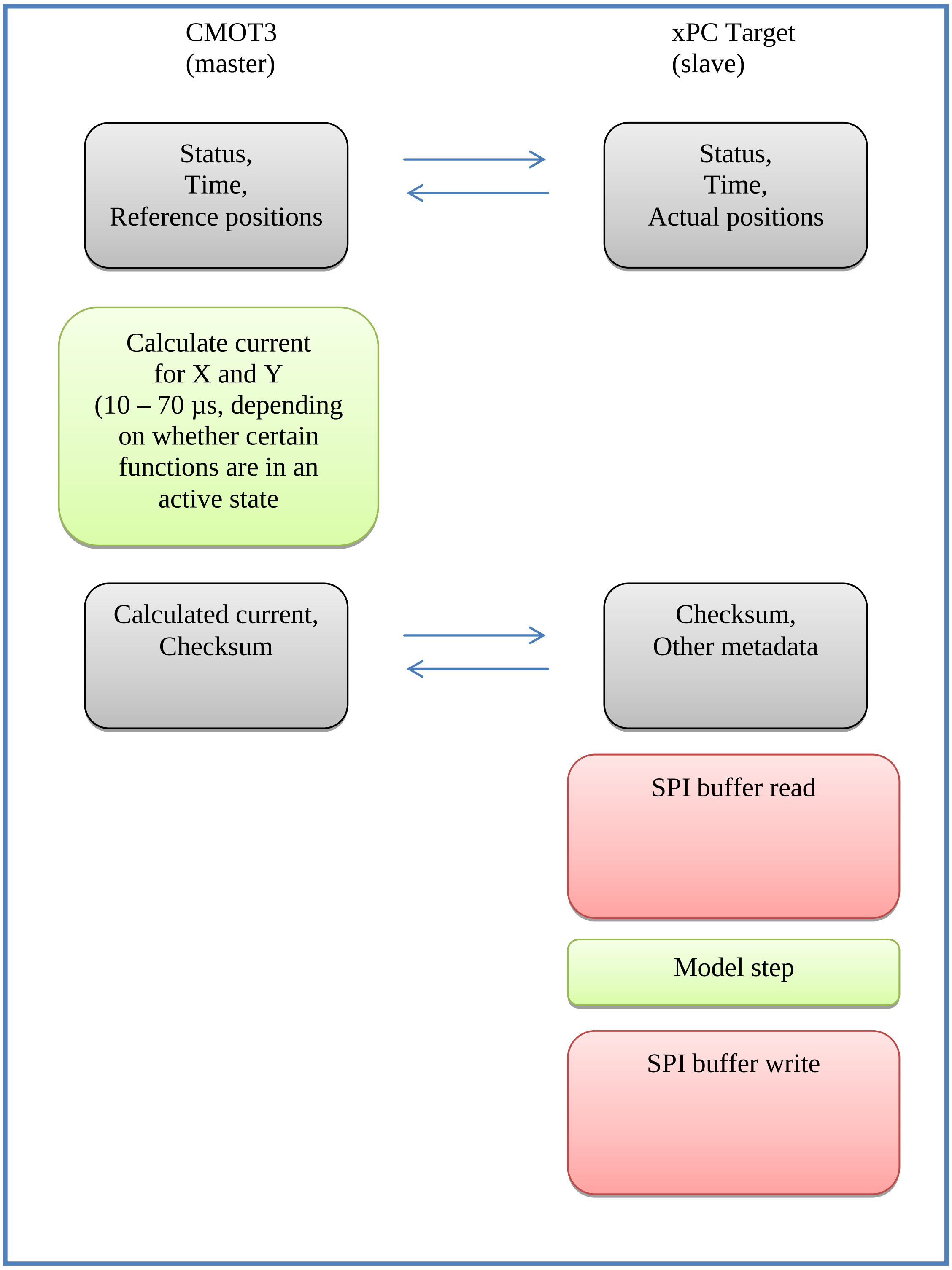
Tano, I. (2012). *Contributions to Multivariate Process Capability Indices.* Luleå: Luleå University of Technology.

Zhou, K. (1995). *Robust and Optimal Control.* New Jersey: Prentice Hall.

Åström, K. J., & Murray, R. M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers.* Princeton: Princeton University Press.

APPENDIX A: SPI COMMUNICATION CYCLE

This chart provides an overview of how the CMOT3 and the xPC Target Speedgoat were planned to communicate and operate during a cycle. The difference between this chart and the actual platform is that the "Model step" part was run several times per control cycle, in an asynchronous manner, due to glitches that were encountered when the IO313 interrupt function was used together with the CMOT3.


CMOT3
(master)
xPC Target
(slave)
Status, Time, Reference positions
Status, Time, Actual positions
Calculate current for X and Y (10 – 70 µs, depending on whether certain functions are in an active state
Calculated current, Checksum
Checksum, Other metadata
SPI buffer read
Model step
SPI buffer write

APPENDIX B: EVALUATION SCRIPT EXAMPLE VIEW

```
List of test results:

(1) : HILtest_20140527T123640
(2) : HILtest_20140527T124239
(3) : HILtest_20140527T124630
(4) : HILtest_20140527T130419
(5) : HILtest_20140527T151055
(6) : HILtest_20140528T150056
(7) : HILtest_20140602T101136
(8) : HILtest_20140602T101207
(9) : HILtest_20140602T110624
(10) : HILtest_20140602T131556
(11) : HILtest_20140603T144708
(12) : HILtest_20140604T112155

Choose directory to examine: 5

 ########

Test datestamp: 2014-05-27 15:10:55

Settings:
 Specification limit X : ±30μm
 Specification limit Y : ±30μm

Test type: 1 variable (Mc_x).

Nominal data:
        m_beam: 19.8630
       m_wagon: 12.9000
            dv: 2.0000e-04
     x_max_pos: 1.4500
     x_min_pos: 0.9500
     y_max_pos: 1.4500
     y_min_pos: 0.9500
          Mc_x: 15.6000
          Mc_y: 54.2000
          Ff_x: 1.8953
          Ff_y: 13.9254
          df_x: 1.3170
          df_y: 8.3445
    x_init_pos: 1.3000
    y_init_pos: 1.3000


Processing 70 simulation logs. Please wait...
```

APPENDIX C: SIMULINK CODE BLOCK CONTENTS

```
function [Type,CKTime,RefPosX,RefPosY,RefCurX,RefCurY,ChkSumSent,ChkSumRecv]
= data_in(inbytes)
bytes=uint8(fliplr(inbytes'));
Type        = uint8(0);
CKTime      = typecast(bytes(1:4),'int32');
RefPosX     = typecast(bytes(5:8),'single');
RefPosY     = typecast(bytes(9:12),'single');
RefCurX     = typecast(bytes(13:16),'single');
RefCurY     = typecast(bytes(17:20),'single');
ChkSumSent  = uint8(0);
ChkSumRecv  = uint8(0);
end

function outbytes = data_out(State,SimTime,CurPosX,CurPosY,ChkSumSent)
bytes=uint8(zeros(1,20));
% bytes(1)      = uint8(State);
bytes(1:4)    = typecast(int32(SimTime),'uint8');
bytes(5:8)    = typecast(single(CurPosX),'uint8');
bytes(9:12)  = typecast(single(CurPosY),'uint8');
bytes(13:16)  = uint8([0 0 0 0]);
bytes(17:20)  = uint8([0 0 0 0]);
% bytes(22)     = uint8(ChkSumSent);
% bytes(23)     = uint8(0);
end

% Collision/reset block, found inside "Plant" block
function [x_init_pos, y_init_pos, x_reset, y_reset] = reset_pos(start,
x_init, y_init, x_pos, y_pos, limits)
x_min=limits(1,1);
x_max=limits(1,2);
y_min=limits(2,1);
y_max=limits(2,2);
bounce=0.0001;
if (start==uint8(1))
    x_init_pos=double(x_init);
    y_init_pos=double(y_init);
    x_reset=1;
    y_reset=1;
else
    x_init_pos=double(x_pos);
    y_init_pos=double(y_pos);
    if x_pos>x_max
        x_init_pos=x_max-bounce;
        x_reset=1;
    elseif x_pos<x_min
        x_init_pos=x_min+bounce;
        x_reset=1;
    else
        x_reset=0;
    end
    if y_pos>y_max
        y_init_pos=y_max-bounce;
        y_reset=1;
    elseif y_pos<y_min
        y_init_pos=y_min+bounce;
        y_reset=1;
    else
        y_reset=0;
    end
end
```

```
end
```

APPENDIX D: CMOT3 SSH COMMAND MACRO SCRIPT (FOR USE TOGETHER WITH PLINK.EXE)

```
% Log-in data for servo computer
host     = '[redacted]';
username = '[redacted]';
password = '[redacted]';
login = ['plink ' host ' -l ' username ' -pw ' password ' '];

 % Allow external commands
msg_head = 'TERM=xterm; export TERM; source .profile; export
IS_LOCAL_TTY=yes; sleep 0;';
msg_end  = ' ; sleep 0;';

 % Syntax: send('string') for sending commands directly to the CMOT3 host
computer
send   = @(msg) system([login, msg_head, msg, msg_end]);

% List of macros for sending commands to the CMOT3 unit
dispcmd    = @(msg) []*fprintf(['\n Sending command to ' msg '...\n\n']);
%null-return macro for displaying when another command is made
loadmot    = @() send([dispcmd('run loadmot') 'loadmot -vf --canxy']);
initcom    = @() send('q_ setpar canxy 12 37 1'); % Communication states
stopcom    = @() send('q_ setpar canxy 12 37 0');
initctrl   = @() send('q_ setpar canxy 12 38 1'); % Controller states
stopctrl   = @() send('q_ setpar canxy 12 38 0');
msetdynx   = @() send([dispcmd('start X axis') 'q_ msetdyn x 1']);   % arg 1
renders short variant of msetdyn
msetdyny   = @() send([dispcmd('start Y axis') 'q_ msetdyn y 1']);
movx       = @(pos) send(['q_ mov x '
num2str((pos>0)*((pos<25)*pos*1000000+(pos>250000)*pos),'%.0f')]);     %
Accepts both m and µm
movy       = @(pos) send(['q_ mov y '
num2str((pos>0)*((pos<25)*pos*1000000+(pos>250000)*pos),'%.0f')]);
readposx   = @() send('q_ readpos x');
readposy   = @() send('q_ readpos y');
parsemsg   = @(msg) str2num(msg(find([' ',msg]==' ',1,'last'):end));
status     = @(num) send(['status ' num2str(num)]);
servossr   = @() send('servossr');
%wsu = @() send('wsu my500thesis@1917alevuj@fermenta xy3.mcf');
sendHWIcmds = @(HWI_dir,HWI_file) send([dispcmd('run sendHWIcmds')
'sendHWIcmds -a xy -f ' HWI_dir HWI_file]);
getHWIdata  = @(HWI_dir,HWI_file) send(['cat ' HWI_dir HWI_file]);
acceval     = @() send([dispcmd('run acc_eval') 'acc_eval.pl']);
getdots     = @() send('cat lib/mot1/tinytools/tmp/canxy.dcl');
HWIdir      = @() send(['dir ' HWI_dir]);
```

APPENDIX E: USER_RUN.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  Main script for HIL test (CMOT3/xPC) %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Clear workspace
clc;clear all;hold off;close all;

 % Create 'Results' directory.
if ~isdir('Results'), mkdir('Results'); end

 % Flag for identification of completed tests
completed=0;    % Changes to 1 when the whole script is completed

 % System model and SPI data handling
simulink_filename = 'main';

 % Text file with system constants
sysconstants_filename = 'constants.txt';

 % HWI job for testing
HWI_dir   = '/home/tpsys/lib/mot1/tinytools/sendHWIcmdsJobs/';
HWI_file  = 'servoMovTPLog_Demo20_Senju_LF.txt';

 % Macros for sending commands to CMOT3
 % Note that the UNIX function return arrives as the second argument, [~,msg]
= foo()
run loadmacros;

% Scan the text file with system parameters
datascan=init_system_parameters(sysconstants_filename);
if ~(numel(datascan)==2 || size(datascan{1,1})==size(datascan{1,2}))
    error('Error in system constants file.')
end

disp(' ')
disp(' ')
disp('HIL tester for CMOT3')
disp(['Model parameters loaded from ' sysconstants_filename])
disp(' ')
header = ''; %header = input('Enter file header (blank input renders
"HILtest"): ','s');
if isempty(header)
    header='HILtest';
end

disp(' ######## ')
disp(' ')

data=[];
varnames={};
for index=1:length(datascan{1,1})
    eval(['data.' cell2mat(datascan{1,1}(index)) ' = '
num2str(datascan{1,2}(index),'%1.20e') ';']);
    varnames{index}=cell2mat(datascan{1,1}(index)); %#ok<SAGROW>
    disp(['(' num2str(index) ') ' cell2mat(datascan{1,1}(index)) ])
    disp(['Value: ' num2str(datascan{1,2}(index)) ]);
    disp(' ')
end
```

```
disp(' ######## ')
disp(' ')
disp('0: Nominal system test')
disp('1: Test one variable')
disp('2: Test two variables')
disp(' ')
testvars=input('Select test mode: ');
disp(' ')
if ~(numel(testvars)==1) || ~(testvars==0 || testvars==1 || testvars==2)
    error('Invalid input.')
end

if testvars==0
    maxtests=1;
else
    switch testvars
        case 1
            var1=input('Select variable 1 (by index number): ');
            var2=1; % Dummy assign, needed
        case 2
            var1=input('Select variable 1 (by index number): ');
            var2=input('Select variable 2                  : ');
            if var1==var2
                error('Index error (duplicate selection).')
            end
    end

    vars=[var1 var2];

    disp(' ')
    if ~min(sign(vars))==1 || ~min(round(vars)==vars)==1  ||
max(vars)>length(datascan{1,1})
        error('Index error.')
    end

    disp([' Variable 1: ' cell2mat(datascan{1,1}(var1)) ' (Nominal value: '
num2str(datascan{1,2}(var1)) ')'']);
    min1=input('  Min value : ');
    max1=input('  Max value : ');
    inc1=input(' Increments : ');

    min2=1;
    max2=1;
    inc2=1; % Dummy assign, needed

    if testvars==2
        disp(' ')
        disp([' Variable 2: ' cell2mat(datascan{1,1}(var2)) ' (Nominal value:
' num2str(datascan{1,2}(var2)) ')'']);
        min2=input('  Min value : ');
        max2=input('  Max value : ');
        inc2=input(' Increments : ');
    end

    incs=[inc1 inc2];

    if min1>max1 || min2>max2
        error('Error (max<min).')
    elseif ~min(sign(incs))==1 || ~min(round(incs)==incs)==1
        error('Invalid amount of increments.')
    end
    maxtests=prod([inc1 inc2]);
    alltests=[floor((0:(maxtests-1))/inc2)' , mod((0:(maxtests-1)),inc2)']+1;

    values1=linspace(min1,max1,inc1);
    values2=linspace(min2,max2,inc2);
end

datestamp=datestr(now,30);
location=['Results/' header '_' datestamp '/'];
mkdir(location);

filename_lead=[header '_' datestamp];
filename_meta=[filename_lead '_meta'];

save([location
filename_meta],'completed','datestamp','data','varnames','testvars','HWI_dir'
,'HWI_file');

if testvars>0
    save([location filename_meta],'var1','var2','values1','values2','-
append');
end

 % [~,HWIdata]=getHWIdata(HWI_dir,HWI_file);

index=0;
disp(' ');disp(' ');
while (index<maxtests)
    index=index+1;
    if maxtests>1
        disp(['(Test run ' num2str(index) ' of ' num2str(maxtests) ')'])
    end

    if testvars>0
        eval(['data.' cell2mat(datascan{1,1}(var1)) ' = '
num2str(values1(alltests(index,1))) ';']);
        if testvars==2
            eval(['data.' cell2mat(datascan{1,1}(var2)) ' = '
num2str(values2(alltests(index,2))) ';']);
        end
    end

    % Set the CMOT3 to a known state by using loadmot
    [loadmot_msg1,loadmot_msg2]=loadmot();
    if ~loadmot_msg1==0
        error('Error when contacting CMOT3 to run loadmot.');
    elseif isempty(regexp(loadmot_msg2,'Loaded', 'once'))
        error('"Loaded" not found in return message from loadmot.');
    else
        disp('Compiling xPC Target program...')
    end

    % Compile program for xPC Target
    rtwbuild(simulink_filename);

     %%%%% Start xPC Target unit %%%%%
    +slrt; tic;

    if maxtests>1
        disp(['Test run ' num2str(index) ' of ' num2str(maxtests) '.'])
    end

    % Initialize SPI communication and control of external system
    [~,~]=initcom();
    [~,~]=initctrl();

    % Initialize X axis
    [~,msg_x] = msetdynx(); toc
    %if ~parsemsg(msg_x)==0
    %    -slrt;
    %    error(['Failed msetdyn x. Error: ' num2str(parsemsg(msg_x)) ]);
    %end

    % Initialize Y axis
    [~,msg_y] = msetdyny(); toc
    %if ~parsemsg(msg_y)==0
    %    -slrt;
    %    error(['Failed msetdyn y. Error: ' num2str(parsemsg(msg_y)) ]);
    %end

    if parsemsg(msg_x)==0 && parsemsg(msg_y)==0
        % Move printer head to (1300000,1300000)
        [~,~]=movx(1.3);
        [~,~]=movy(1.3);

        % Start print job
         [~,msgHWIcmds]=sendHWIcmds(HWI_dir,HWI_file); toc

        %%%%% Stop xPC Target unit  %%%%%
        -slrt;

        % Get acc_eval log
        [~,~]=acceval();
        [~,dotlog]=getdots();
        dotdata=strread(dotlog(2+regexp(dotlog,'-\n'):end));
        dotdata=dotdata(1:find(dotdata(:,1)>0,1,'last'),:);

        % Generate filename and save simulation data
        if testvars==0
            filename=[filename_lead '_0'];
        else
            filename=[filename_lead '_' num2str(alltests(index,1)) '_'
num2str(alltests(index,2))];
        end
        slrtdata=get(slrt);
        save([location filename],'slrtdata','dotdata') % + messages, etc.
        toc
        disp(' ')
        disp(' #### Simulation data saved. #### ')
        disp(' ')
    else
        -slrt;
        disp(' ')
        disp([' Initialization for test run ' num2str(index) ' of '
num2str(maxtests) ' failed.'])
        disp(' ')
    end
end

 % Register that the test has been completed
completed=1;
save([location filename_meta],'completed','-append');
```

```
disp('Test complete.')
disp(' ')
```

APPENDIX F: ANALYZE.M

```
clc;clear all;close all;
run loadmacros;

xpctargetping;

x_speclimit = 30e-6;        % Specification limits X and Y
y_speclimit = 30e-6;

%% Folder presentation %%
location_info=ls('Results');
location_info=location_info(3:end,:);

disp(' ')
disp(' ')
disp('List of test results:')
disp(' ')
for ind=1:size(location_info,1)  %
    disp(['(' num2str(ind) ') : ' location_info(ind,:)]);
end

disp(' ')
if size(location_info,1)==0
    error('Folder is empty.')
end

dir_ind=input('Choose directory to examine: ');
disp(' ')
if isempty(dir_ind)
    dir_ind=size(location_info,1);
    disp('Blank input - latest test result selected.')
    disp(' ')
elseif ~sum(dir_ind==(1:size(location_info,1)))
    error('Invalid input.')
end
disp(' ######## ')
disp(' ')

%% Read test result metadata %%

testid=location_info(dir_ind,1:max(regexp(location_info(dir_ind,:),'\w[\s]*')
));
location=['Results/' testid '/'];
filename_meta=[testid '_meta'];

warning off all
load([location filename_meta],'completed');
warning on
if ~completed
    error('The selected test was not run to completion.')
end
warning off all
load([location filename_meta]);
warning on

disp(['Test datestamp: ' datestr(datenum(datestamp,'yyyymmddTHHMMSS'),31)])
disp(' ')
disp('Settings:')
disp([' Specification limit X : ±' num2str(x_speclimit*1e6) 'µm'])
disp([' Specification limit Y : ±' num2str(y_speclimit*1e6) 'µm'])
disp(' ')

%% Conditions for the three different test categories %%

if testvars==0

    disp('Test type: nominal.')
    disp(' ')
    disp('Nominal data:')
    disp(data)
    disp(' ')
    warning off all
    load([location testid '_0']);
    warning on

    dot_time_log=dotdata(:,6)/100;
    sim_index=  (1+find(diff(slrtdata.OutputLog(:,5))<-
1,1,'first')):length(slrtdata.TimeLog);
    CKtime_log      = slrtdata.OutputLog(sim_index,5);
    trajectory_log  = slrtdata.OutputLog(sim_index,1:2);
    [~,unique_index,~] = unique(CKtime_log,'first');

dot_places=interp1(CKtime_log(unique_index),trajectory_log(unique_index,:),-
16+dot_time_log);        %-16

    start_index=find(CKtime_log<dot_time_log(1),1,'last')-100;

    firstdots_index=find(dotdata(:,5)==1);
    first10dots_index=find(dotdata(:,5)<=10);

    firstdotdata=dotdata((dotdata(:,5)==1),:);
    firstdot_places=dot_places((dotdata(:,5)==1),:);
    first10dotsdata=dotdata((dotdata(:,5)<=10),:);
    first10dots_places=dot_places((dotdata(:,5)<=10),:);

    figure;hold on;
    plot(dotdata(:,1),dotdata(:,3),'bx')

plot(trajectory_log(start_index:end,1),trajectory_log(start_index:end,2),'k-
')
    plot(dot_places(:,1),dot_places(:,2),'r.')
    plot(firstdot_places(:,1),firstdot_places(:,2),'k.')

    hold off;

    %%%%

    xdiff=dotdata(:,1)-dot_places(:,1);
    ydiff=dotdata(:,3)-dot_places(:,2);

            %xdiff = sort(xdiff);ydiff = sort(ydiff);ydiff =
[ydiff((1+length(ydiff)/2):length(ydiff));ydiff(1:length(ydiff)/2)];

    sigma_dist                = std([xdiff,ydiff]);
    sigma_dist_um             = sigma_dist*1e6;
    sigma_dist_firstdots      =
std([xdiff(firstdots_index),ydiff(firstdots_index)]);
    sigma_dist_um_firstdots   = sigma_dist_firstdots*1e6;
    sigma_dist_first10dots    =
std([xdiff(first10dots_index),ydiff(first10dots_index)]);
```

```
    sigma_dist_um_first10dots = sigma_dist_first10dots*1e6;

    xdiff_norm = xdiff/x_speclimit;
    ydiff_norm = ydiff/y_speclimit;
    diffdist_norm  = sqrt(xdiff_norm.^2+ydiff_norm.^2);


    score = 12.1*mean(gauss(diffdist_norm,0,1/36));  %score, narrow variant
with 1/36
    score = 2.266*mean(gauss(diffdist_norm,0,1/6));

    figure;hold on
    plot(xdiff(dotdata(:,5)>1),ydiff(dotdata(:,5)>1),'b.')
    plot(xdiff(dotdata(:,5)==1),ydiff(dotdata(:,5)==1),'k.')

    circle_x=(-x_speclimit):1e-7:x_speclimit;
    circle_y=sqrt(y_speclimit^2-circle_x.^2);
    plot(circle_x, circle_y)
    plot(circle_x,-circle_y)

    axis([-1.5*x_speclimit 1.5*x_speclimit -1.2*y_speclimit 1.2*y_speclimit])
    hold off;

    diffdist  = sqrt(xdiff.^2+ydiff.^2);
    diffangle = atan2(ydiff,xdiff);

    disp(' ')
    disp(['Standard deviation X : ' num2str(sigma_dist_um(1)) 'µm ' '(First
dots: ' num2str(sigma_dist_um_firstdots(1)) 'µm, ' 'Ten starting dots: '
num2str(sigma_dist_um_first10dots(1)) 'µm)'])
    disp(['Standard deviation Y : ' num2str(sigma_dist_um(2)) 'µm ' '(First
dots: ' num2str(sigma_dist_um_firstdots(2)) 'µm, ' 'Ten starting dots: '
num2str(sigma_dist_um_first10dots(2)) 'µm)'])
    disp(' ')
    disp(['Cp value X :  ' num2str(x_speclimit/(3*sigma_dist(1)))  ' (First
dots: ' num2str(x_speclimit/(3*sigma_dist_firstdots(1))) ', ' 'Ten starting
dots: ' num2str(x_speclimit/(3*sigma_dist_first10dots(1))) ' )'])
    disp(['Cp value Y :  ' num2str(y_speclimit/(3*sigma_dist(2)))  ' (First
dots: ' num2str(y_speclimit/(3*sigma_dist_firstdots(2))) ', ' 'Ten starting
dots: ' num2str(y_speclimit/(3*sigma_dist_first10dots(2))) ' )'])
    disp(' ')
    disp(['Cpk value X : ' num2str(abs(mean(xdiff)-
x_speclimit)/(3*sigma_dist(1))) ' (First dots: '
num2str(abs(mean(xdiff(firstdots_index))-
x_speclimit)/(3*sigma_dist_firstdots(1))) ', ' 'Ten starting dots: '
num2str(abs(mean(xdiff(first10dots_index))-
x_speclimit)/(3*sigma_dist_first10dots(1))) ' )'])
    disp(['Cpk value Y : ' num2str(abs(mean(ydiff)-
y_speclimit)/(3*sigma_dist(2))) ' (First dots: '
num2str(abs(mean(ydiff(firstdots_index))-
y_speclimit)/(3*sigma_dist_firstdots(2))) ', ' 'Ten starting dots: '
num2str(abs(mean(ydiff(first10dots_index))-
y_speclimit)/(3*sigma_dist_first10dots(2))) ' )'])
    disp(' ')
    disp(['Cg : ' num2str(score)])
    disp(' ')
    e_norm=sqrt((xdiff/x_speclimit).^2+(ydiff/y_speclimit).^2);
    score_test=sum(sqrt(1-e_norm.^2))/numel(e_norm) ;

elseif testvars==1
```

```
    score=zeros(length(values1),1);
    stddev=zeros(length(values1),2);

    dotdata_len=zeros(1,length(values1));

    disp(['Test type: 1 variable (' varnames{var1} ').'])
    disp(' ')
    disp('Nominal data:')
    disp(data)
    disp(' ')
    fprintf(['Processing ' num2str(length(values1)) ' simulation logs. Please
wait\n']);
    dotdata_len_prev=-1;

    for ind1=1:length(values1)
       try
            warning off all
            load([location testid '_' num2str(ind1) '_1']);
        catch
            dot_time_log=[];
        end
        warning on all

        % slrtdata=tgdata; % for old version

        dot_time_log=dotdata(:,6)/100;
        sim_index=  (1+find(diff(slrtdata.OutputLog(:,5))<-
1,1,'first')):length(slrtdata.TimeLog);
        CKtime_log      = slrtdata.OutputLog(sim_index,5);
        trajectory_log  = slrtdata.OutputLog(sim_index,1:2);
        [~,unique_index,~] = unique(CKtime_log,'first');

dot_places=interp1(CKtime_log(unique_index),trajectory_log(unique_index,:),-
16+dot_time_log);        %-16

        xdiff=dotdata(:,1)-dot_places(:,1);
        ydiff=dotdata(:,3)-dot_places(:,2);


        xdiff_norm = xdiff/x_speclimit;
        ydiff_norm = ydiff/y_speclimit;


        diffdist_norm  = sqrt(xdiff_norm.^2+ydiff_norm.^2);

%          figure;plot(xdiff,ydiff,'.')
%          axis([-x_speclimit*1.5 x_speclimit*1.5 -y_speclimit*1.2
y_speclimit*1.2])

        %score(ind1) = sum(sqrt(1-diffdist_norm.^2))./numel(diffdist_norm);

        score(ind1) = 12.1*mean(gauss(diffdist_norm,0,1/36));  %score, narrow
variant with 1/36
        score(ind1) = 2.266*mean(gauss(diffdist_norm,0,1/6));
        %score(ind1) = 1.42*mean(gauss(diffdist_norm,0,1/3));

        stddev(ind1,1:2) = [std(xdiff),std(ydiff)];

        % Check if the amount of dots is consistent between each run
        dotdata_len(ind1)=length(dotdata);
        if ind1>1
            if ~(dotdata_len(ind1)==dotdata_len(ind1-1))
                warning(['Inconsistent dot count ('
num2str(dotdata_len(ind1)) ' vs. ' num2str(dotdata_len(ind1-1)) ').' ])
                disp(' ')
            end
        end
        fprintf('\b.\n');
    end
    disp(' ')
    dotdata_len(~(dotdata_len==max(dotdata_len)))=NaN+NaN*1i;
    score(~(dotdata_len==max(dotdata_len)))=NaN+NaN*1i;
    %figure;plot(values1,1./(1-real(score)),'k');%

    figure;plot(values1,score,'k');%
    title('Score (c_g)')
    xlabel('System constant under test')

    figure;plot(values1,stddev);
    title('Standard deviations of errors')
    xlabel('System constant under test')

elseif testvars==2
    disp(['Test type: 2 variables (' varnames{var1} ' and ' varnames{var2}
').'])

  %%%%
    score=zeros(length(values2),length(values1));
    disp(' ')
    disp('Nominal data:')
    disp(data)
    disp(' ')
    fprintf(['Processing ' num2str(length(values2)*length(values1)) '
simulation logs. Please wait\n']);
    dotdata_len_prev=-1;
  %%%%

    % Indexing: (ind2, ind1)
    for ind1=1:length(values1)
        for ind2=1:length(values2) %#ok<ALIGN>
           try
                warning off all
                load([location testid '_' num2str(ind1) '_1']);
            catch
                dot_time_log=[];
            end
            warning on all

            dot_time_log=dotdata(:,6)/100;
            sim_index=  (1+find(diff(slrtdata.OutputLog(:,5))<-
1,1,'first')):length(slrtdata.TimeLog);
            CKtime_log      = slrtdata.OutputLog(sim_index,5);
            trajectory_log  = slrtdata.OutputLog(sim_index,1:2);
            [~,unique_index,~] = unique(CKtime_log,'first');

dot_places=interp1(CKtime_log(unique_index),trajectory_log(unique_index,:),-
16+dot_time_log);        %-16

            xdiff=dotdata(:,1)-dot_places(:,1);
            ydiff=dotdata(:,3)-dot_places(:,2);

            xdiff_norm = xdiff/x_speclimit;
            ydiff_norm = ydiff/y_speclimit;

            diffdist_norm  = sqrt(xdiff_norm.^2+ydiff_norm.^2);

%          figure;plot(xdiff,ydiff,'.')
%          axis([-x_speclimit*1.5 x_speclimit*1.5 -y_speclimit*1.2
y_speclimit*1.2])

        %score(ind2,ind1) = sum(sqrt(1-
diffdist_norm.^2))./numel(diffdist_norm);

        score(ind2,ind1) = 12.1*mean(gauss(diffdist_norm,0,1/36));  %score,
narrow variant with 1/36
        score(ind2,ind1) = 2.266*mean(gauss(diffdist_norm,0,1/6));

            % Check if the amount of dots is consistent between each run
            dotdata_len=length(dotdata);
            if ~(dotdata_len==dotdata_len_prev) && ~(dotdata_len_prev==-1)
                warning(['Inconsistent dot count (' num2str(dotdata_len) '
vs. ' num2str(dotdata_len_prev) ').' ])
                disp(' ')
            end
            dotdata_len_prev=dotdata_len;
            fprintf('\b.\n');
        end
    end

    disp(['Dot count (last file): ' num2str(dotdata_len)])
    disp(' ')
    figure;surf(values1,values2,score);
    %axis([])

end

%% - %%
```