Working model 2.0

Ortmans, R.M.H.M.

Published: 01/01/1996

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 07. Dec. 2018
Stageverslag

Begeleiding:

- Ir. A.M. Rankers
  Philips (CFT, Predictive Modeling)
- Dr. Ir. M.J.G. van de Molengraft
  TUE

R.H.M.H. Ortmans
Faculteit Werktuigbouwkunde
Technische Universiteit Eindhoven
Mei 1996
Summary

The subject of this research is the validation of Working Model. Working Model is interesting to investigate because geometric non-linearities can be modeled and a controller can be implemented in an external application. So servo-systems with geometric non-linearities can be modeled in Working Model in combination with Matlab. The validation is done by implementing an existing four beam mechanism, the ADAT.

Working Model is a two dimensional rigid multi body package. Working Model runs under Microsoft Windows and is very easy to learn. The ADAT was implemented very quickly in Working Model. Real time communication with external applications is possible. This is called dynamic data exchange. One of such external applications is Matlab. In Matlab a controller is implemented for the ADAT, which can be done very simply. Working Model has not the possibility to determine directly frequency responses. But using DDE, the time response can be transferred into a frequency response in Matlab. These frequency responses are reliable.

As mentioned the ADAT was implemented in Working Model and a controller in Matlab. The geometric non-linearities are seen in simulations. So Working Model is very suitable to model servo systems with significant geometric non-linearities.
# Table of contents

1. **Introduction**  

2. **Working Model 2.0**  
   2.1 What is Working Model ?  
   2.2 Rigid bodies  
   2.3 Constraints  
   2.4 Meters and controls  
   2.5 External application interface  
   2.6 Simulation  

3. **Dynamic Data Exchange**  
   3.1 What is DDE ?  
   3.2 External application interface  
   3.3 Sequence of DDE  

4. **Frequency response**  
   4.1 How to determine a frequency response from the time response  
   4.1.1 Frequency response based on the powerspectra of the time signal  
   4.1.2 Frequency response based on an estimation of the system matrices  
   4.2 Results  

5. **The ADAT**  
   5.1 What is the ADAT  
   5.2 Rigid body dynamics of the ADAT  
   5.3 Modeling the ADAT in Working Model  
   5.4 Frequency response of the ADAT  
   5.5 Implementing a controller  

6. **Conclusions and recommendations**  
   6.1 Conclusions  
   6.2 Recommendations  

**References**
Appendices

A  Time lag  25

B  Matlab spectrum and arx / trf routines  27
   B.1  Spectrum  27
   B.2  Arx / trf  28

C  Response comparison  30
   C.1  How comparing responses  30
   C.2  A single mass system  30
   C.3  Mass-spring-damper system  33

D  Simulation results in the frequency domain  36
   D.1  One mass system  36
   D.2  Mass-spring-damper system  40

E  Geometry and properties of the ADAT  46
   E.1  Geometry of the ADAT  46
      E.1.1  Angle of the crank for the pick and place position  46
      E.1.2  Displacement of the tip  47
   E.2  Geometry and properties of the crank  48
   E.3  Geometry and properties of the linkage  51
   E.4  Geometry and properties of the transfer arm  52

F  The ADAT in Working Model  54

G  Estimation of the inertia of the ADAT  56
   G.1  Theoretical estimation  56
      G.1.1  \( J_{\text{equivalent}} \) for the ADAT when \( \varphi = 0 \) degrees  57
      G.1.2  \( J_{\text{equivalent}} \) for the ADAT when \( \varphi = 90 \) degrees  58
   G.2  Practical estimation  59
      G.2.1  \( J_{\text{equivalent}} \) for the ADAT when \( \varphi = 0 \) degrees  59
      G.2.2  \( J_{\text{equivalent}} \) for the ADAT when \( \varphi = 90 \) degrees  60

H  Controller design  61
   H.1  Trajectory  61
      H.1.1  Design of the trajectory  61
   H.2  PID-controller  62
      H.2.1  Continuous time case  62
      H.2.2  Discrete time case  64
   H.3  Feedforward  64
   H.4  Filter  65
      H.4.1  Digital filters  66
      H.4.2  Filter design  68
   H.5  Measurement noise & torque limitation  68
      H.5.1  Measurement noise  68
      H.5.2  Torque limitation  68
H.6 Stability

H.7 Settings
  H.7.1 PID
  H.7.2 Acceleration feedforward
  H.7.3 Filter
  H.7.4 Measurement noise and torque limitation
  H.7.5 Definitive settings

H.8 Matlab-files
  H.8.1 WM281101.M
  H.8.2 CC2811.M
  H.8.3 H_PID.M
  H.8.4 FILT.M
Chapter 1 - Introduction

The subject of this research is the validation of Working Model, a two dimensional rigid multi body software package. Working Model is a simulation tool in which mass and geometry properties have to be added to the bodies. This adjudication of mass and geometry properties to the bodies makes Working Model differ from most simulation tools. This makes Working Model suitable for simulating geometric non-linearities. A system with significant geometric non-linearities in combination with a controller can result in unexpected situations. So it is very useful to model a system with significant geometric non-linearities together with a controller correctly to avoid unexpected situations.

Working Model models the geometric non-linearities but does not have the possibility to implement a controller. However Working Model can communicate with external applications. This communication is real time so it could be possible to implement a controller in an external application. Matlab is an application that is suitable for this purpose.

Another problem is the controller design. Designing a stable controller is mostly based on the frequency response of the system to control and the bandwidth in which the system has to perform. The frequency response of a system can not directly be determined in Working Model, so the frequency response has to be determined in an alternative way. Matlab offers a possibility to do this. Matlab uses the time response of a system to determine the frequency response of a system.

So with Working Model and Matlab it should be possible to model a controlled system with geometric non-linearities correctly. The Philips ADAT, an existing for beam mechanism with significant non-linearities is used as an example to investigate this.
Chapter 2 - Working Model 2.0

2.1 What is Working Model?

Working Model is software of Knowledge Revolution. This software runs under Microsoft Windows. How to use Working Model is easy to learn. The manual [1] is easy to understand and after reading the manual you can start using Working Model. Knowledge Revolution offers technical support to registered users when questions remain unanswered. This support consists of Internet access, fax support and telephone support.

Working Model is a two-dimensional rigid multi-body software. Two-dimensional means that Working Model only works with models in the two dimensional plane. So a three-dimensional mechanism has to be reduced to a two dimensional model, as far as this is possible. Multi-body means that Working Model will model the mechanism as a finite number of bodies that are connected with each other. Between these bodies relative movement is possible. These relative movements are constrained by constraints. Rigid means that in Working Model only rigid bodies and constraints can be implemented, so no flexibility can be added to the bodies. However Working Model does model elasticity, but not in the meaning of flexibility. The elasticity in Working Model corresponds to the coefficient of restitution used in simulating collisions. Besides the bodies (objects) Working Model can work with controls, meters and external application interfaces.

So real world mechanics can be implemented in Working Model with the restrictions that are mentioned. If this is done well, real world mechanics can be simulated on the computer.

2.2 Rigid bodies

All bodies in Working Model have mass, inertia and other physical properties. As mentioned before, flexible bodies can not be implemented. All bodies have a certain geometry in the two dimensional plane. According to the manual [1] in Working Model all bodies are one millimeter thick, independent of the unit system you use. All mass of a body is divided homogeneous over a body in Working Model.

2.3 Constraints

Four classes of constraints are in Working Model; linear constraints, rotational constraints, joint constraints and forces and torque’s. How to use these constraints can be found in the manual [1].


2.4 Meters and controls

Meters display the measured values of the physical properties in a simulation. Controls are used to adjust simulation parameters before and while simulating, e.g., motor torque. The meters can be read manually or by an external application like Matlab and the controls can be defined manually or by an external application like Matlab too. So those two tools are the in- and outputs of Working Model.

2.5 External application interface

Application interface, measurements and controls are strongly connected with each other. As mentioned, meters and controls are the tools that are the inputs and outputs of a simulation in Working Model. Working Model uses an external application interface to communicate with an external application. Such an external application interface can exchange data in real time using Dynamic Data Exchange (DDE).

2.6 Simulation

Working Model has other tools that are useful to create a simulation. How to use these tools you can read in the manual. These tools can define gravity, workspace, air resistance, animation, etc.

In Working Model the accuracy influences the simulation results too. To get good results the accuracy has to be set correct. The accuracy can be set under the world-menu. Pay attention to Appendix A of the manual [1] when setting the accuracy.
Chapter 3 - Dynamic Data Exchange

3.1 What is DDE?

As mentioned in Chapter 1, Working Model can communicate with external applications using an external application interface in combination with meters and controls. The manual says about this: "Working Model can exchange data in real time with external applications using Apple Events on the Macintosh or Dynamic Data Exchange (DDE) on Windows. This feature allows Working Model to exchange data with other applications once every animation step." This can be represented as shown in Figure 3.1.1.

![Figure 3.1.1 Communication between WM and Matlab](image)

This implies that data is sent to an external application, this application executes a command and data is sent back to Working Model. The applications under Windows that support DDE include:

- Microsoft Excel
- Quattro Pro
- MATLAB (version 4.2 or later)
- Microsoft Word for Windows

3.2 External Application Interface

How to implement an external application interface is described in the manual in chapter 9.14. - Real time Links with External Applications. Exercise 6 (Cruise Control using Matlab) in the tutorial [8] is a good example how to use an external application interface with Matlab. Running a simulation with Matlab as external application interface Working Model automatically starts up Matlab.

For communication with Matlab, implementing an external application interface can be summarized as follows:

1. Create meters and / or controls for the properties that you want to use for exchanging data with the external application
2. Select a New Application Interface from the define menu. This gives the following icon:
3. Double click the icon gives the following properties utility window

![Properties window](image)

Figure 3.2.2
Properties window of an application interface

4. Click the Application button and choose application. In this case Matlab, C:\...\matlab.exe. The version of Matlab must be 4.2 or higher. (Browsing is possible; double click Application.)

5. The document to link is always ENGINE in the case of Matlab.

6. Connect the inputs and outputs that have to exchange data with Matlab. You can attach a variable that will be known in Matlab as these in- and outputs.

7. In the Initialize-box you can insert a Matlab-routine or function that is executed at the beginning of the simulation.

8. In the Execute-box you can insert a Matlab routine or function that is executed every frame.

The most general structure for the matlab routines in the Initialize- and Execute box is:

\[
\text{inputs} = \text{name(output1, output2, .....)}
\]  

(3.2.1)

Where name is a function.

### 3.3 Sequence of DDE

Using Working Model 2.0, data exchange between WM and an external application takes place as follows:
Initialize remote commands
loop while simulation continues {
  get input data from external application
  run simulation step
  send output to external application
  Execute remote commands
}

In the case where the animation step size and integration step size are both equal to 1 and the external application is Matlab, the simulation will proceed as follows:

At $t = 0$

WM sends the "Initialize" command to Matlab
Matlab executes the "Initialize" command
WM gets values for Inputs from Matlab
WM integrates one step
WM sends values for Outputs to Matlab
Matlab executes the "Execute" command

At $t = 1$

WM gets values for Inputs from Matlab
WM integrates one step
WM sends values for Outputs to Matlab
Matlab executes the "Execute" command

and so on.

So the values send to Matlab at $t = 0$ are the values after integrating one step, so the values at $t = 1$. The values WM gets from Matlab at $t = 1$ are calculated at $t = 0$, thus a one frame time lag is introduced, see Appendix A. This might introduce some error in the data. This problem is known and is corrected for in version 3.0 of Working Model, which is now available.
Chapter 4 - Frequency response

The stability of a system that has to achieve a certain behavior is very important. Most mechanisms are designed to achieve a certain behavior, so these systems have to be stable. Many stability criteria are based on the frequency response of a system. So to say something about the stability of a system, the frequency response of that system is needed. Working Model does not have the possibility to measure the frequency response directly. In this chapter alternatives are mentioned how to determine the frequency response of a system.

4.1 How to determine a frequency response from the time response

The only response Working Model calculates is a time response of a given system. Now it is the point to find a way to determine the frequency response from time domain data. Matlab offers this possibility in two ways. The first is to use the powerspectra of the signals and the second method is based on an estimation of the system matrices. Both methods are described below.

4.1.1 Frequency response based on the power spectra of the time signals

The frequency response is the response of a system in the frequency domain. Working Model calculates the time response of a system, this is in the time domain. It is possible to transform time domain data to frequency domain data by using the Fourier transform. [2] describes how to determine the frequency response for a linear system with 1 input and 1 output based on the time domain data.

Given a linear system with 1 input and 1 output, see Figure 4.1.1.1.

\[ F(x(t)) = X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi if} dt \]  

\[ (4.1.1.1) \]

The time signals \( x(t) \) and \( y(t) \) can be transformed to the frequency domain by the Fourier transformation:

The autopowerspectra and crosspowerspectrum of \( x(t) \) and \( y(t) \) can be determined from \( X(f) \) and \( Y(f) \). The relation between \( X(f) \) and \( Y(f) \) is:
It is possible to determine the frequency response and coherence from these autopowerspectra and crosspowerspectrum.

If the autopowerspectrum $S_{xx}(f)$ and crosspowerspectrum $S_{xy}(f)$ are estimated, an estimation of the frequency response $H_{xy}(f)$ can be made as follows:

$$\hat{H}_{xy}(f) = \frac{\hat{S}_{xy}(f)}{\hat{S}_{xx}(f)}$$  \hspace{1cm} (4.1.1.3)

With the coherence $\gamma_{xy}^2(f)$ as a measure for the part of the output signal that comes from the input signal:

$$\hat{\gamma}_{xy}^2(f) = \frac{|\hat{S}_{xy}(f)|^2}{\hat{S}_{xx}(f)\hat{S}_{yy}(f)}$$  \hspace{1cm} (4.1.1.4)

The coherence lies between 0 and 1. If the coherence is 1 the output signal can be completely explained by the input signal. A coherence much smaller than 1 means that the output data cannot be completely explained by the input signal. This can be due to a low signal / noise ratio or non-linear system behavior. The relation between time domain and frequency domain is represented in Figure 4.1.1.2.

**Figure 4.1.1.2.**
Schematic representation from time domain to coherence and frequency response.

The estimation as described is based on the time response. This all is calculated in the Matlab routine `SPECTRUM` from the signal toolbox, that returns the frequency response and coherence of a time response.
4.1.2 Frequency response based on an estimation of the system matrices

This estimation is based on the time response too. Suppose that the system can be described as follows:

\[ A(q)y(t) = \frac{B(q)}{F(q)} u(t - nk) + \frac{C(q)}{D(q)} e(t) \]  \hfill (4.1.2.1)

Hence:

\[ y(t) = G(q)u(t - nk) + H(q)e(t) \]  \hfill (4.1.2.2)

with:

\[ G(q) = \frac{B(q)}{A(q)F(q)} \]  \hfill (4.1.2.3)

\[ H(q) = \frac{C(q)}{A(q)D(q)} \]  \hfill (4.1.2.4)

Where \( y(t) \) is the output signal and \( u(t) \) is the input signal.

The system matrices can be fitted to the known input and output signal. An estimation of the frequency response is \( H(q) \).

In the case of a single input single output system an ARX model is sufficient to determine the frequency response. An ARX model can be represented as follows:

\[ y(t) = H(q)u(t - nk) + G(q)e(t) \]  \hfill (4.1.2.5)

with

\[ G(q) = \frac{B(q)}{A(q)} \]  \hfill (4.1.2.6)

\[ H(q) = \frac{1}{A(q)} \]  \hfill (4.1.2.7)

This method is available in the identification toolbox of Matlab. The used routines are ARX and TRF, see Appendix B.

4.2 Results

The quality of the results depends on the accuracy and size of the dataset. The higher the accuracy the better the results will be. Calculation time becomes larger to realize higher accuracy. Larger datasets will also increase calculation time.

When you compare frequency responses of simple systems with the theoretical frequency response of these systems you see that the differences are small, see Appendix C. The difference in magnitude is determined as follows:

\[ \text{difference in magnitude} = 20 \log_{10}|H_{\text{model}}| - 20 \log_{10}|H_{\text{theoretical}}| \]  \hfill (4.2.1)
The difference in phase is determined as follows:

\[
\text{difference in phase} = |\text{phase}_{\text{theoretical}} - \text{phase}_{\text{WM}}| \tag{4.2.2}
\]

For example the frequency response, determined with SPECTRUM and ARX, of a mass-spring-damper system in Working Model with a positional error of 1 e-6 m gives a maximum difference in magnitude of 5 dB for SPECTRUM and 2 dB for ARX in the area between 1 and 10 Hz, see Figures 4.2.2 and 4.2.4. Frequencies up to one tenth are reliable, higher frequencies are less reliable. The maximum of 5 dB is near the resonance frequency, for other frequencies the difference is smaller than 3 dB, which is acceptable.

The difference in phase is less than 45 degrees between 1 and 10 Hz, see Figures 4.2.6 and 4.2.8.
Figure 4.2.5
Phase of Working Model (spectrum) response and theoretical phase

Figure 4.2.6
Difference in phase between Working Model (spectrum) and theoretical phase

Figure 4.2.7
Phase of Working Model (arx, n=7) response and theoretical phase

Figure 4.2.8
Difference in phase between Working Model (arx, n=7) and theoretical phase

For simulations with Matlab this difference is not significant, see Figure 4.2.10. The fact that the difference for the Working Model simulation significant is, is due to the sampling. Sampling introduces an additional time delay, so it introduces a phase lag.
So if you use the right settings in simulation and calculation, it is possible to determine a good frequency response of a system implemented in Working Model. It can be useful to use the SPECTRUM and ARX/FTR routines next to each other. This because fitting gives a smoother frequency response, especially for the phase, and the comparison with SPECTRUM to see you chose proper orders of the system matrices.
Chapter 5 - The ADAT

In the previous chapters it is mentioned how Working Model communicates with external applications and how frequency responses can be determined. A controller, that gives a stable closed-loop response, can be designed with the help of frequency responses of a system. In this chapter a real mechanism known as the ADAT will be implemented in Working Model. Matlab will calculate the frequency response of this mechanism from the time domain data from Working Model. A controller has been designed so the ADAT can perform certain desired movements. The desired movements and realized movements will be compared.

5.1 What is the ADAT

During production of IC’s, a wafer-die or crystal has to be picked up from a wafer and placed on a lead frame. The ADAT, a Philips die bonder, can do the above mentioned operations, i.e., to pick and place dies. This has to be achieved with a high accuracy. This high accuracy is reached by using a cleverly designed four-beam mechanism. So the ADAT can be modeled as a four-beam mechanism, see Figure 5.1.1.

![Figure 5.1.1](image)
The ADAT modeled as a four beam mechanism

The crank is driven by a motor and the transfer arm commutes between two extreme positions, when the crank rotates, see Figure 5.1.2. When the transfer arm is in the right extreme position, \( \theta \) is about 30 degrees. And when the transfer arm is in the left extreme position \( \theta \) is about 210 degrees. In these positions the unit at the tip of the transfer arm has to pick or place dies. Whether the unit picks or places a die depends on the position of the transfer arm. The cleverness of the four-beam mechanism is in the chosen geometry, see Appendix E. This geometry makes that the position of the tip in the extreme positions stays within the tolerance boundaries when the angle of the crank varies within the resolution of the angle measurements. In other words, the relative displacement of the tip is much smaller than the relative displacement of the crank in the extreme positions.
An other problem is that the tip of the transfer arm has to move from the one extreme position to the other. The extreme positions have to be reached by the tip of the transfer arm within a certain time and has to be kept a certain time so the pick or place action can take place. This can be realized by implementing a controller which makes the transfer arm to follow a certain trajectory. For this controller and trajectory the angle of the crank, $\theta$, is measured. An other parameter that is used is $\phi$. $\phi$ is 0 degrees for the place position and 180 degrees for the pick position. (for corresponding $\theta$ see Appendix E)

![Diagram](image)

**Figure 5.1.2.**
Pick and place positions of the ADAT

### 5.2 Rigid body dynamics of the ADAT

Working Model sees all masses as rigid bodies. In this case only the rigid body dynamics have to be considered.

[3] gives the rigid body dynamics that can be described by the equation of motion:

$$\frac{dJ(\theta)\dot{\theta}}{dt} = kI$$  \hspace{1cm} (5.2.1)

The inertia $J(\theta)$ depends on the angle $\theta$, and the motor is modeled as static gain. Equation 5.2.1 is equal to

$$J(\theta)\ddot{\theta} + J(\theta)\dot{\theta} = kI$$  \hspace{1cm} (5.2.2)

Hence,

$$J(\theta)\ddot{\theta} = -J(\theta)\dot{\theta} + kI$$  \hspace{1cm} (5.2.3)
This equation shows that when the angular speed is high, and recalling the fact that the inertia changes rapidly, the additional nonlinear term adds damping (either positive or negative) to the system dynamics.

Linearizing the system in each operating point \((\bar{\theta} = 0, \bar{I} = 0)\), the following model is obtained:

\[
J(\bar{\theta})\ddot{\theta} = kI
\]  \hspace{1cm} (5.2.4)

In which \(\bar{\theta}\) denotes the operating point, and where \(\theta\) and \(I\) now denote small perturbations around the operating point. Equation 5.2.4 shows that in each operating condition as defined above, the rigid body linear model has a standard double integrator structure, of which the gain is a function of the operating point.

5.3 Modeling the ADAT in Working Model

The ADAT is modeled in Working Model, see Figure 5.3.1 and appendix F.

![Figure 5.3.1](image)

The ADAT modeled in Working Model

The values and parameters of this model can be found in Appendix E.

5.4 Frequency response of the ADAT

The dynamics of the ADAT die bonder mechanism depend strongly on the position of the beams, just like its inertia. If the ADAT is modeled and implemented in Working Model correctly, the results of simulations should be the same as the theory.

The inertia of the ADAT strongly depends on position, so the frequency responses depend on the position of the ADAT. The frequency response is determined for \(\phi = 0\) degrees (Figure 5.4.1) and \(\phi = 90\) degrees (Figure 5.4.2). Fit with ARX (the order of
both system matrices A and B is set to 7) and SPECTRUM (size of the dataset is about 2000, so the sequence size \( N = 512 \) and the overlap \( m = 500 \)).

\[ \begin{align*}
\text{Figure 5.4.1} & \quad \text{Magnitude of the ADAT, } \varphi = 0 \\
\text{Figure 5.4.2} & \quad \text{Magnitude of the ADAT, } \varphi = 90
\end{align*} \]

An equivalent inertia can be estimated from these frequency responses. This because in each operating point the model has a double integrator structure. The equivalent inertia can be estimated as follows:

\[
\hat{J}_{\text{equivalent}} = \frac{1}{|H(f)|^2 (2\pi f)^2} \tag{5.4.1}
\]

Filling in points from Figures 5.4.1 and 5.4.2:

\[
\begin{align*}
\hat{J}(\varphi = 0) &= 2.1 \times 10^{-4} \text{ Ns}^2/\text{rad} \\
\hat{J}(\varphi = 90) &= 9.6 \times 10^{-4} \text{ Ns}^2/\text{rad}
\end{align*}
\]

This seems realistic because the theoretical and estimated equivalent inertia are of the same magnitude, see Appendix G.

**5.5 Implementing a controller**

A certain behavior is desired, so a controller has to be implemented to realize this behavior. Chosen is to implement a PPD controller with acceleration feedforward. A filter has to be implemented to eliminate unknown higher order dynamics, noise has to be added to the measured angle to create a realistic simulation and a motor restriction has to be implemented to create a realistic simulation too. These mentioned parts are designed and implemented, see Appendix H. This can be represented as follows:
The design of the parameters of all elements is done in Appendix H. The chosen parameters are:

\[ f_{\text{PID}} = 60 \text{ Hz} \]
\[ P = 9.4 \]
\[ I = 354.0 \]
\[ D = 0.075 \]
\[ J_{\text{feedforward}} = 2.6 \times 10^{-4} \text{ Nm}^2 \text{s} \]
\[ \text{filter at 300 Hz} \]
\[ \text{noise} = 0.1 \times (\text{rand} - 0.5) \]
\[ \text{torque limitation} = 1.7 \text{ Nm} \]

With these parameters you get results as follows:

**Time domain**

The used parameters are the parameters chosen in Appendix H.7.5 and mentioned before. The trajectory is the trajectory described in Appendix H.1.

In Figures 5.5.2, 5.5.3 and 5.5.4 the ADAT is controlled with a PID controller, Appendix H.2. Figure 5.5.2 shows the position of the crank, Figure 5.5.3 shows the position error of the crank and Figure 5.5.4 the torque on the crank.

**Figure 5.5.2**
Position of the crank for only PID

**Figure 5.5.3**
Position error of the crank for only PID
The trajectory from place to pick exists of three parts, first a positive acceleration, second no acceleration and as last an acceleration equal to the first acceleration but in the opposite direction. The torque is expected to have a symmetric shape with a positive maximum equal to the negative maximum apart from the direction. Figure 5.5.4 shows an other development of the torque. This other torque can be explained by the geometric non-linearity of the ADAT.

In Figures 5.5.5, 5.5.6 and 5.5.7 feedforward, see Appendix H.3, is added to the PID controller.

**Figure 5.5.4**
Torque on the crank for only PID

**Figure 5.5.5**
Position of the crank for PID with feedforward

**Figure 5.5.6**
Position error of the crank for PID with feedforward
The additional feedforward improves the behaviour of the crank. The pick and place position of the ADAT are reached earlier.

Because the motor has a maximum torque it can supply, in the simulations the torque is limited. In Figures 5.5.8, 5.5.9 and 5.5.10 the ADAT is controlled with a PID controller and the torque is limited at 1.7 Nm, see Appendix H.5.2.
Limiting the torque makes the behaviour worse, but stresses the fact that the ADAT has geometric non-linearities. The maximum torque is reached for slowing down, but the maximum torque is not reached for accelerating.

At last the ADAT is controlled with a PID controller together with feedforward, the torque is limited, the torque is filtered and measurement noise is added, see Appendix H. The behaviour of the ADAT is shown in Figures 5.5.11, 5.5.12 and 5.5.13.

**Figure 5.5.10**
Torque on the crank for PID with limited torque

**Figure 5.5.11**
Position of the crank

**Figure 5.5.12**
Position error of the crank
When the ADAT is controlled and modeled with measurement noise and torque limitation, it behaves well. The pick and place positions can be reached and held within the desired accuracy.

**Frequency domain**

The frequency response depends on the position in which the ADAT is. For two of these positions, \( \varphi = 0 \) degrees and \( \varphi = 90 \) degrees, the frequency response is determined. This frequency response is determined by holding the ADAT in the desired position by the corresponding setpoint and measuring the input and output signals. The frequency response is determined with both methods, ARX and SPECTRUM. The magnitude and phase are plotted in the Figures 5.5.14 up to and including 5.5.17. The left plots are for 0 degrees and the right plots are for 90 degrees. For the phase you have to correct the SPECTRUM results because an angle of \(-\pi\) is equal to a phase of \(\pi\).

![Figure 5.5.13](image)

**Figure 5.5.13**
Torque on the crank

![Figure 5.5.14](image)

**Figure 5.5.14**
Magnitude for \( \varphi = 0 \)

![Figure 5.5.15](image)

**Figure 5.5.15**
Magnitude for \( \varphi = 90 \)
**Figure 5.5.16**  
Phase for $\varphi = 0$

**Figure 5.5.17**  
Phase for $\varphi = 90$
Chapter 6 - Conclusions and recommendations

6.1 Conclusions

- Communication between Working Model can be realized very simply. This communication is reliable with a high accuracy.
- The time response of the simulation in Working Model can be easily transferred to Matlab and can be used to determine a reliable frequency response.
- Simulations in Working Model are discrete, so controller settings and sampling frequency must be chosen correctly.

6.2 Recommendations

- Investigate how easy or difficult it is to implement flexibility correctly in Working Model, because Working Model has not the possibility to do this directly.
- Investigate how reliable the frequency responses of systems with flexibility in Working Model are.
References


[6] Introduction to Control Theory for Engineers


[8] Tutorial, Knowledge Revolution, 1994, USA
Appendix A - Time lag

Example with Working Model in combination with Matlab to illustrate the one frame time lag that is introduced.

Essential settings of Working Model:

- **Gravity**: none
- **Accuracy**: Runge Kutta 4
- **Force**: integration time = variable
- **Mass**: animation time = 0.02 sec.
- **Force**: circle with mass = 1 kg
- **Meters**: on the mass, calculated in WM (2*\text{sin}(t))
- **External application**: force on mass, position of mass and time
- **Application**: application = Matlab
- **Initialize**: initialize = -
- **Execute**: execute = timelag
- **Inputs**: inputs = no
- **Outputs**: outputs = time, force, position

Matlab timelag.m:

```matlab
global T
global X
global F

T=[T time];
X=[X position];
F=[F force];
```

If you plot the values Matlab saved (plot(T,F)) versus 2*\text{sin}(t) and blow this up, you get Figure A.1. The time is shifted 0.02 seconds.

![Figure A.1.](image)

*Figure A.1.*
Two sines, one calculated in Matlab the other in Working Model
You see a time lag equal to one times the animation time (0.02 seconds) between the data from Working Model and \(2\sin(t)\).
Appendix B - Matlab spectrum and arx / trf routines

As mentioned the Matlab routines SPECTRUM, ARX and TRF have been used to calculate the frequency response of time responses.

B.1 Spectrum

The help text of the SPECTRUM routine is:

SPECTRUM Power spectrum estimate of one or two data sequences.
P = SPECTRUM(X,Y,M) performs FFT analysis of the two sequences X and Y using the Welch method of power spectrum estimation. The X and Y sequences of N points are divided into K sections of M points each (M must be a power of two). Using an M-point FFT, successive sections are Hanning windowed, FFT'd and accumulated. SPECTRUM returns the M/2 by 8 array
P = [Pxx Pyy Pxy Txy Cxy PXXC PyyC PxxC]
where
Pxx = X-vector power spectral density
Pyy = Y-vector power spectral density
Pxy = Cross spectral density
Txy = Complex transfer function from X to Y
(Use ABS and ANGLE for magnitude and phase)
Cxy = Coherence function between X and Y
PxxC, PyyC, PxxC = Confidence range (95 percent).

See SPECPLOT to plot these results.
P = SPECTRUM(X,Y,M,NOVERLAP) specifies that the M-point sections should overlap NOVERLAP points.
Pxx = SPECTRUM(X,M) and SPECTRUM(X,M,NOVERLAP) return the single sequence power spectrum and confidence range.

This routine returns the frequency response based on the time response of a system. The only parameters that need more explanation are M and NOVERLAP.

M determines the size of the sections on which the FFT is applied. M must be large enough so you have enough datapoints to determine a good FFT. M must be not to close to the size of X and Y, because you middle the results of the FFT’s of each section. M must be a power of two. A good value for M is the closest power of two near half the size of the input or output data.

The higher NOVERLAP is the more sections SPECTRUM can make out of the dataset to determine the FFT. The higher NOVERLAP is the better the result of SPECTRUM will be.

You get the best result of SPECTRUM for a large dataset, with M at about half the datasize and with a NOVERLAP that is 90 % of M. The only thing to take into account is the memory and calculation time of Matlab. When the chosen parameters make SPECTRUM take to many sections or to many points out of the dataset, memory problems occur and calculation time will be large.
The frequency response can be plotted using `SPECPLT`, but this gives a linear scale for the frequency and modified values of the frequency. An alternative method to plot the frequency response is to define a frequency matrix. The number of frequency points is equal to \(0.5 \times M\). The frequency begins at \(f = 0\) and ends at half the sampling frequency you used. So the frequency matrix can be defined as follows:

\[
f = 0 : \text{fsample}/M : 0.5 \times \text{fsam} - \text{fsam}/M; \quad (B.1.1)
\]

When you have the right frequency matrix you can plot the magnitude (\(\text{abs}(P(:,4))\)), the phase (\(\text{angle}(P(:,4))\)) and the coherence (\(P(:,5)\)) against the frequency (\(f\)).

### B.2 Arx / trf

The help text of the `ARX` routine is:

**ARX** Computes LS-estimates of ARX models

\[
\text{TH} = \text{arx}(Z, NN)
\]

- \(\text{TH}\): returned as the estimated parameters of the ARX model
- \(A(q) y(t) = B(q) u(t-nk) + e(t)\)
- along with estimated covariances and structure information.
- For the exact structure of TH see HELP THETA.

- \(Z\): the output-input data \(Z=[y \ u]\), with \(y\) and \(u\) as column vectors.
- For multivariable systems \(Z=[y_1 y_2 \ldots y_p \ u_1 u_2 \ldots u_m]\). For time series \(Z=y\) only.

- \(NN\): \(NN =[na \ nb \ nk]\), the orders and delays of the above model.
- For multi-output systems, \(NN\) has as many rows as there are outputs.
- \(na\) is then an \(n_{y} \times n_{y}\) matrix whose \(i-j\) entry gives the order of the polynomial (in the delay operator) relating the \(j\)-th output to the \(i\)-th output. Similarly \(nb\) and \(nk\) are \(n_{y} \times n_{u}\) matrices. (\(n_{y}\):# of outputs, \(n_{u}\):# of inputs). For a time series, \(NN=na\) only.
- Some parameters associated with the algorithm are accessed by \(TH = \text{arx}(Z, NN, \text{maxsize}, T)\)
- See HELP AUXVAR for an explanation of these and their default values.

And the help text of the `TRF` routine is:

**TRF** Computes a model's frequency function.

\[
\text{G} = \text{trf}(\text{TH}) \quad \text{or} \quad [\text{G, NSP}] = \text{trf}(\text{TH})
\]

- \(TH\): A matrix defining a model, as described in HELP THETA
- \(G\) is returned as the transfer function estimate, and \(NSP\) (if specified) as the noise spectrum, corresponding to the model \(TH\). These matrices contain also estimated standard deviations, calculated from the covariance matrix in \(TH\), and are of the standard frequency function format (see HELP FREQFUNC). If \(TH\) describes a time series, \(G\) is returned as its spectrum.

- If the model \(TH\) has several inputs, \(G\) will be returned as the transfer functions of selected inputs \# \(j_1 j_2 \ldots j_k\) by \(G = \text{trf}(\text{TH}, [j_1 j_2 \ldots j_k])\) [default is all inputs]. The functions
are computed at 128 equally spaced frequency-values between 0(excluded) and \( \pi T \), where \( T \) is the sampling interval specified by \( \text{TH} \). The functions can be computed at arbitrary frequencies \( \omega \) by \( G = \text{trf}(\text{TH}, \omega) \). The transfer function can be plotted by \text{BODEPLOT}. \text{bodeplot}(\text{trf}(\text{TH})) \) is a possible construction. If the model \( \text{TH} \) has several outputs, \( G \) will be returned as the frequency function at selected outputs \( ky \) (a row vector) by \( G = \text{trf}(\text{TH}, \omega, ky) \); (Default is all outputs). See also \text{TH2FF}.

\text{ARX} \) determines a matrix \( \text{th} \) that includes the matrices \( A \) and \( B \) of the \text{ARX}-model. The result of the frequency response you determine with \text{ARX} \) and \text{TRF} \) depends strongly on the order you choose for the \( A \) and \( B \) matrices. To see if you have chosen the right order, you could compare the result with the result of \text{SPECTRUM}, paying attention to the coherence of the \text{SPECTRUM} results. For \( nk \) you chose 0, because you assume there is no timelag.

With \( \text{th} \) you do not have a frequency response. The frequency response can be determined with the \text{TRF} \) routine and the output, \( \text{th} \), of \text{ARX}. You get \( H \) simply by typing:

\[ H = \text{trf}(\text{th}); \quad \text{(B.2.1)} \]

Next you must define the proper frequency matrix, \( \text{farx} \). \text{TRF} \) assumes that the response is determined for 129 frequencies from 0 to half the sampling frequency:

\[ \text{farx} = 0 : \text{fsam}/256 ; 0.5*\text{fsam} ; \quad \text{(B.2.2)} \]

If you have done this you can plot the magnitude (\( H(:,2) \)) and the phase (\( H(:,3) \)) against the frequency (\( \text{farx} \)).

29
Appendix C - Response comparison

C.1 How comparing responses

The only response Working Model can calculate is a time response. Matlab can calculate a frequency response out of this time response. The question is how far the time response and corresponding frequency response are comparable with the real frequency response. Frequency responses of Working Model, Matlab and theoretical frequency responses are compared to check how good the time and frequency response of Working Model are. The systems to implement have to be very simple so that the exact frequency responses are known. For a single mass system and a mass-spring-damper system the exact frequency responses can be easily calculated by Matlab. The frequency response of a single mass system can even be determined analytically.

Two systems were implemented to compare the frequency responses with the theoretical frequency response. First a single mass system is implemented in Simulink and the input signal is offered to the same system in Working Model and Matlab. The output signals have been compared in time and frequency domain. After this experiment a mass-spring-damper system is implemented in Matlab and the same input signal is offered to Working Model. The output signals have compared too. The result of this comparison gives an idea how good a frequency response of data from Working Model is.

C.2 A single mass system

The system that has been implemented is a single mass of 1 kilogram. This system was implemented in Simulink, Matlab and Working Model. The simulation results were transferred to the frequency domain by the use of matlab's SPECTRUM routine and have been compared with the theoretical response. The system has been implemented in the several applications as follows:

Simulink

The single mass system is implemented in Simulink, see Figure C.2.1.
The mass is 1 kilogram so the system equations are as follows:

\[
\dot{x} = Ax + Bu = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{C.2.1}
\]

\[
y = Cx + Du = [10]x + [0]u \tag{C.2.2}
\]

The input signal is generated by the addition of following three signals with a sample frequency of 100 Hz:

\[
u_1 = 1 \times \sin(50 \times t); \quad \text{ (sinewave)} \tag{C.2.3}
\]

\[
u_2 = 2 \times \sin(150 \times t); \quad \text{ (sinewave1)} \tag{C.2.4}
\]

\[
u_3 = 3 \times (\text{rand} - 0.5); \quad \text{ (signalgenerator)} \tag{C.2.5}
\]

\[
u = u_1 + u_2 + u_3 \quad \text{(sum)} \tag{C.2.6}
\]

Simulink calculates the time response. The time response is calculated for 999 time steps of 0.01 seconds. The fact that the timestep is 0.01 seconds is due to the fact that the signal generators have been set to a sample frequency of 100 Hz. The integration method is Runge Kutta 5 with a tolerance of 1e-10, maximum stepsize of 1e-10 and minimum stepsize of 1e-20.

The frequency response has been determined using the Matlab routine spectrum. The spectrum routine is described in Appendix B.1. The parameters of spectrum have been set to:

- section of \( N \) points; \( N = 512 \)
- overlap; \( m = 500 \)

Matlab

The input signal Simulink has generated, the time signal from Simulink and the corresponding system matrices \((A, B, C & D)\) have been inserted in the Matlab routine Lsim. This routine returns a time response for an input signal in combination with the corresponding time signal. This time response was transformed to the frequency
response by using the SPECTRUM routine in the same way as was done for the Simulink results.

**LSIM** uses the LTITR - function and a zero order hold for the input signal. How LTITR calculates the time response is unknown because LTITR is a built in function.

**Working Model**

A single mass is represented by a circle with a mass of 1 kilogram in Working Model. An external application interface with Matlab in combination with the proper control and meters is added to the Working Model system. The animation frequency has been set equal to the sample frequency of Simulink to get the same timesteps as in the Simulink simulation. The integration method of Working Model has been set to Runge Kutta 4 with a positional error of 10e-14. Matlab offers every frame the input signal Simulink has generated to Working Model, by using DDE. Working Model returns the time and time response to Matlab. Matlab stores these data.

The results of this Working Model simulation been have transferred to the frequency response in the same way as has been done for the Simulink and Matlab results. The only difference is that the time response of Working Model is corrected for the time lag introduced by using DDE as mentioned in chapter 2.

**Theoretical**

The single mass system is represented by the following transfer function:

\[ H(s) = \frac{1}{s^2} \]  

(C.2.7)

This can be written with polynomials as follows:

\[ H = \frac{\text{num}}{\text{den}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \]  

(C.2.8)

The num and den in combination with the \( \omega \), which are the same as for the other responses, are inserted in the Matlab routine BODE. This routine returns the frequency response for this system.

**Results**

In Figure C.2.2 you see the plotted results in the time domain (the timelag of Working Model is corrected for) and in Appendix D.1. the plotted results in the frequency domain for the single mass system as described before.
The difference in the time domain is due to different tolerance and integration methods. For example you create a cosine signal for the force \( F = 2 \cos(10 \times (t-0.01)) \) on a 1 kg mass for a certain time (0.01 sec.) in Working Model (integration method is RK-4, tol = 1e-10). The same signal you offer to the Matlab \( \text{LSIM} \) routine and you determine the exact response (this is possible for a cosine). You compare these responses, see Figures C.2.3 & C.2.4, and you will see a difference due to the difference in integration method.

The system is a mass-spring-damper system with the following parameters:

- spring constant: \( k = 2000 \text{ [N/m]} \)
- damping constant: \( b = 10 \text{ [Ns/m]} \)
- mass: \( m = 1 \text{ [kg]} \)

This gives the following system equation:
\begin{align*}
\dot{x} &= Ax + Bu = \begin{bmatrix} 0 & 1 \\ -1 & \frac{1}{k} \\ b & 1 \end{bmatrix}x + \begin{bmatrix} 0 \\ 1 \\ m \end{bmatrix}u = \\
&= \begin{bmatrix} 0 & 1 \\ -0.0005 & -0.1 \end{bmatrix}x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}x + \begin{bmatrix} 10 \end{bmatrix}u \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}x + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}u \
\end{align*}
(C.3.1)

\[ y = Cx + Du = \begin{bmatrix} 10 \end{bmatrix}x + \begin{bmatrix} 0 \end{bmatrix}u \]  
(C.3.2)

The transfer function can be represented as:

\[ H(s) = \frac{1}{s^2 + 10s + 2000} \]  
(C.3.3)

This can be written with polynomials as follows:

\[ H = \frac{\text{num}}{\text{den}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 10 & 2000 \end{bmatrix} \]  
(C.3.4)

The frequency response has been determined for the simulation with Working Model and for the Matlab time response based on the same input signal. The theoretical frequency response is used as reference to compare with the responses of Working Model and Matlab. This theoretical frequency response has been calculated in the same way as described for the single mass system. The time responses are not compared because they differ and the real time response is unknown. In the case of the mass-spring-damper system the frequency response is also determined by fitting the datasets to the system matrices. This is done by using Matlab’s ARX and TRF routines.

**Matlab**

The time and frequency response have been calculated in the same way as described for the single mass system.

The frequency response of the fitted system matrices is determined as follows:

With the Matlab routine ARX and the time response the system matrices are fitted. Next the frequency response is calculated with the Matlab TRF routine. The chosen model is ARX and the order of both matrices A and B is 4 and 7.

**Working Model**

The frequency response has been calculated in the same way as described for the single mass. The time response has been determined almost in the same way as described for the single mass system. The difference is that a damper and a spring, with the mentioned constants, have been added. These constraints have been connected to the background and to the mass. The positional error has been changed to 1e-6 instead of 1e-14 to speed up calculation. The decrease of the positional error leads to a noisier frequency response.

The frequency response based on the fitted system matrices is determined in the same way as described for the mass-spring-damper system in Matlab.
Results

In appendix D.2. you see the plotted results in the frequency domain for the mass-spring-damper system as described before.
Appendix D - Simulation results in the frequency domain

D.1 One mass system

Simulations for a system existing of one mass. The mass is 1 kg.

![Figure D.1.1](image1)
**Figure D.1.1**
Magnitude for the simulation in Matlab and the theoretical magnitude

![Figure D.1.2](image2)
**Figure D.1.2**
Difference in magnitude between the theoretical response and the Matlab response

![Figure D.1.3](image3)
**Figure D.1.3**
Phase for the simulation in Matlab and the theoretical magnitude

![Figure D.1.4](image4)
**Figure D.1.4**
Difference in phase between the theoretical response and the Matlab response

36
Figure D.1.5
Magnitude for the simulation in Simulink and the theoretical magnitude

Figure D.1.6
Difference in magnitude between the theoretical response and the Simulink response

Figure D.1.7
Phase for the simulation in Simulink and the theoretical magnitude

Figure D.1.8
Difference in phase between the theoretical response and the Simulink response
Figure D.1.9
Magnitude for the simulation with Working Model and the theoretical magnitude

Figure D.1.10
Difference in magnitude between the theoretical response and the Working Model response

Figure D.1.11
Phase for the simulation with Working Model and the theoretical phase

Figure D.1.12
Difference in phase between the theoretical response and the Working Model response
Figure D.1.13
Difference in magnitude between Working Model and Matlab response

Figure D.1.14
Difference in phase between Working Model and Matlab response
D.2 Mass-spring-damper system

Simulations for a system consisting of a mass, spring and damper. The mass is 1 kg, the spring has a spring constant of 2000 N/m and the damper has a damping constant of 10 Ns/m.

**Figure D.2.1**
Magnitude of Working Model response (spectrum) and theoretical magnitude

**Figure D.2.2**
Difference in magnitude between Working Model (spectrum) and theoretical response

**Figure D.2.3**
Phase of Working Model (spectrum) response and theoretical phase

**Figure D.2.4**
Difference in phase between Working Model (spectrum) and theoretical phase
Figure D.2.5
Magnitude of Matlab response (spectrum) and theoretical magnitude

Figure D.2.6
Difference in magnitude between Matlab (spectrum) and theoretical response

Figure D.2.7
Phase of Matlab (spectrum) response and theoretical phase

Figure D.2.8
Difference in phase between Matlab (spectrum) and theoretical phase
Figure D.2.9
Magnitude of Working Model response (arx, n=4) and theoretical magnitude

Figure D.2.10
Difference in magnitude between Working Model (arx, n=4) and theoretical response

Figure D.2.11
Phase of Working Model (arx, n=4) response and theoretical phase

Figure D.2.12
Difference in phase between Working Model (arx, n=4) and theoretical phase
Figure D.2.13
Magnitude of Working Model response (arx, n=7) and theoretical magnitude

Figure D.2.14
Difference in magnitude between Working Model (arx, n=7) and theoretical response

Figure D.2.15
Phase of Working Model (arx, n=7) response and theoretical phase

Figure D.2.16
Difference in phase between Working Model (arx, n=7) and theoretical phase
Magnitude of Matlab response \( (\text{am } n=4) \) and theoretical magnitude

\[
\begin{align*}
\text{phase} [\text{deg}] & \quad \text{log(frequency) [Hz]} \\
0 & \quad 0 \\
-90 & \quad 1 \\
-180 & \quad 2
\end{align*}
\]

Figure D.2.17
Magnitude of Matlab response \((\text{arx } n=4)\) and theoretical magnitude

Difference in magnitude between Matlab \((\text{arx, } n=4)\) and theoretical response

\[
\begin{align*}
difference [\text{dB}] & \quad \text{log(frequency) [Hz]} \\
0 & \quad 0 \\
1 & \quad 1 \\
4 & \quad 2
\end{align*}
\]

Figure D.2.18
Difference in magnitude between Matlab \((\text{arx, } n=4)\) and theoretical response

Phase of Matlab \((\text{arx, } n=4)\) response and theoretical phase

\[
\begin{align*}
\text{phase} [\text{deg}] & \quad \text{log(frequency) [Hz]} \\
0 & \quad 0 \\
-90 & \quad 1 \\
-180 & \quad 2
\end{align*}
\]

Figure D.2.19
Phase of Matlab \((\text{arx, } n=4)\) response and theoretical phase

Difference in phase between Matlab \((\text{arx, } n=4)\) and theoretical phase

\[
\begin{align*}
difference [\text{deg}] & \quad \text{log(frequency) [Hz]} \\
0 & \quad 0 \\
45 & \quad 1 \\
90 & \quad 2
\end{align*}
\]

Figure D.2.20
Difference in phase between Matlab \((\text{arx, } n=4)\) and theoretical phase
Figure D.2.21
Magnitude of Matlab response (arx n=7) and theoretical magnitude

Figure D.2.22
Difference in magnitude between Matlab (arx, n=7) and theoretical response

Figure D.2.23
Phase of Matlab (arx, n=7) response and theoretical phase

Figure D.2.24
Difference in phase between Matlab (arx, n=7) and theoretical phase
Appendix E - Geometry and properties of the ADAT

E.1 Geometry of the ADAT

The ADAT is a four beam mechanism, that exists of a crank, a linker and a transfer arm, with a geometry as shown in Figure E.1.1.

![Figure E.1.1](image)

Geometry of the ADAT as four beam mechanism

All beams are made of aluminum. Aluminum has a density of $2.7 \times 10^{-3}$ kg/mm$^3$.

E.1.1 Angle of the crank for the pick and place position

The extreme positions of the transfer arm are those positions in which the crank and the linkage are in line. The angles the crank makes with the horizontal position are calculated by Equations E.1.1.1, E.1.1.2 and E.1.1.3 corresponding to Figure E.1.1.1.
With:

\[ a = 120 \text{ mm} \]
\[ b_1 = 120 \text{ mm} \]
\[ b_2 = 240 \text{ mm} \]
\[ c = 208 \text{ mm} \]

Angle for the pick position:

\[ a^2 = b_1^2 + c^2 - 2b_1c \cos(\alpha_1) \]  \hspace{1cm} (E.1.1.1)

gives:

\[ \alpha_1 = 0.5223 \text{ radians} \quad (\text{about 30 degrees}) \]

Angle for the place position:

\[ a^2 = (b_1 + b_2)^2 + c^2 - 2b_1c \cos(\alpha_1) \]  \hspace{1cm} (E.1.1.2)
\[ \alpha_2 = \alpha_1 + \pi \]  \hspace{1cm} (E.1.1.3)

gives:

\[ \alpha_2 = 3.664 \text{ radians} \quad (\text{about 210 degrees}) \]

E.1.2 Displacement of the tip

The small displacement of the tip of the transfer arm due to a small displacement of \( \phi \) in the extreme positions, is much smaller than the displacement of \( \phi \). This can be shown by varying the angle of the crank, \( \phi \), and measuring the displacement of the tip.
of the transfer arm. This can be done in Working Model by changing and reading the values in the properties of the crank and point 16. This gives:

\[
\begin{align*}
\varphi_{180} &= 3.664871 \, \text{e}^0 \,[\text{rad}] \\
\varphi_{180} + \delta\varphi &= 3.662871 \, \text{e}^0 \,[\text{rad}] \\
\delta\varphi &= 2.0 \, \text{e}^{-3} \,[\text{rad}] \\
\delta p_{16} &= 1.0 \, \text{e}^{-7} \,[\text{m}] \\
\varphi_0 &= 0.523278 \, \text{e}^0 \,[\text{rad}] \\
\varphi_0 + \delta\varphi &= 0.525278 \, \text{e}^0 \,[\text{rad}] \\
\delta\varphi &= 2.0 \, \text{e}^{-3} \,[\text{rad}] \\
\delta p_{16} &= 2.0 \, \text{e}^{-8} \,[\text{m}]
\end{align*}
\]

\[
\begin{align*}
x_{p16} &= 2.880104 \, \text{e}^{-8} \,[\text{m}] \\
y_{p16} &= 1.197331 \, \text{e}^{-1} \,[\text{m}] \\
x_{p16} &= 2.543860 \, \text{e}^{-8} \,[\text{m}] \\
y_{p16} &= 1.197332 \, \text{e}^{-1} \,[\text{m}] \\
x_{p16} &= 2.076923 \, \text{e}^{-1} \,[\text{m}] \\
y_{p16} &= 2.399998 \, \text{e}^{-1} \,[\text{m}] \\
x_{p16} &= 2.076921 \, \text{e}^{-1} \,[\text{m}] \\
y_{p16} &= 2.399998 \, \text{e}^{-1} \,[\text{m}] \\
x_{p16} &= 2.0 \, \text{e}^{-7} \,[\text{m}] \\
y_{p16} &= 5 \, \text{e}^{-8} \,[\text{m}]
\end{align*}
\]

The angle \( \varphi = 0 \) degrees is most important because the dies are placed there. The displacement of the tip of the transfer arm is 2 e-8 m when the angle of the crank varies about the resolution of the encoder, 2 e-3 rad. This means that the dies can be placed with an accuracy of 2 e-8 m.

**E.2 Geometry and properties of the crank (Motor included)**

The simplified representation of the crank is shown in Figure E.2.1.

![Figure E.2.1](image_url)

Figure E.2.1.
Simplified representation of the crank of the ADAT

\[
\begin{align*}
\text{rp} &= \text{rotation point} \\
\text{cm} &= \text{center of mass} \\
b_{1\text{crank}} &= 25 \, \text{mm} \\
I_{\text{crank}} &= 70 \, \text{mm}
\end{align*}
\]
In Working Model the crank is modeled as a rectangle, see Figure E.2.2., with following geometry and properties.

**Figure E.2.2**
The crank of the ADAT simplified to a rectangle

**Geometry:**

- \( l_{\text{crank}} = 60 \text{ mm} \)
- \( b_{\text{crank}} = 10 \text{ mm} \)
- \( z_{\text{crank}} = 30 \text{ mm} \)

point \( p_1 \) is attached to: the motor  
point \( p_2 \) is attached to: the linkage

**Properties:**

Used estimation of the mass of the crank is the average width of the crank * the length of the crank * the height of the crank * density of aluminum, this gives:

\[
\hat{m}_{\text{crank}} = \frac{(b_1_{\text{crank}} + b_2_{\text{crank}})}{2} \times l_{\text{crank}} \times h_{\text{crank}} \times \rho \quad (E.2.1)
\]

To model the ADAT correctly the inertia of the motor has to be modeled too. The inertia of the motor and crank are combined in the inertia of the crank.

The motor used in the ADAT is an Electro Craft S-243-1A with following specifications:
Table E.2.1. Specification of the Electro Craft S-243-1A

The inertia of the Electrocraft S-243-1A motor is 0.52 kg·cm².

The inertia of crank and motor on the driving axis is measured and appears to be 1.0 e⁻⁴ kg·m². To realize this an additional inertia has to be introduced besides the mass of the crank. This inertia can be estimated by using Steiner’s law.

\[ J_{rp} = J_{cm} + ma^2 \]  
\[ (E.2.2) \]

With \( J_{cm} \) the inertia round the axis through the center of mass.

\[ J_{rp} = 1.0 \times 10^{-4} \text{ kg·m}^2 \]

Used estimation of the inertia of the crank;

\[ J_{cm} = J_{rp} - m_{crank}a^2 = 0.69 \times 10^{-4} \text{ kg·m}^2 \]  
\[ (E.2.3) \]

\[ \dot{J}_{crank} = 0.69 \times 10^{-4} \text{ kg·m}^2 \]
E.3 Geometry and properties of the linkage

The simplified representation of the linkage is shown in Figure E.3.1.

![Diagram of the linkage](image)

Figure E.3.1
Simplified representation of the linkage of the ADAT

\[ l_A = l_C = 15 \text{ mm} \]
\[ l_B = 100 \text{ mm} \]
\[ b_A = b_C = 15 \text{ mm} \]
\[ h_A = h_C = 40 \text{ mm} \]
\[ d_b = 6 \text{ mm} \]
\[ z_A = z_C = 57.5 \text{ mm} \]
\[ z_B = 0 \text{ mm} \]
\[ \rho = 2.7 \times 10^{-3} \text{ gr / mm}^3 \]

In Working Model the linkage is modeled as a rectangle, see Figure E.2.2, with following geometry and properties.

**Geometry:**

- \[ l_{\text{linkage}} = 180 \text{ mm} \]
- \[ b_{\text{linkage}} = 10 \text{ mm} \]
- \[ z_{\text{linkage}} = 90 \text{ mm} \]

- point \( p_1 \) is attached to: the crank
- point \( p_2 \) is attached to: the transfer arm

**Properties:**

The linkage is modeled as three separate masses. Used estimation of the mass of the linkage is the sum of the separated estimated masses.

\[ m_{\text{part A}} = m_{\text{part C}} = l_A * b_A * h_A * \rho = 24 \text{ gr} \quad (E.3.1) \]
\[ m_{\text{part B}} = (d_c/2)^2 * \pi * l_C * \rho = 8 \text{ gr} \quad (E.3.2) \]
\[ m_{\text{linkage}} = 2 * m_{\text{part A}} + m_{\text{part B}} = 56 \text{ gr} \quad (E.3.3) \]

Estimation of the inertia of the linkage is the sum of the inertia of the separated parts. The inertia is calculated to the center of mass of the complete linkage. Because the linkage is modeled to be symmetric, the center of mass is exactly in the middle of the linker.

\[ J_{\text{part A}} = J_{\text{part C}} = 1/12 * m_A * (l_A^2 + b_A^2) + m_A * z_A^2 = 0.80 \times 10^{-4} \text{ kg m}^2 \quad (E.3.4) \]
\[ J_{\text{part B}} = 1/2 * m_B * (6 * l_B^2 + l_B^2) + m_B * z_B^2 = 0.38 \times 10^{-4} \text{ kg m}^2 \quad (E.3.5) \]
\[ J_{\text{linkage}} = J_{\text{part B}} + 2 * J_{\text{part A}} = 2.0 \times 10^{-4} \text{ kg m}^2 \quad (E.3.6) \]
E.4 Geometry and properties of the transfer arm

The simplified representation of the transfer arm is shown in Figure E.4.1.

Figure E.4.1
Simplified representation of the transfer arm of the ADAT

The transfer arm is built of four parts, a cylinder and three beams with the same geometry. The three beams form a T-profile. Cm is the center of mass of the complete transfer arm and rp the rotation point of the complete transfer arm.

\[ Z_c = 70 \text{ mm} \quad z_1 = z_2 = z_3 = 60 \text{ mm} \]
\[ d_i = 20 \text{ mm} \quad d_o = 30 \text{ mm} \]
\[ l = 260 \text{ mm} \quad b = 25 \text{ mm} \]
\[ h = 25 \text{ mm} \quad d = 1 \text{ mm} \]

In Working Model the transfer arm is modeled as a rectangle, see Figure E.2.2, with following geometry and properties.

**Geometry:**

- \( l_{\text{transfer arm}} = 240 \text{ mm} \)
- \( b_{\text{transfer arm}} = 10 \text{ mm} \)
- \( Z_{\text{transfer arm}} = 120 \text{ mm} \)

- point \( p_1 \) is attached to: the linkage
- point \( p_2 \) is attached to: the background

**Properties:**

The mass of the transfer arm is equal to the sum of the masses of the four parts:

\[ m_{\text{beam}} = \rho * l * b * d \quad = 17.5 \text{ gram} \tag{E.4.1} \]
\[ \hat{m}_{\text{cylinder}} = \rho * 2\pi * \frac{(d_i + d_o) * (d_o - d_i)}{2} = 70.0 \text{ gram} \tag{E.4.2} \]
\[ \hat{m}_{\text{transfer arm}} = 3 * \hat{m}_{\text{beams}} + \hat{m}_{\text{cylinder}} = 122.6 \text{ gram} \tag{E.4.3} \]

The inertia is estimated as follows; First the inertia of each separate part relative to its own center of mass is determined and next these inertia are calculated round the center of mass of the complete transfer arm. This is done by using Steiner’s law, see Equation E.2.2. The beams do not have the same inertia, the lying beam has a different inertia from the two standing beams.
\[ J_{\text{cylinder, cm-cylinder}} = \frac{1}{2} m_{\text{cylinder}} \left( \frac{d_c}{2} \right)^2 + \left( \frac{d_c}{2} \right)^2 = 0.11 \times 10^{-4} \text{ kg m}^2 \] (E.4.4)

\[ J_{\text{lying-beam, cm-lying-beam}} = \frac{1}{12} m_{\text{beam}} \left( b^2 + l^2 \right) = 1.0 \times 10^{-4} \text{ kg m}^2 \] (E.4.5)

\[ J_{\text{standing-beam, cm-standing-beam}} = \frac{1}{12} m_{\text{beam}} \left( b^2 + l^2 \right) = 1.0 \times 10^{-4} \text{ kg m}^2 \] (E.4.6)

Now the inertia round the center of mass of the complete transfer arm can be calculated. First you take the inertia of the three beams round the center of mass of the complete transfer arm. Assume that the distances from the center of mass of the separate beams to the center of mass of the complete transfer arm \( z_{1,2,3} \) are all equal to 70 mm.

\[ J_{\text{lying-beam, cm-transfer arm}} = \frac{1}{12} m_{\text{beam}} \left( b^2 + l^2 \right) + m_{\text{beam}} z_{2}^2 = 1.86 \times 10^{-4} \text{ kg m}^2 \] (E.4.7)

\[ J_{\text{standing-beam, cm-transfer arm}} = \frac{1}{12} m_{\text{beam}} \left( b^2 + l^2 \right) + m_{\text{beam}} z_{1,3}^2 = 1.86 \times 10^{-4} \text{ kg m}^2 \] (E.4.8)

Next you take the inertia of the cylinder round the center of mass of the complete transfer arm:

\[ J_{\text{cylinder, cm-transfer arm}} = \frac{1}{2} m_{\text{cylinder}} \left( \frac{d_c}{2} \right)^2 + \left( \frac{d_c}{2} \right)^2 + m_{\text{cylinder}} z_{c}^2 = 3.54 \times 10^{-4} \text{ kg m}^2 \] (E.4.9)

Then you add these inertia and you get the estimated inertia of the transfer arm round its center of mass:

\[ J_{\text{transfer arm, cm-transfer arm}} = J_{\text{cylinder, cm-transfer arm}} + 2 J_{\text{standing-beam, cm-transfer arm}} + J_{\text{lying-beam, cm-transfer arm}} \] (E.4.10)

\[ J_{\text{transfer arm, cm-transfer arm}} = 9.12 \times 10^{-4} \text{ kg m}^2 \]

**E.5 Remark**

If you compare simulation results based on the estimated values with measurements on the ADAT, you will see that the results differ. I did not try to get the estimated values closer to the real values.
Appendix F - The ADAT in Working Model

For the values of mass, inertia, geometry etc see Appendix E. Default settings were used for values that are not mentioned in Appendix E. In this appendix you see with which elements the ADAT is implemented in Working Model and how the elements are connected to each other.

Simulation

Custom
Animation Step  5.000e-04 s ; 2000 Hz
Positional Error  1.000e-10 m
Integrator Runge Kutta 4
Integration time step Variable

Gravity None

Air Resistance None

Masses

Mass[1] rectangle crank
Mass[2] rectangle transfer arm
Mass[3] rectangle linkage

Constraints

Constraint[6] Pin Joint
Constraint[9] Pin Joint
Point[7] , Point[8]
Constraint[14] Pin
Point[12] , Point[10]
Constraint[15] Pin
Constraint[20] Torque
value : Input[21]
basepoint : Point[19]

Points

Point[4] Background
Constraint[6]
Point[8] Background 
Point[12] Constraint[9] 
Point[16] Constraint[14] 
Point[3] Square Point 
Point[20] Mass[2], tip of transfer arm 
Point[16] Basepoint 

All points lie at the outline of the rectangles and at the middle of the line on which they lie.

Meters and Controls

Input[21] Torque 
Output[17] Time 
Output[18] Phi , rot = mass[1].p.r*57.296 
Output[26] Position of Square Point 16

External Application

External document #22 Application
Document c:\matlab42\bin\matlab.exe 
Connected outputs engine 
output[17].y1 variable: twm 
output[18].y1 variable: phi 
T = 0; 
T = wm281101(phi,twm); 
input[21] variable: T 
Timeout 30
Appendix G - Estimation of the inertia of the ADAT

When estimating inertia always remind that inertia can be dependent of position. So the estimated inertia is only correct for that specific position. The theoretical estimation is used to check if the practical estimation, which should be the real equivalent inertia, has the same magnitude.

G.1 Theoretical estimation

The inertia of the ADAT can be estimated theoretically in the following way:

1. Vary the position of the system with a small perturbation in a certain position, see Figure G.1.1. In each timestep $\delta t$, you get the following perturbations: $\delta \phi_1$, $\delta \phi_2$, $\delta \phi_3$. The velocities of these perturbations are $\delta \phi_1/\delta t$, $\delta \phi_2/\delta t$ and $\delta \phi_3/\delta t$. To compare the results of the theoretical estimation with the practical estimation, you should perturbate in the same positions as the position in which the frequency response is determined.

2. These perturbations and their velocities can be expressed in one perturbation. When you want to estimate an inertia to compare with the practical estimated inertia, the perturbation you express the other perturbation in should be the output signal of the system. In the case of the ADAT this is $\delta \phi_1$ and you get:

\[
\begin{align*}
\delta \phi_1 &= \delta \phi_1 & \quad \text{(G.1.1)} \\
\delta \phi_2 &= \delta \phi_2(\delta \phi_1) & \quad \text{(G.1.3)} \\
\delta \phi_3 &= \delta \phi_3(\delta \phi_1) & \quad \text{(G.1.5)}
\end{align*}
\]

\[
\delta \phi_1/\delta t = \frac{\delta \phi_1}{\delta t} & \quad \text{(G.1.2)} \\
\delta \phi_2/\delta t = \frac{\delta \phi_2}{\delta t}(\delta \phi_1) & \quad \text{(G.1.4)} \\
\delta \phi_3/\delta t = \frac{\delta \phi_3}{\delta t}(\delta \phi_1) & \quad \text{(G.1.6)}
\]

3. An equivalent inertia for the ADAT expressed in perturbations of the mechanism can be given as follows:

\[
T_{\text{kinetic}} = \frac{1}{2} \left( \frac{\delta \phi_1}{\delta t} \right)^2
\]

(G.1.7)

With:

\[
T_{\text{kinetic}} = \frac{1}{2} \left( \sum_{n=1}^{n-1} \left( \frac{\delta \phi_n(\delta \phi_1)}{\delta t} \right)^2 + m_a \left( \frac{\delta \phi_n(\delta \phi_1)}{\delta t} \right)^2 \right)
\]

(G.1.8)
Hence:

\[ J_{\text{equivalent}} = \sum_{n=1}^{\infty} \left( J_n \left( \frac{\partial \varphi_n}{\partial t} \right)^2 + m_n z_n^2 \left( \frac{\partial \varphi_n}{\partial t} \right)^2 \right) \]  \hspace{1cm} (G.1.9)

\[ J_{\text{equivalent}} = \sum_{n=1}^{\infty} \left( J_n \left( \frac{\partial \varphi_1}{\partial t} \right)^2 + m_n z_n^2 \left( \frac{\partial \varphi_1}{\partial t} \right)^2 \right) \]  \hspace{1cm} (G.1.10)

With \( z_n \) distance between center of mass of beam \( n \) and axes where the motor drives the crank. Assumed that the difference in \( z_n \) is negligible to the value of \( z_n \), so the value of \( z_n \) is the value of \( z_n \) before perturbing.

**Figure G.1.1**

Perturbations of the ADAT as a four beam mechanism

The accuracy of the estimation of the inertia depends on the accuracy of the measurement of the perturbations.

**G.1.1 J_{\text{equivalent}} for the ADAT when \( \varphi = 0 \) degrees**

The perturbations and \( z_n \) have been determined by Working Model. When you perturbate a beam with a certain amount in the properties of that beam, the perturbation of the other beams can be read in their properties. The Numbers & Units in the View menu have to be set to the correct number of digits. The number of digits has to be the size of the smallest perturbation at least.

\( Z_n \) can be determined as follows:

\[ z_n = \sqrt{x_n^2 + y_n^2} \]  \hspace{1cm} (G.1.1.1)

Where the \( x \) and \( y \) position of the center of mass of a beam can be read in the properties of that beam.
perturbations and their velocities

\[
\delta \phi_1 = \delta \phi_1 \\
\left( \frac{\delta \phi_1}{\delta \phi_1} \right)^2 = 1
\]

\[
\delta \phi_2 = 0.34 \times \delta \phi_1 \\
\left( \frac{\delta \phi_2}{\delta \phi_1} \right)^2 = 0.11
\]

\[
\delta \phi_3 = 0.02 \times \delta \phi_1 \\
\left( \frac{\delta \phi_3}{\delta \phi_1} \right)^2 = 4 \times 10^{-4}
\]

Filling in \( z_n \) and the perturbations in equation G.1.10 gives:

\[
J_{\text{equivalent}} = 2.5 \times 10^{-4} \quad [\text{Nm s}^2]
\]

G.1.2 \( J_{\text{equivalent}} \) for the ADAT when \( \phi = 90 \) degrees

The perturbations and \( z_n \) have been determined in the same way as done for 0 degrees. You get the following perturbations and \( z_n \):

perturbations and their velocities

\[
\delta \phi_1 = \delta \phi_1 \\
\left( \frac{\delta \phi_1}{\delta \phi_1} \right)^2 = 1
\]

\[
\delta \phi_2 = 1.23 \times 10^{-2} \times \delta \phi_1 \\
\left( \frac{\delta \phi_2}{\delta \phi_1} \right)^2 = 1.5 \times 10^{-4}
\]

\[
\delta \phi_3 = 0.52 \times \delta \phi_1 \\
\left( \frac{\delta \phi_3}{\delta \phi_1} \right)^2 = 0.27
\]

Filling in \( z_n \) and the perturbations in Equation G.1.10 gives:

\[
J_{\text{equivalent}} = 13.9 \times 10^{-4} \quad [\text{Nm s}^2]
\]
G.2 Practical estimation

The inertia of the ADAT can be estimated practically by using the frequency response. The frequency response of a system with no non-linear terms can be described as follows:

\[ H(j\omega) = \frac{1}{-\omega^2 J + j\omega b + k} \]  

(G.2.1)

When \( \omega \) goes to infinity, \(|H(j\omega)|\) goes to \(1/\omega^2\). By plotting the frequency response in double logarithmic scales, the plot will show a straight line with a decay of 2 for large \( \omega \) when no nonlinearities are in the system.

By fitting a straight line through the frequency response at large \( \omega \), an estimation of the inertia can be made by filling in the following equation:

\[ J_{\text{estimated}} = 1/(|H(\omega)|^2 \omega^2) \]  

(G.2.2)

G.2.1 \( J_{\text{estimated}} \) for the ADAT when \( \varphi = 0 \) degrees

The magnitude of the frequency response of the ADAT for \( \varphi = 0 \) degrees is plotted in Figure G.2.1.1.

![Figure G.2.1.1](image)

**Figure G.2.1.1**

Frequency response of the ADAT for \( \varphi = 0 \) degrees

The frequency response is determined with SPECTRUM and ARX. The positional error in Working Model was to 1 e-1 m. The values to fill in in Equation G.2.2 are from the arx-dataset. These values are:

\[ f^* = 101.56 \quad \text{[Hz]} \]
\[ |H(0,f^*)| = 0.6772 \quad \text{[rad/Nm]} \]

These values give:

\[ J_{\text{estimated}} (0) = 2.1 \times 10^{-4} \quad \text{[Nm s^2/rad]} \]
G.2.2 \( J_{\text{estimated}} \) for the ADAT when \( \phi = 90 \) degrees

The magnitude of the frequency response of the ADAT for \( \phi = 90 \) degrees is plotted in Figure G.2.1.2.

![Figure G.2.1.2](image)

**Figure G.2.1.2**
Frequency response of the ADAT for \( \phi = 90 \) degrees

The frequency response is determined with SPECTRUM and ARX. The values to fill in in Equation G.2.2 are from the arx-dataset. The positional error in Working Model was to 1 e-1 m. These values are:

\[
\begin{align*}
    f^* &= 101.56 \text{ [Hz]} \\
    |H(90,f^*)| &= 0.147 \text{ [rad/Nm]}
\end{align*}
\]

These values give:

\[
\begin{align*}
    J_{\text{estimated}} (90) &= 9.6 \times 10^{-4} \text{ [Nm s}^2\text{/rad]}
\end{align*}
\]
Appendix H - Controller design

The design of the controller described in this appendix, is the design of:

- trajectory
- PID-controller
- feedforward
- filter
- measurement noise & torque limitation

At last something is said about stability and the Matlab files of the implemented controller are added.

H.1 Trajectory

If a system, such as the ADAT, has to perform a certain behavior, this behavior has to be specified. This specification is called the trajectory. The desired behavior of the ADAT is moving from place position to pick position and back in a very short time. A constraint to this is that the ADAT has to hold the pick and place position a certain time. Because it is not possible to jump from the pick to the place position, a smooth trajectory has to be designed. The time to move from pick position to place position or back is 105 msec. and the time to hold the pick position or place position is set to 300 msec.

A second order trajectory is chosen, so the velocity will be smooth, but acceleration will contain discontinuities. Between the pick and place position is a difference of 180 degrees, so the trajectory will be from 0 degrees to 180 degrees. The moving time is divided in three equal parts of 35 msec. In the first part the acceleration is positive, in the second part the acceleration is zero and in the last part the acceleration is negative. At the starting and ending time the velocity is zero, because the pick and place position will be hold for a certain time. Now problem and conditions are defined and the trajectory can be designed.

H.1.1 Design of the trajectory

\[
\begin{align*}
  t &= 0 - 35ms & a &= +u & \text{(H.1.1.1)} \\
  t &= 35 - 70ms & a &= 0 & \text{(H.1.1.2)} \\
  t &= 70 - 105ms & a &= -u & \text{(H.1.1.3)}
\end{align*}
\]

\[
\begin{align*}
  v &= +at + v(0) & x &= +.5at^2 + x(0) & \text{(H.1.1.4)} \\
  v &= v(35) & x &= .5v(35)(t - 35)^2 + x(35) & \text{(H.1.1.5)} \\
  v &= -a(t - 70) + v(70) & x &= -.5a(t - 70)^2 + x(70) & \text{(H.1.1.6)}
\end{align*}
\]

\[
\begin{align*}
  v(0) &= 0 & v(105) &= 0 & x(0) &= 0 & x(105) &= 180
\end{align*}
\]
Solving these equations with initial conditions gives an acceleration of:

\[ a = 7.347 \times 10^4 \text{ m/s}^2 \]

With this acceleration the position, velocity and acceleration will have the shape as shown in Figure H.1.1.1.

This trajectory is calculated in Simulink and used in the controller implemented in Matlab. The trajectory determined with a sample frequency of 800 Hz and 2000 Hz. These datasets are stored in trace800.mat and trac2000.mat.

**H.2 PID-controller**

If the ADAT will have to pick and place dies it has to follow a trajectory such as specified before. To follow this trajectory a controller is desired. A PID controller is chosen. The controller has to be designed for the continuous time case, because you simulate on a computer that is in discrete time. The controller is not directly designed for the discrete time case, but first for the continuous time case and then the discrete time case is examined.

**H.2.1 Continuous time case**

The controller has been designed in the frequency domain. A PID-controller is a controller with a frequency response shown in Figure H.2.1.1.
A stable PID-controller for a system is a PID controller where the parameters have the following relations:

\[
\begin{align*}
1/\tau_i &= 1/10 \times \omega_b & (H.2.1.1) \\
1/\tau_d &= 1/3 \times \omega_b & (H.2.1.2) \\
K &= 1/H_{sys}(\omega_b) & (H.2.1.3) \\
\omega_b &= 2\pi \times f_b & (H.2.1.4)
\end{align*}
\]

For a position error \(e(t)\) and an input signal \(u(t)\) for the system in combination with equation H.2.1.1 up to Equation H.2.1.4 inclusive, you get the block scheme of Figure H.2.1.2.

The transfer function of a PID-controller can be written as follow:

\[
H_{PID}(s) = K \times (1 + \frac{1}{\tau_i s}) \times (1 + \tau_d s)
\]

\[
H_{PID}(s) = K \times (1 + \frac{\tau_d}{\tau_i} + K \times \tau_d s + \frac{K}{\tau_i s})
\]

PID:

\[
H_{PID}(s) = P + Ds + I/s
\]
Hence:

\[
K \times \left(1 + \frac{\tau_s}{\tau_i}\right) = P \tag{H.2.1.8}
\]

\[
K \times \tau_s = D \tag{H.2.1.9}
\]

\[
\frac{K}{\tau_i} = I / s \tag{H.2.1.10}
\]

Equations H.2.1.1 up to and including H.2.1.4 and H.2.1.8 up to and including H.2.1.10 give:

\[
P = 1.3 \times K \tag{H.2.1.11}
\]

\[
D = \frac{3 \times K}{2 \times \pi \times f_b} \tag{H.2.1.12}
\]

\[
I = \frac{2 \times \pi \times f_b \times K}{10} \tag{H.2.1.13}
\]

**H.2.2 Discrete time case**

As mentioned, you first design the controller for the continuous time case and then you examine the controller for the discrete time case. In this section the controller for the discrete time case is examined.

The output (u(n)) of a discrete time PID-controller can be represented as a function of the input (e(n)) by the following equation:

\[
u(n) = P \times e(n) + D \times (e(n) - e(n-1))/Ts + \sum_{k=-\infty}^{k=+n} e(k) \times Ts \tag{H.2.2.1}
\]

with

\[
Ts = \text{sample time} \tag{H.2.2.2}
\]

The P, I and D values calculated in the continuous time case can be used as values for the discrete time case under the restriction that the sampling frequency is about ten times the bandwidth frequency. In the low frequency range the discreet and continuous controller show the same behavior.

**H.3 Feedforward**

Feedforward is only useful when you know in what way the force to apply to the system depends in a simple linear way on a system parameter. In the case of the ADAT you know the relationship between the torque to apply and the inertia:

\[
T = J \times \ddot{\phi} \tag{H.3.1}
\]

Where \( \ddot{\phi} \) is known, first +a, then 0 and then -a. So acceleration feedforward can be used.
H.4 Filter

A filter is used in a controller to get rid of certain frequencies which will effect the system negatively. Often a lowpass filter is implemented in the system to filter out high frequencies. Otherwise these high frequencies would excitate the high-order dynamics of the system. This can be undesirable because the dynamics of the system at those frequencies are unknown.

The low pass filter that has to implemented in the case of the ADAT has to be of lowest possible order to realize the desired gain characteristic. This to minimize calculation time. The desired gain characteristic has a gain of 1 up to the cut off frequency and from the cut off frequency a decay of two decades of magnitude per decade of frequency.

The cut-off frequency of a filter is often chosen at five times the bandwidth, see Figure H.4.1.

\[ H(s) = \frac{a}{(s + b)^N} \]  

Figure H.4.1
Magnitude characteristic of a low pass filter

A N-th order filter can be represented in the s-domain by the following equation:

Where \( a/b^N \) is the gain in the pass band and where \( b \) is the cut-off frequency. A second order filter has a decay of two decades of magnitude per decade of frequency. Figure H.4.2 gives a second order filter with a gain of 1 and a cut-off frequency of 10 Hz. (N = 2, b = 10, a = 100)
This is a representation of the time continue situation, but can not be implemented in a computer routine. For filtering with a computer, digital filters are needed.

**H.4.1 Digital filters**

As mentioned the filter to implement has to be a digital filter. [5] is used as a guide for the design of the digital filter. There are two types of digital filters; recursive and non-recursive. A recursive digital filter is one whose block diagram contains one or more closed paths. The block diagram of a non-recursive digital filter contains no closed path. Due to this difference the order of a non-recursive digital filter that has the same gain characteristic as a recursive digital filter has a much higher order. A property that a recursive filter may have is that of instability. In spite of this possible property a non-recursive digital filter like a fir filter seems less suitable in this case than a recursive filter. Yet the stability of the designed filter has be taken into account.

A recursive digital second order filter can be represented as a block diagram as shown in Figure H.4.1.1.
Recursive digital second order filter

The corresponding transfer function of this recursive second order digital filter is:

\[
H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \quad (H.4.1.1)
\]

The matrices \(A = [a_0 \ a_1 \ a_2]\) and \(B = [b_1 \ b_2]\) can be determined by using Matlab routines.

It is not possible to make a filter that exactly produces the gain characteristic of an ideal lowpass filter. So filters are an approximation of the ideal lowpass characteristic.

There are several filters available with each of them having its own characteristics. In Matlab two filters are available. These are the Butterworth filter and the Chebyshev filter. The gain characteristics of N-th order Butterworth filter and Chebyshev filter are given by the expressions:

**Butterworth filter:**

\[
|H_{\text{Butterworth}}(e^{j2\pi f})| = \frac{1}{\sqrt{1 + \left(\frac{\tan \pi f T}{\tan \pi f_c T}\right)^{2N}}} \quad (H.4.1.2)
\]

For frequencies less than \(f_c\) the gain is approximately unity, for frequencies exceeding \(f_c\) the gain is close to zero.

**Chebyshev filter:**

\[
|H_{\text{Chebyshev}}(e^{j2\pi f})| \approx \frac{1}{\sqrt{1 + \varepsilon^2 V_N^2 \left(\frac{\tan \pi f}{\tan \pi f_c T}\right)}} \quad (H.4.1.3)
\]

where \(\varepsilon\) is a design parameter and \(V_N(x)\) is a Chebyshev polynomial of order \(N\).
To both filters applies that the higher the value of N, the filter order, the better the approximation to the ideal lowpass characteristic. An advantage of the Chebyshev filter is that the gain in the stopband decreases more rapidly than the gain of a Butterworth filter of the same order. A disadvantage is that the gain characteristic of the Chebyshev filter has ripple in the passband.

**H.4.2 Filter design**

The filter used in the routine to filter the Torque is a second order Chebyshev filter. A Chebyshev filter is chosen because of the fact that the gain decreases more rapidly in the stopband. So a lower order filter can be used, which reduces the computing time. Second order is sufficient, the decay from the cut off frequency is two decades of magnitude per decade of frequency.

The matrices A and B are calculated by the routine `CHEBY1`. `CHEBY1` is chosen because then the ripple can be set.

The Torque is filtered by using the filter routine in combination with storing the past two values of the Torque.

**H.5 Measurement noise & torque limitation**

Measurement noise and torque limitation have to take into account to create realistic simulations. The measurement noise depends on the encoder and the torque limitation on the motor.

**H.5.1 Measurement noise**

In the ADAT an encoder is used to measure the position of the crank. This measurement is not exactly, the real angle can lie between two increments of the encoder. You can model this measurement uncertainty with random noise between minus and plus half the resolution of the encoder with a mean of zero. The encoder of the ADAT has a resolution of 4000 increment per rotation. The measurement noise can be modeled like:

\[
\text{measurement} = \text{real value} + \text{resolution} \times (\text{rand} - 0.5)
\]  

(H.5.1.1)

**H.5.2 Torque limitation**

The torque that can be offered depends on the motor that is used in the ADAT. This motor can not supply any torque. The torque of the motor in the ADAT, an Electro Craft S-243-1A, is limited at 1.7 Nm. The torque in the simulation also has to be limited at 1.7 Nm to create a realistic simulation.
H.6 Stability

A criterion whether a linear system is stable or not is the nyquist criterion [6]. The designed PID controller with filter in combination with the ADAT is stable according to this criterion, the point -1 is not enclosed by the openloop response of the PID controller with the ADAT. Feedforward influences the stability not according to [7]. The filter is the only element that could cause instability, but the designed filter has no instability. The measurement noise does not influence the stability. The torque limitation also does not influence the stability, this only can enlarges settling time.

H.7 Settings

H.7.1 PID

The design of the PID-values is not is easy as describes, because you have no constant frequency response of the system. So you have to try PID-values for several inertia, see Figures H.7.1.1 up to and including H.7.1.5.

Figure H.7.1.1
\[ J = 2.1 \times 10^{-4} \text{ Nm s}^2 \]
\[ P = 9.4, I = 354.0 & D = 0.075 \]

Figure H.7.1.2
\[ J = 5.8 \times 10^{-4} \text{ Nm s}^2 \]
\[ P = 33.7, I = 977.8 & D = 0.21 \]
The BID values with the shortest settling time are; $P = 33.7$, $I = 977.8$ and $D = 0.21$. If there were no other elements this would be the ideal values. But when you combine the PID controller with the filter, you get a stable system but the deviation from the pick and place position is too large. The best behavior is wanted at the pick and place positions so the PID-values are determined for the inertia corresponding with these positions. These values give a satisfactory result in combination with the filter, see filter.

**H.7.2 Acceleration feedforward**

The design of the acceleration feedforward has the same problem as for the PID values, the frequency response is not constant. Too large feedforward causes overshoot too much overshoot, that enlarges the settling time and too small feedforward gives a too large settling time, see Figure H.7.2.1 up to and including H.7.2.6. The PID-values are $P = 33.7$, $I = 977.8$ & $D = 0.21$. 

---

**Figure H.7.1.3**

$J = 7.7 \text{ e-4 Nm s}^2$

$P = 44.8, I = 1298.1 \& D = 0.27$

**Figure H.7.1.4**

$J = 9.6 \text{ e-4 Nm s}^2$

$P = 55.8, I = 1618.5 \& D = 0.34$

**Figure H.7.1.5**

$J = 13.9 \text{ e-4 Nm s}^2$

$P = 80.8, I = 2343.4 \& D = 0.49$
Figure H.7.2.1
\( J = 2.1 \times 10^{-4} \text{ Nm}^2 \)

Figure H.7.2.2
\( J = 3.0 \times 10^{-4} \text{ Nm}^2 \)

Figure H.7.2.3
\( J = 3.9 \times 10^{-4} \text{ Nm}^2 \)

Figure H.7.2.4
\( J = 4.8 \times 10^{-4} \text{ Nm}^2 \)

Figure H.7.2.5
\( J = 5.8 \times 10^{-4} \text{ Nm}^2 \)

Figure H.7.2.6
\( J = 9.6 \times 10^{-4} \text{ Nm}^2 \)
The optimum for the feedforward is found for the following setting, $J_{\text{feedforward}} = 3.9 \times 10^{-4}$ Nm s$^2$, when you use a tolerance on the position of 0.1 degree, see Figure H.7.2.3. Acceleration feedforward reduces the settling time and does not effect the stability.

**H.7.3 Filter**

The filter is set at 300 Hz. Now we check if the filter is stable and meet the requirements. The requirements are that the angle differs with a maximum of 0.1 radian from the extreme positions so that the tip stays within the tolerance. You want to realize this behavior for the extreme positions, so the chosen PID values are for the PID values corresponding with $J_{\min}$. $P = 9.4$, $I = 354.0$ & $D = 0.075$. The settling time can be reduced by choosing a suitable feedforward. This settings must be determined when measurement noise and torque limitation are added.

**I.7.4 Measurement noise and torque limitation**

The behavior of the ADAT with measurement noise stays within the admissibility, see Figure H.7.4.1. The noise is $0.1 \times (\text{rand} - 0.5)$.

![Graph showing position of the crank with measurement noise](image)

**Figure H.7.4.1**

Position of the crank with measurement noise

The torque limitation enlarges the settling time, see Figure H.7.4.2. The torque is limited at 1 Nm, see Figure H.7.4.3.
Torque limitation does not effect the stability but enlarges the settling time. The values of the torque become larger when the feedforward or PID values are too large. The larger the difference between calculated torque and the torque limitation the more the settling time enlarges.

**H.7.5 Definitive settings**

The PID values are the values for the extreme positions. The filter, torque limitation and measurement noise are extern determined. The only parameter to set is the feedforward. You can choose the feedforward equal to the $J_{min}$, but when you chose the feedforward so large that there is no overshoot you have the shortest settling time. The definite settings are as follows:

\[
\begin{align*}
    f_{b,\text{PID}} &= 60 \text{ Hz} \\
    P &= 9.4 \\
    I &= 354.0 \\
    D &= 0.075 \\
    J_{\text{feedforward}} &= 2.6 \times 10^{-4} \text{ Nm s}^2 \\
    \text{filter} &= \text{at 300 Hz} \\
    \text{noise} &= 0.1 \times (\text{rand} - 0.5)
\end{align*}
\]

These settings give the results as shown in chapter 4.
H.8 Matlab-files

The following Matlab files are added:

- `wm281101.m`: controller, filter, noise, torque limitation
- `cc2811.m`: calculates the time that corresponds to the begin position
- `h2id.m`: determines PID values
- `filt.m`: determines A and B matrices of the Chebyshev filter

H.8.1 WM281101.M

function Tf = wm281101(phi, t)

% This is a routine used in combination with Working Model.

% Pay attention to:
% - P, I, D and J values
% - dt (fsample) and dtt (trace)
% - cc28111.m
% - load trac2000.mat (dtt)
% - WM -> correct .wm
% -> input and output

% Set parts of controller on/off

% Set filter
  Sf = 1;
% Set Torque max
  St = 1;
% Set Feedforward
  Sff = 1;
% Set measure noise
  Sn = 1;
% Set trajectory
  Sp = 0;

% Define variables

global mn
global n1
global x

global xref

global Y

global dtt

global M

global e2

% Parameters

%
\[ dt = 0.0005; \quad \% \text{related to fsample} \\
\text{dtt} = 0.0005; \quad \% \text{sample time of trace}____.mat \\
\]

\[ P = 9.4; \quad \% \text{To determin PID use H_tot with L} = 0; \\
I = 354.0; \\
D = 0.075; \\
\]

\[ \text{max} = 1.7; \quad \% \text{Maximum Torque} \\
J = 2.6e-4; \\
\]

\[ \text{dr} = \pi/180; \quad \% \text{Factor for going from degrees to radians} \\
\text{pos} = 30; \quad \% \text{Position when trace is of} \\
\]

\% Pick and place time
\[ p = 300e-3; \]

\% New time between 0 and 280 ms
\[ \text{ttr} = t; \]
\[ \text{while } \text{ttr} \geq (210e-3 + 2*p) \]
\[ \text{ttr} = \text{ttr} - (210e-3 + 2*p); \]
\[ \]
\% Load trajectory
\[ \text{if } t == 0 \]
\[ \text{load trac2000.mat ;} \]
\[ x = (x + 30); \quad \% \text{correct to angle in WM} \\
\text{nn} = 0; \quad \% \text{initialize move ; pick -> place} \\
M = [0 0 0]; \quad \% \text{initialize memory matrix for filtering} \\
e = 0; \\
e1 = 0; \\
e2 = 0; \]
\[ \text{end} \]

\% Choose trajectory
\[ \text{if } \text{Sp} == 1 \]

\% New value phik (between -90 and 270)
\[ \text{while phik < -60} \]
\[ \text{phik} = \text{phik} + 360; \]
\[ \text{end} \]
\[ \text{while phik} > 300 \]
\[ \text{phik} = \text{phik} -360; \]
\[ \text{end} \]

\% Calculate \( r \); \( r \) = number for xref
if nn == 0
    r = ttr/dtt + 1;
else
    rt = ttr;
    rt = rt - (105e-3 + p);
    r = rt/dtt + 1;
end

% Initialize t
if t == 0
    nt = cc2811(phik);  % Calculation of time corresponding with begin
end

% Trajectory; define reference for 0 < t < 210 + 2p
if ttr > 105e-3
    if ttr <= (105e-3 + p)
        xref = 210;
        a = 0;
        nn = 1;
    else
        if ttr >= (210e-3 + p)
            xref = 30;
            a = 0;
            nn = 0;
        else
            xref = (210 - x(r,1) + 30);
            a = -xpp(r,1);
        end
    end
else
    xref = x(r,1);
    a = xpp(r,1);
end

% Generate measure error on angle
if Sn == 1
    phik = phik + 0.1*(rand - .5);
end

% Reference is noise if Sp = 0 ; .5 degrees = 8.8e-3 rad
else
    xref = pos + .1*(rand-0.5)
    a = 0;
    nt = 0;
end
end

% Compute error
\[ e = (xref - phik) \times dr; \]
\[ e2 = e2 + e; \]

% Compute Torque
\[ \text{if } Sf \equiv 1 \]
\[ Kfa = a^*J^*dr; \]
\[ \text{else} \]
\[ Kfa = 0; \]
\[ \text{end} \]

\[ \text{if } (t-nt) < -2*dt \]
\[ T = 0; \]
\[ u = 0; \]
\[ v = 0; \]
\[ w = 0; \]
\[ \text{else} \]
\[ u = P*e; \]
\[ v = D*(e-e1)/dt; \]
\[ w = I*e2*dt; \]
\[ T = u + v + w + Kfa; \]
\[ \text{end} \]

% Filter
\[ \text{if } Sf \equiv 1 \]
\[ T1 = \text{filter}(B,A,M); \]
\[ T1 = T1(1,3); \]
\[ \text{else} \]
\[ T1 = T; \]
\[ \text{end} \]

% Motor restriction

% Filter matrix for filtering
\[ B = \begin{bmatrix} 0.2771 & 0.5542 & 0.2771 \end{bmatrix}; \]
\[ A = \begin{bmatrix} 1.0000 & -0.0910 & 0.2123 \end{bmatrix}; \]

% Memory matrix for filtering
\[ M = \begin{bmatrix} T & M(1,1) & M(1,2) \end{bmatrix}; \]

% Filtered Torque
\[ \text{if } Sf \equiv 1 \]
\[ T = \text{filter}(B,A,M); \]
\[ T = T(1,3); \]
\[ \text{else} \]
\[ T = T; \]
\[ \text{end} \]

% Motor restriction
if St == 1
    if abs(Tf) > max
        Tf = sign(Tf)*max;
    end
end

% Store interesting variables

e1 = c;
E = [E e];
Tg = [Tg t];
Y = [Y phik];
tr = [tr xref];
U = [U Tf];
K = [K T];
function nt = cc2811(phik)

    global tt
    global x
    global dt
    global dtt
    global xpp

    % Phik = angle of motor-arm
    % TF = ending time of the simulation
    % Animation interval-time

    TF = 105e-3;
    i = 1;
    tol = 1e-3;
    dtt = .0005;

    % Redefine phik between -90 and 270

    while phik < -90
        phik = phik + 360;
    end
    while phik > 270
        phik = phik - 360;
    end

    % Starts SIMULINK to calculate trajectory

    load trac2000.mat

    % Calculate error in position

    x = x + 30;
    err = x - phik;

    % Search for location minimum value

    n = min(abs(err));

    while abs(err(i,:)) == n
        i = i + 1;
    end

    % Calculate new time

    vt = tt(i,1);

    f = dt/dtt;
    z = i/f;
zz = z;

s = 0;
while zz > 1
    zz = zz - 1;
    s = s + 1;
end

if zz < 0.5
    s = s - 1;
else
    s = s;
end

nt = s*dt;
H.8.3 H_PID.M

%f Frequency response of a PID-controller

function [H, phase, f] = H_pid(fb, Ks, dt, N);

%f Calculation of w-bandwith

wb = 2*pi*fb;

%f Calculation of Ti and Td

Ti = 1/(wb/10);
Td = 1/(wb/3);

%f Calculation of K ( |K(fb)*Ks(fb)| = 1 )

%a = (1+(Td/Ti)) + ((1/Ti)*(1/wb)) + (Td*wb)

a1=(1/(Ti*wb))-Td*wb
a2=(1+(Td/Ti))
A=sqrt(a1^2 + a2^2)
Ks
K=1/(A*Ks)

%f Calculation of P, I and D

P = (1+(Td/Ti))*K
I = (1/Ti)*K
D = Td*K

H.8.4 FILT.M

%f Calculate A and B matrices for filtering with cheby1-filter

fb = 60; % Cut off on 5 * fb
fs = 2000;

N =2; % Order
wn = 5*fb/(.5*fs); % Cut off
r = .1; % Ripple

[B, A] = cheby1(N, r, wn)