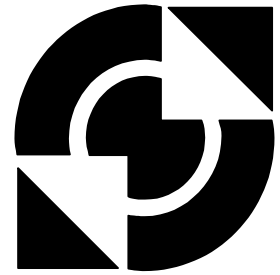


University of
Twente



Faculty of Electrical Engineering,
Mathematics & Computer Science

Design and Realization of a Safe Control System for a Parallel Manipulator

M. Eglence

M.Sc. Thesis

Supervisors: prof. dr. ir. J. van Amerongen
dr. ir. T.J.A. de Vries
dr. ir. J.F. Broenink
ir. B.J. de Kruif

June 2003
Report Number
010CE2003

Control Laboratory
Faculty of Electrical
Engineering,
Mathematics &
Computer Science

University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

The Control Laboratory at University of Twente has purchased an Imotec *xyz*-manipulator. The manipulator is delivered with an open operating system. This thesis treats the design and realization of a safe guarded controller system for the Imotec manipulator. Safety issues are discussed to make the manipulator operate safely for doing research experiments by students and staff. This concerns safety of the manipulator itself and of the people working with it. The first experiments will be concerned with the research on Learning Feed Forward Control (LFFC).

The safe guarded controller system is implemented as a Multi Agent Controller (MAC) system. The controller is tested in simulation (20-Sim) and has proven to work correctly.

The designed controller has largely been implemented on the real system and has been found to work according to expectations. For a slow, large stroke movement, a maximum tracking error of 250 [μm] was found at moments of velocity reversal; otherwise, the max tracking error amounted 100 [μm].

Because of time constraints, the experiments with LFFC have not been carried out.

A path generator tool has been developed that can be used for creating path for manipulators in general. It creates curves and straight movements in (x,y,z) . With the tool a reference motion can be given up in segments from which afterwards a file is created which holds the reference points for the sample times.

Acknowledgements

Thanks go out to a lot of people who made it possible for me to graduate. First of all, I would like to thank my family for their never-ending support and encouragement.

I would like to thank my supervisors who gave me the opportunity to work on this interesting thesis. Especially I would like to thank Theo de Vries for his valuable guidance and tips during the thesis, the good time we had at Imotec and for the great fun we had with testing the robot.

Herman, Jan, Peter and Richie thanks for the good time and interesting discussions at Imotec.

Job van Amerongen I would like to thank for the opportunity to graduate at the Control Laboratory.

Last but certainly not least I would like to thank the lady that made my life so much nicer by stepping in to it. Sabire, thanks for your love and encouragement.

Enschede, June 2003

M. Eglence

Table of contents

1	INTRODUCTION	1
1.1	BACKGROUND	1
1.2	LEARNING FEED-FORWARD CONTROL	1
1.3	THE IMOTEC MANIPULATOR	2
1.4	THE ASSIGNMENT	4
1.5	THESIS STRUCTURE	4
2	DESIGN CONCERNING SAFETY ISSUES.....	5
2.1	INTRODUCTION.....	5
2.2	MALFUNCTIONING OF THE MANIPULATOR.....	5
2.2.1	CHECKING THE MECHANICAL STRUCTURE.....	5
2.2.2	CHECKING THE POWER SUPPLY	12
2.2.3	CHECKING THE LINEAR MOTORS.....	12
2.2.4	CHECKING THE MOTOR AMPLIFIERS	12
2.2.5	CHECKING THE INTERFACE CARDS	12
2.2.6	CHECKING THE COMPUTER SYSTEM.....	13
2.3	FAULTS BY THE ENVIRONMENT.....	14
2.4	APPLIED CHECKS AND THEIR RESPONSES	14
3	IMPLEMENTATION OF THE SAFE CONTROLLER SYSTEM.....	15
3.1	INTRODUCTION.....	15
3.2	POSSIBLE IMPLEMENTATION METHODS OF THE SAFE CONTROLLER SYSTEM	15
3.3	THE CONCEPT OF MAC SYSTEMS	16
3.3.1	CONTROLLER AGENTS.....	16
3.3.2	SENSOR AGENTS.....	17
3.3.3	ACTUATOR AGENTS.....	17
3.4	COORDINATION OF AGENTS	17
3.5	MAC SYSTEM FOR THE MANIPULATOR.....	18
3.5.1	STARTUP AGENT.....	19
3.5.2	ALARM AGENT	20
3.5.3	GUARDED EMERGENCY AGENT	20
3.5.4	GUARDED STANDARD AGENT	20
3.5.5	MODE SWITCH CONTROLLER AGENT	21
3.6	TOTAL OVERVIEW OF THE MAC SYSTEM	23
4	PATH GENERATION.....	25
4.1	INTRODUCTION.....	25
4.2	TOOLS FOR PATH SPECIFICATION	25
4.3	THE PATH GENERATOR	26
4.4	TRAPEZOIDAL PROFILE ALGORITHM	28
4.5	S-CURVE PROFILE ALGORITHM	30
4.6	STRAIGHT LINE MOVEMENT	32
4.7	ARC MOVEMENT.....	33
4.8	ASSIGNING THE SAMPLE REFERENCE POINTS TO THE ARC MOVEMENT	34
5	SIMULATIONS AND EXPERIMENTS	37

5.1	INTRODUCTION	37
5.2	DESIGN OF DEMONSTRATOR PATH	37
5.3	THE 20-SIM MODEL OF THE MANIPULATOR	38
5.4	VERIFYING THE STANDARD MODE IN SIMULATION	39
5.5	VERIFYING THE ERRORGUARD	41
5.6	VERIFYING THE EMERGENCY SITUATION IN SIMULATION	42
5.7	VERIFYING THE ALARM MODE IN SIMULATION	43
5.7	EXPERIMENTAL RESULTS	44
6	CONCLUSIONS AND RECOMMENDATIONS	47
6.1	CONCLUSIONS	47
6.2	RECOMMENDATIONS	48
	APPENDIX A: WORKING PRINCIPLE AND DRIVE OF LINEAR MOTOR	49
	APPENDIX B: HARDWARE OVERVIEW OF THE MANIPULATOR	51
	APPENDIX C: AN INTRODUCTION TO XML	53
	APPENDIX D: CONTROLLER SETTINGS	57
	APPENDIX E: MANIPULATOR MAC SYSTEM CODE	59
	BIBLIOGRAPHY	113

1 Introduction

1.1 Background

At the Control Engineering Laboratory of the University of Twente (UT) research is done on Mechatronic Systems in general and in particular on the role of the controller in it. An ongoing project is concerned with Learning Feed-Forward Control (LFFC) [De Kruif, 2003]. To carry out experiments for this research, an *xyz*-manipulator with parallel kinematical configuration has been purchased by the UT. The manipulator has been developed and built by the company Imotec, which is also a sponsor of the research. The manipulator has been delivered with an open controller system, in order to make it possible to implement advanced control and identifications algorithms. The first research on the manipulator will be concerned with the application of modern function approximators in closed loop control. When doing research, people will work closely with the manipulator. Therefore, safety is an important issue. The manipulator should operate in a safe manner without causing danger for people working with it and without causing damage to itself.

1.2 Learning Feed-Forward Control

For obtaining good performances with classical control systems, the parameters of the controlled plant need to be known well. Not knowing the plant parameters accurate enough result in not entirely knowing the dynamics of the plant. In many control problems, the plant is given as a model of which the parameters are not known exactly. The reasons for this are various, for instance:

- Low-precision production processes make the fabricated part to differ from the specification.
- Manufacturing tolerances lead to a spread in dynamic behaviour.
- Complexity of the plant makes parameter estimation difficult.
- Changes in plant characteristics as time proceeds.
- Non-linear effects like friction and cogging.

In many applications the goal is to come to a high precision servo system with low price components. In these situations the classical feedback controller demerits. However, the principle of LFFC can offer a solution [Velthuis, 2000],[Starrenburg *et al*, 1995]. In figure 1.1 a control loop is given in which a learning component is included in the feed-forward path. In this case, this component is a function approximator (FA).

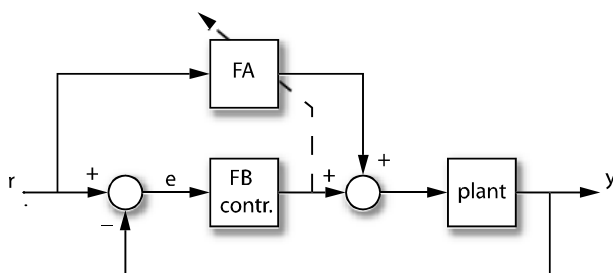


Fig. 1.1 A control loop with function approximator.

The function approximator realizes an input-output mapping gained by experience. The mapping summarizes the examples by a function. This function for instance can be the force that is needed to compensate for the friction loss depending on the velocity of the mechanical setup. In the depicted control loop, the feedback controller is needed for stability of the closed loop system and to present samples for the learning mechanism of the function approximator. The function approximator approximates the inverse dynamics of the plant based on the output of the feedback controller. After learning, it compensates for the non-linear state dependent behavior of the plant.

1.3 The Imotec manipulator

The Imotec manipulator, of which a sketch is given in figure 1.2, is a simplified Stewart platform with three degrees of freedom namely x , y and z . It is driven by three Tecnotion linear motors in vertical direction. For working principle and drive of linear motors, see Appendix A. For an overview of the manipulator hardware see Appendix B.

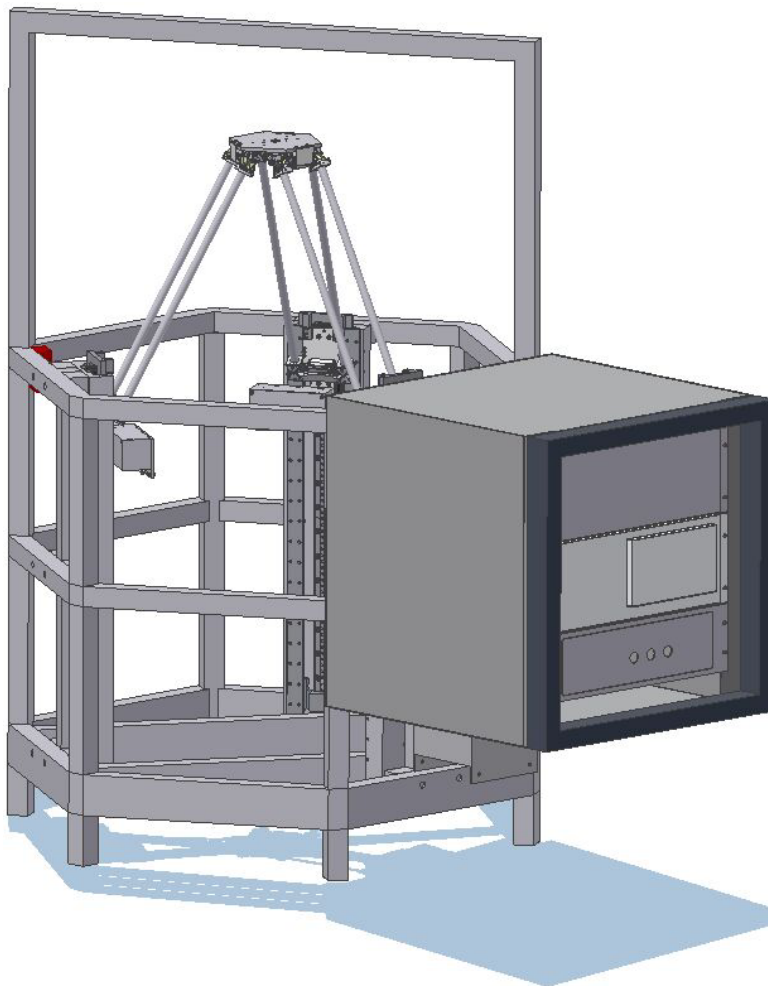


Fig. 1.2 The Imotec xyz- manipulator.

A set of arms in parallelogram construction attached to the translators of the motors, holds a platform, which is the end-effector of the manipulator. The joints of the arms allow movement only in φ and θ direction, whereas the translators only move in z direction (see figure 1.3). This altogether results in an x, y, z motion of the platform. The parallelogram construction of the arms restricts rotational movements of the platform. The three arms together also keep the platform in the horizontal plane.

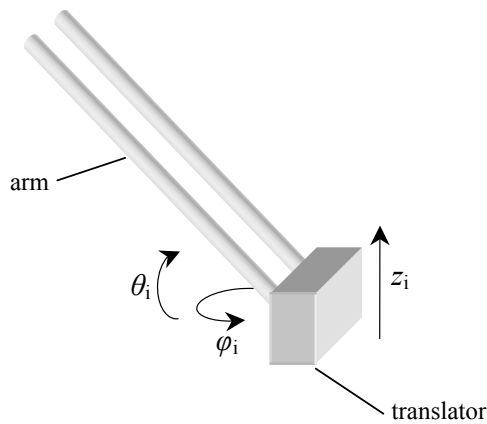


Fig. 1.3 Joint motion of the parallelogram constructed arms.

The manipulator has a safe work area that is shaped as a cylinder with radius 200 [mm] and height 250 [mm]. Although the end-effector of the manipulator can exceed the dimensions of the cylinder, it is not recommended because this will cause overloading of the leaf springs from the joints. Specifications of the manipulator are:

- Max Payload 5 [kg]
- Max Speed 1 [m/s]
- Max Acceleration 30 [m/s²]
- Max Stroke Lin. Motor 520[mm]

The manipulator is given in its functional blocks in figure 1.4. The setup can be divided in three major parts:

- The computing system
- The electrical circuitry and components
- The mechanical setup.

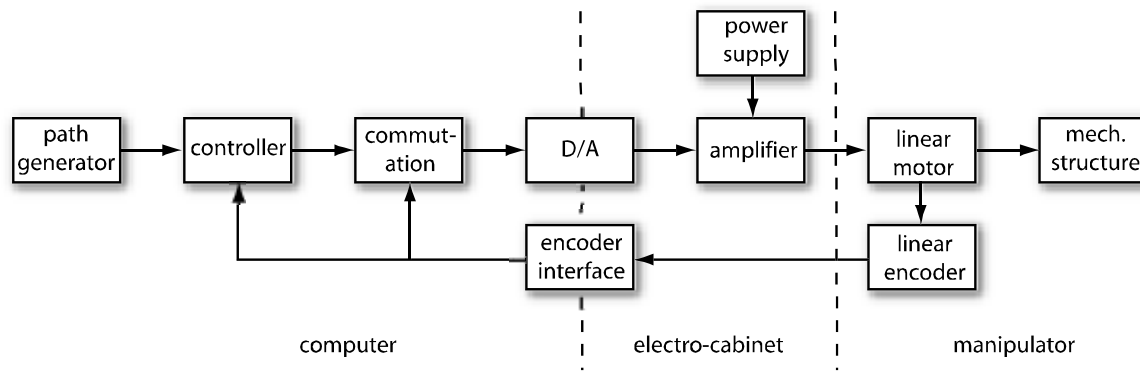


Fig. 1.4 The manipulator in its functional blocks.

1.4 The assignment

The electrical circuitry and mechanical setup of the manipulator have been realized and tested. To become operational, the manipulator's computing system needs to be programmed. This means that the right actions need to be taken depending on the signals from the electro-cabinet and on user input. The objective of the assignment is to develop and realize a safe control system. This incorporates the following points:

- ✓ Design and realization of a safeguard system to make it possible to carry out experiments with the manipulator without danger for the environment and for the setup itself.
- ✓ Design of a generally applicable tool for path specification of manipulators.
- ✓ Implementation of the path specification tool for the parallel kinematical *xyz*-manipulator, including setpoint generation.
- ✓ Implementation of a relatively simple closed loop control system in which a modern function approximator is integrated.

1.5 Thesis Structure

After this introduction, in chapter 2 safety issues concerning the manipulator will be discussed. In chapter 3 these safety issues will be implemented for the Imotec manipulator. In chapter 4, path generation in general for manipulators will be discussed. The path generator specific for the *xyz*-manipulator will also be developed in this chapter. In chapter 5 simulations and experimental results will be discussed. Finally in chapter 6, the conclusions and recommendations are presented.

2 Design concerning safety issues

2.1 Introduction

Safety concerns two major aspects in manipulators. First and most important is the safety of people operating and working with the manipulator. Second, the manipulator should not damage itself by some motion. Usually manipulators are placed in industrial environments where people do not work closely nearby the manipulator. In case of the Imotec manipulator, people will work close to the manipulator when doing research. Therefore considerable attention will be paid to human safety. The manipulator is used for doing research in the field of control engineering. It is imaginable that a bad controller setting (unstable) can cause unsafe situations. This chapter deals with the measures that can be taken to make the manipulator operate safely. There are two major causes that can lead to dangerous situations in case of the manipulator.

- Malfunctioning of a component of the manipulator itself.
- Irresponsible behavior of people in the environment.

First the possible faults of the manipulator itself will be discussed and then faults that can arise by the environment. Measures to handle these situations will also be discussed.

2.2 Malfunctioning of the manipulator

The Imotec manipulator consists of six basic parts, which can all lead to faults:

- The mechanical structure
- The linear motors
- The motor amplifiers
- The power supply
- The interface cards
- The computer system

These parts will be discussed subsequently in the next sections.

2.2.1 Checking the mechanical structure

The mechanical structure of the manipulator is difficult to check automatically during operation. A method, which is the most feasible in general, is to compare the response of the manipulator with a proper mathematical model [Van De Mast, 1992]. In case of the Imotec manipulator this is not applicable for the moment because the position of the end-effector is not measured. Only the positions of the translators are measured by means of the linear encoders. For instance, if an arm would break, this will not influence the position of the translator much, whereas the platform (end-effector) will be positioned totally different. For the moment, no check for mechanical structure integrity is done. A feasible measure is to avoid situations that can cause

the mechanical structure to break apart. This can be done by assuring that the manipulator operates in its safe work area. The encoder readings from the linear motor can be used to determine the position of the platform. This means that the forward kinematics of the manipulator has to be known. The Imotec manipulator has complex forward kinematics due to its parallel configuration. However, the inverse kinematics can be derived easily. When z_1 , z_2 and z_3 are the positions of the translators in z direction, then the following holds:

$$\begin{aligned} z_1 &= -\sqrt{l^2 - (x - x_1)^2 - (y - y_1)^2} + z + z_o \\ z_2 &= -\sqrt{l^2 - (x - x_2)^2 - (y - y_2)^2} + z + z_o \\ z_3 &= -\sqrt{l^2 - (x - x_3)^2 - (y - y_3)^2} + z + z_o \end{aligned} \quad [2-1]$$

Where l is the known length of the arms and x , y , z are the coordinates of the moving platform. x_i is the x position of the translator i and y_i its y position. Note that the translators only move in z direction, therefore x_i and y_i are fixed and known. z_o is the initial height of the platform in z direction when the translators are all in the bottom position ($z_i = 0$, $i = 1,2,3$). z_o is calculated as:

$$z_o = \sqrt{l^2 - r^2} \quad [2-2]$$

Where r is the known radius of the circle in which the translators are aligned, see figure 2.1.

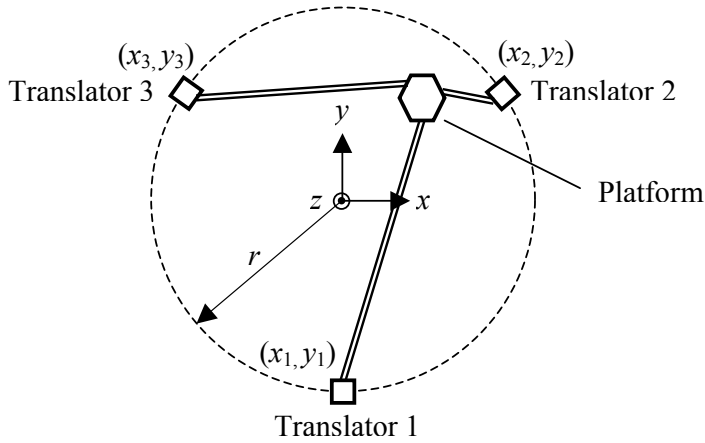


Fig. 2.1 Orientation and setup of the translators.

Equations [2-1] represents the inverse kinematics of the manipulator. We are interested in the forward kinematics, i.e., we wish to regard [2-1] as three simultaneous equations with three unknowns, (x, y, z) . However, this problem cannot be solved explicitly because of its complexity; the root function in [2-1] makes this hard. Therefore, an approximation is done [De Kruif], to remove the root function. First a transformation is done from Cartesian coordinates to cylindrical coordinates by using the following coordinate change:

$$\begin{aligned} x &= p \sin(\phi) \\ y &= p \cos(\phi) \\ z &= z \end{aligned} \tag{2-3}$$

Where p is the distance between the origin and the position of the platform in the x - y plane. As stated earlier the manipulator has a safe work area of cylindrical shape with radius 200 [mm] and height 250 [mm]. Therefore the distance p may not exceed 200 [mm]. Hence, we can reformulate the safety check as solving [2-1] and [2-3] for p and checking it for the given limit. Using the coordination transform of [2-3] in [2-1] results in:

$$\begin{aligned} z_1 &= -\sqrt{l^2 - r^2 - p^2 - 2pr \cos(\phi)} + z + z_o \\ z_2 &= -\sqrt{l^2 - r^2 - p^2 - 2pr \cos(\phi + \frac{2}{3}\pi)} + z + z_o \\ z_3 &= -\sqrt{l^2 - r^2 - p^2 - 2pr \cos(\phi + \frac{4}{3}\pi)} + z + z_o \end{aligned} \tag{2-4}$$

We introduce an intermediate function:

$$f(p, \phi) = \sqrt{l^2 - r^2 - p^2 - 2pr \cos(\phi)} \tag{2-5}$$

In [2-4] the root function $f(p, \phi)$ is still present. A plot thereof is given in figure 2.2.

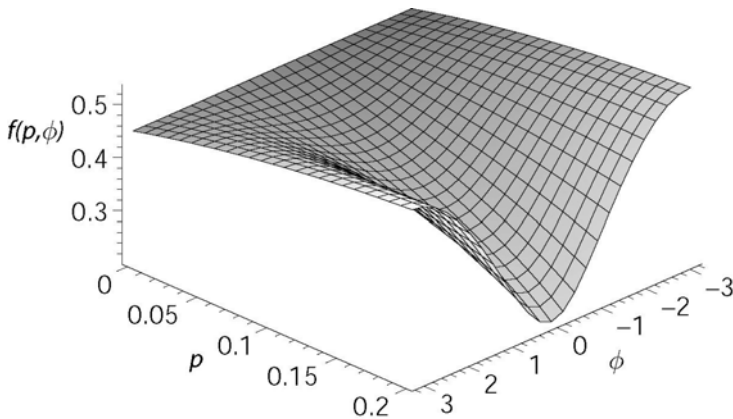


Fig. 2.2 A plot of the root function, $f(p, \phi) = \sqrt{l^2 - r^2 - p^2 - 2pr \cos(\phi)}$.

From the plot can be seen that $f(p, \phi)$ has its minimum and maximum on the line where $p = p_{\max} = 200$ [mm]. With $l = 541$ [mm], $r = 300$ [mm] and $\phi = [-\pi, \pi]$ given, we can derive:

$$0.20659 \leq f(p, \phi) \leq 0.53167 \quad [2-6]$$

$f(p, \phi)$ can be converted into $f(x)$ by stating:

$$f(p, \phi) \hat{=} f(x) = \sqrt{x} \quad [2-7]$$

with

$$x = l^2 - r^2 - p^2 - 2pr \cos(\phi) \quad [2-8]$$

and

$$x \in [x_{\min}, x_{\max}] = [0.20659^2, 0.53167^2] = [0.0427, 0.2827] \quad [2-9]$$

A plot of $f(x)$ is given in figure 2.3.

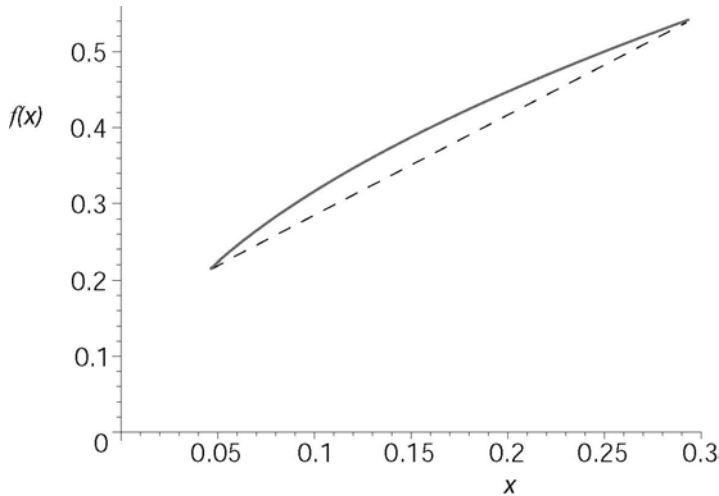


Fig. 2.3 The root function $f(x)$; the dashed line is the linearization of it.

By doing a linearization, $f(x)$ can be approximated. The approximation is given by the dashed line in figure 2.3. This line is described by:

$$f(x) \approx \tilde{f}(x) = f(x)_{\min} + \frac{f(x)_{\max} - f(x)_{\min}}{x_{\max} - x_{\min}} (x - x_{\min}) \quad [2-10]$$

And with the known values filled in:

$$\tilde{f}(x) = 1.355[l^2 - r^2 - p^2 - 2pr \cos(\phi)] + 0.149 \quad [2-11]$$

Substituting equation [2-10] as approximation for $f(p, \phi)$ into equations [2-4] gives:

$$\begin{aligned} z_1 &\approx -1.355[l^2 - r^2 - p^2 - 2pr \cos(\phi)] + z + z_o - 0.149 \\ z_2 &\approx -1.355\left[l^2 - r^2 - p^2 - 2pr \cos\left(\phi + \frac{2}{3}\pi\right)\right] + z + z_o - 0.149 \\ z_3 &\approx -1.355\left[l^2 - r^2 - p^2 - 2pr \cos\left(\phi + \frac{4}{3}\pi\right)\right] + z + z_o - 0.149 \end{aligned} \quad [2-12]$$

Adding up the equations in [2-12] results in (note that the cosine parts add up to zero):

$$z_1 + z_2 + z_3 \approx -3 \cdot 1.355(l^2 - r^2 - p^2) + 3z + 3z_o - 0.447 \quad [2-13]$$

From which it follows that:

$$z + z_o \approx 1.355(l^2 - r^2 - p^2) + \frac{z_1 + z_2 + z_3}{3} + 0.149 \quad [2-14]$$

Combining the first equation of [2-12] with [2-14] gives:

$$z_1 \approx -1.355[l^2 - r^2 - p^2 - 2pr \cos(\phi)] + 1.355(l^2 - r^2 - p^2) + \frac{z_1 + z_2 + z_3}{3} \quad [2-15]$$

From which for z_1 is obtained:

$$z_1 \approx 4.065pr \cos(\phi) + \frac{z_2 + z_3}{2} \quad [2-16]$$

Equally for z_2 the following can be derived:

$$z_2 \approx 4.065pr \cos\left(\phi + \frac{2}{3}\pi\right) + \frac{z_1 + z_3}{2} \quad [2-17]$$

The equations [2-16] and [2-17] can be generalized into:

$$\begin{aligned} \alpha &= A \cos(\phi) \\ \beta &= A \cos\left(\phi + \frac{2}{3}\pi\right) \end{aligned} \quad [2-18]$$

For which a general solution can be found with the use of Maple as:

$$A = \sqrt{\frac{4}{3}\alpha^2 + \frac{4}{3}\beta^2 + \frac{4}{3}\alpha\beta} \quad [2-19]$$

With $A = 4.065 pr$, $\alpha = z_1 - \frac{z_2 + z_3}{2}$ and $\beta = z_2 - \frac{z_1 + z_3}{2}$, the following expression is found for the approximation of the radius:

$$p_{appr} = \frac{0.246}{r} \sqrt{z_1^2 - z_1 z_2 - z_1 z_3 + z_2^2 - z_2 z_3 + z_3^2} \quad [2-20]$$

In figure 2.4 a plot is given of the actual radius of the platform and the approximation of it according to [2-20].

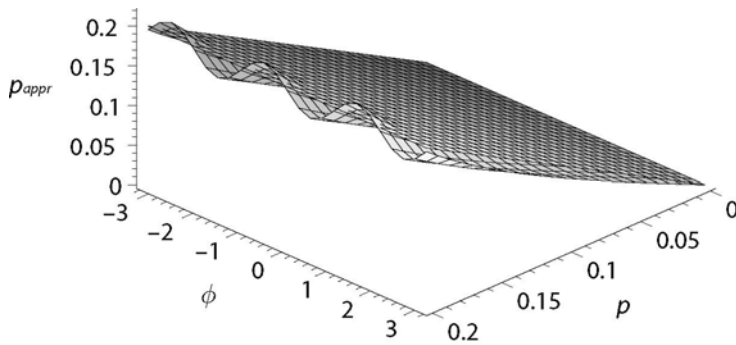


Fig. 2.4 Approximation of the radius according to [2-20].

From the plot can be seen that the approximation that is made, varies between 175 [mm] and 220 [mm] depending on ϕ for $p = 200$ [mm]. The error that is made is given in figure 2.5.

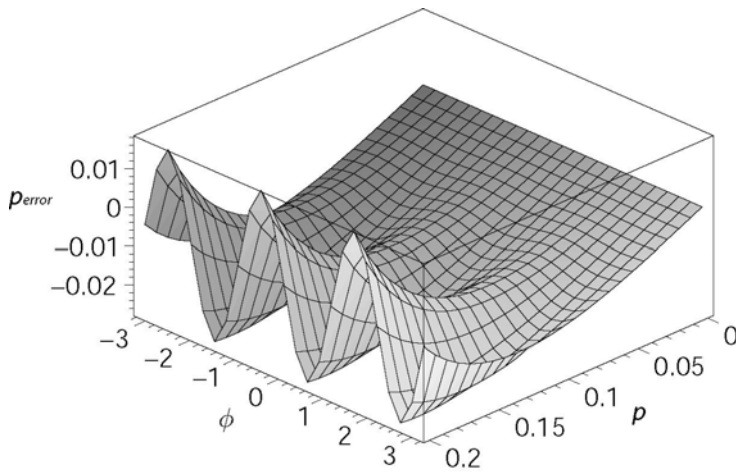


Fig. 2.5 The error that is made with the approximation of the radius according to [2-20].

The threshold value of the safety system to take action should be set at 170 [mm]. The platform then can move up to a radius of 200 [mm] depending on ϕ .

Also the z direction of the platform has to be checked to ensure that the platform does not exceed the cylindrical work area. The height of the platform has to be determined the most accurate in the position where it is at the edge of the radius of his work area, that means $p \approx 200$ [mm] and $z \approx 0$ or $z \approx 250$ [mm]. In this positions the spring leaves of the joint are bended the most.

With equation [2-13] and the approximation for the radius [2-20] the following can be stated:

$$z_1 + z_2 + z_3 = -3 \cdot 1.355(l^2 - r^2 - p_{appr}^2) + 3z + 3z_o - 0.447 \quad [2-21]$$

With the known values filled in, this results in the approximation for checking the z direction of the platform:

$$z_{appr} = \frac{z_1 + z_2 + z_3}{3} + 1.355(l^2 - r^2 - p_{appr}^2) - z_o + 0.149 \quad [2-22]$$

In figure 2.6 a plot is given of the actual z -position of the platform and the approximation of this position according to [2-22].

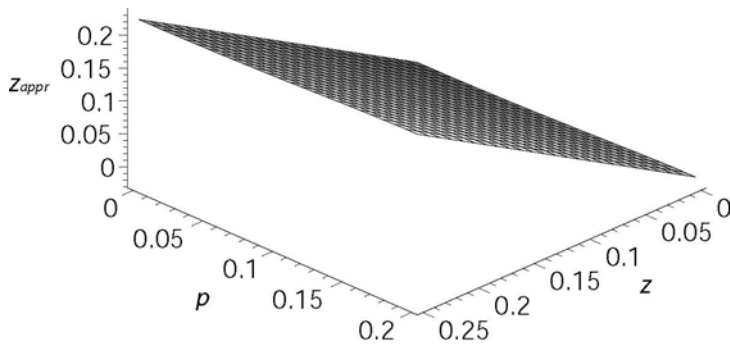


Fig. 2.6 Approximation of the z -position according to [2-22].

From the plot can be seen that the approximation for z -position makes an under-estimation. The error that is made is given in figure 2.7.

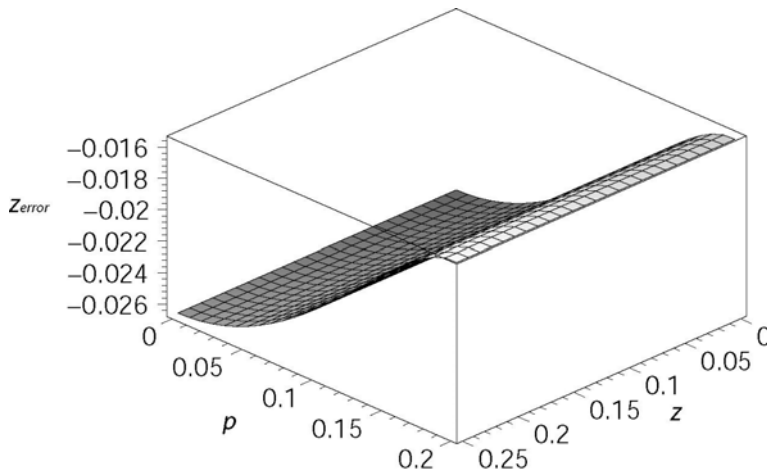


Fig. 2.7 The error that is made with the approximation of the the z -position according to [2-22].

In the error plot can be seen that in the situation that p is zero, that means the platform is in the centre of his work area circle, the error is the largest and has a value of 26 [mm]. In the situation that the platform is at the edge of the safe work area ($p = 200$ [mm]), the error is 16 [mm]. That means that if the threshold value of the safety system is set to 250 [mm], the platform can move up to a height of 266 [mm] in the situation that $p = 200$ [mm]. Therefore the threshold value should be set at 234 [mm]. The platform cannot move above 250 [mm] then.

2.2.2 Checking the power supply

The Imotec manipulator is fed via an Uninterruptible Power Supply (UPS). This UPS is able to supply power to the manipulator for at least 10 seconds in case of a power break down from the supplier net. This should give time enough for bringing the end effector to a safe situation and shutting down the computing system. In case of a power down situation, the computing system is alarmed via a relay that checks whether the power supply from the net is present. This should trigger an appropriate control action.

2.2.3 Checking the linear motors

Linear motors based on permanent magnets are robust and reliable because no transmissions are needed to transform in a linear motion. However, the linear motor still can malfunction. For instance, the coils of the translators can heat up too much. Therefore the translators are equipped with thermal resistors to measure the temperature. This temperature can be checked by the computing system and in case of overheating measures can be taken. Another failure that can occur is that the linear motor gets stuck. This can be detected by a growing tracking error and measures can be taken.

2.2.4 Checking the motor amplifiers

The used amplifiers for the Imotec manipulator have safety checks built in. In case of faults, outputs are set high that can be noticed by the computing system and measures can be taken. In case of malfunctioning of the amplifiers the tracking error will become too large, this can be detected and measures can be taken. The amplifiers can check for under/over voltage of the power supply, short circuiting of motor currents and overheating of the amplifiers themselves.

2.2.5 Checking the interface cards

The Imotec manipulator uses three types of interface cards in the computer system: an encoder card, a digital input/output card and an analogue output card. These can all malfunction. Malfunctioning of the encoder card can be detected by the following.

- The value of the encoder reading does not change at all, thus a growing tracking error.
- The difference in values between two successive samples is much bigger than the translators possibly could move in the time difference.

It is difficult to check the analogue output card directly. A feasible method is to compare the error signal of the controller inside the software with a threshold. For instance, if the analogue card would malfunction it either send zero to the output or a fixed value different from zero. In both cases the translators will not move because there is no commutation performed. The tracking error will grow large and this can be detected. Performing the commutation inside the software and not in the amplifiers makes the manipulator inherent safe.

The digital I/O card can be checked by using redundancy. This means using two input channels to read in one signal. For faultless operation the two inputs should be the same. But this method costs a lot of inputs and will not be used. The faulty operation of the digital I/O card will not cause life-threatening situations because emergency stops are not handled in software only, but are also applied directly to a safety relay. Therefore no checks will be applied to the digital I/O.

2.2.6 Checking the computer system

With the computer system there are two types of faults possible. First it is possible that there is a bug in the software. This can be a bug in the controller software but also a bug in the operating system itself. It is hard to detect this kind of software problems. A bug can result in the situation that the tracking error will grow large during a motion or that the manipulator does not react on commands like start and stop.

Secondly the computer can crash totally. This can be detected by means of a watchdog. This is a hardware component, which receives a signal from the computer system and checks if the computer is running. If the computer has crashed, this signal will not be detected and the watchdog will notice the malfunctioning of the computer system. Then proper action can be taken by the hardware, like activating the safety relay. In figure 2.8 the circuitry is given for the watchdog function.

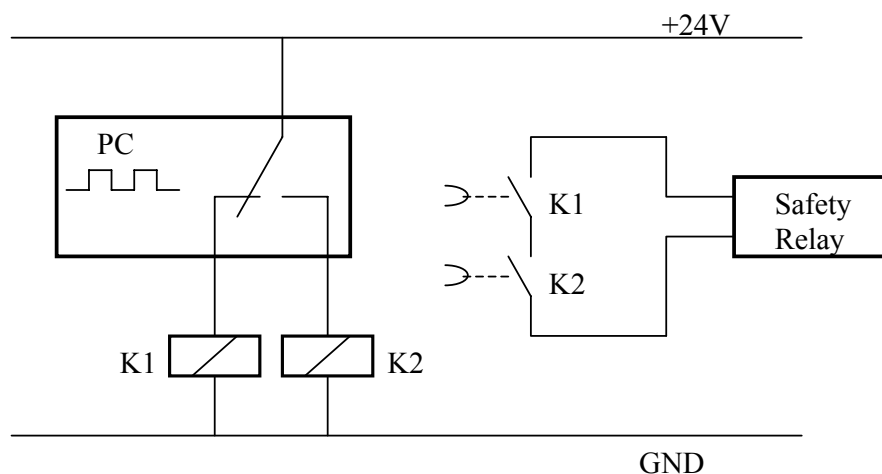


Fig. 2.8 The watchdog circuitry for checking the computer system.

The working principle is as follows. A pulse train from the software drives a switch that activates timer relays K1 and K2. These are of the delayed fall-off type. As long as a pulse train is present with a period time that is smaller than the fall-of time, the contacts K1 and K2 will not

fall-off. When the computer system crashes, the pulse train will stop and after the delay time of the relay, the contact K1 or K2 will fall-off. This will cause the safety relay to take action.

2.3 Faults by the environment

For safe operation of the manipulator, the system should not interact with the environment physically. The Imotec manipulator is covered with a removable transparent security hedge made of lexan. Safety switches mounted on the hedge secure operation of the manipulator only when the hedge is mounted. This hedge ensures that no people or animals can enter the work area of the manipulator during operation. It also provides protection against the possibility that objects are thrown out of the work area by the manipulator.

Another cause of possible faults can be wrong input of the user. The user can cause the manipulator to operate beyond its limits either by mistake or on purpose. The method discussed in 2.2.1 avoids operating the manipulator beyond the limitations even if the input is wrong.

2.4 Applied checks and their responses

In section 2.2 and 2.3 possible faults that can arise have been discussed. Table 2.1 summarizes the checks and their responses for the manipulator.

Check: What	Check: How	Response
Manipulator checks		
Motor temperature	Thermal resistors to digital input	Steering signals zero
Amplifiers	Judging tracking error	Steering signals zero
Encoder cards	Judging tracking error	Steering signals zero
Computer system	Watchdog circuitry (hardware)	Power cut-off
User input		
Exceeding work area	Judging end-effector position	Steering signals zero
Actuator saturation	Judging tracking error	Steering signals zero
Environment		
Entering work area	Sensor on hedge (also hardware)	Power cut-off
Emergency	Emergency stop button to digital input and to hardware	Emergency stop and delayed power cut-off

Table 2.1 Applied checks and their responses for the manipulator.

Most of the safety checks will respond with setting the steering signals to zero if a fault arises. The check of the computer system and entrance of the work area is handled in hardware. The power is cut off by the safety relay that falls off. The remaining checks will be solved in software. The next chapter describes the implementation of the safety checks and the total controller system.

3 Implementation of the safe controller system

3.1 Introduction

Implementation of the controller with all the safeguards in software can be done conveniently by using a high level programming method. This section discusses the several implementation options that are considered. From these options a choice will be made for the final implementation of the safe controller system.

The purpose of the implementation method is to create a complete working controller program with all the mentioned safety checks mentioned in the previous chapter. The program must operate on the computing system of the manipulator, but it is not a requirement that the program must be developed on this system itself.

There are some additional requirements that the used implementation form has to meet:

- The software used must be able to work on a PC under DOS, because that's the operating system used in the manipulator. The preferred programming language is C++. The reason for this is that C++ is an object oriented and fast programming language. Also the Control Laboratory is experienced with C++.
- Testing the created software by means of simulation before implementing it has large advantages. Therefore, it is highly preferable that the software environment has an easy way of doing simulations.

3.2 Possible implementation methods of the safe controller system.

The first and most obvious option is to write code in C++. This is not a high level method and will be laborious. Also, doing tests with the created code by simulations is not easy. A better method is the concept of agent based controller systems as proposed by [Van Breemen, 2000]. This method is more structured than a general programming language. It allows incremental design, which means that functionalities can be added later on without interfering with earlier implemented parts. This is very suitable for the safe controller design.

An agent is an abstract entity that is able to solve a particular part of a total complex problem. Cooperation of multiple agents provides a solution to the total complex problem. For the present context, this is referred to as a Multi Agent Controller system (MAC). In [Bajracharya, 2003] an integrated design tool for Multi Agent Controller systems (IDITMAC) is developed. This tool also makes it easy to test the created software by simulation. A Dynamically Linked Library (dll) file is created, which can be incorporated with 20-Sim modeling and simulation environment. This tool will be used for implementing the safe controller system of the Imotec manipulator.

3.3 The concept of MAC systems

The term agent is widely used in the field of software engineering and artificial intelligence. In [Franklin, Graesser, 1997] an autonomous agent is defined as:

“An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.”

Stated in other words, an agent can decide for it self whether it should undertake some actions. To undertake the action, the agent first has to become active.

A MAC system consists of three basic agents:

- Controller agent
- Sensor agent
- Actuator agent

In figure 3.1 the symbols that are used for the basic agents are given. The next sections will give the explanation and describe the function of the various parts.

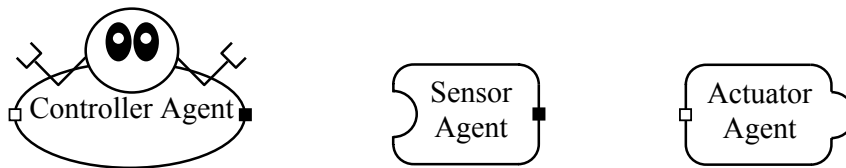


Fig. 3.1 Symbols of the basic agents in a MAC system.

3.3.1 Controller agents

Two types of controller agents can be distinguished in the MAC system.

- Elementary agent
- Composite agent

The elementary agent is the fundamental agent of a MAC system. It implements the local control solution for a part of the global control problem. The composite agent is a pool that consists of elementary and/or other composite agents. This allows hierarchical organization of the MAC system. The overall agent that contains all the other agents and is responsible for the total control system is called a Main agent. This is also a composite agent.

3.3.2 Sensor agents

The sensor agent senses the environment and presents the data to the other agents. The data from the environment can be user input for controlling the system, measured values from the plant and disturbances. Because of the discrete implementation, the data will come from Encoder interface cards, A/D converters and digital inputs cards.

3.3.3 Actuator agents

Actuator agents present the processed control data to the environment. This control data can be signals for indicator lights, steering signals for the plant etc. The data will be presented to the environment by D/A converters and digital output cards.

3.4 Coordination of agents

Cooperation of agents is determined by a coordination mechanism. Agents that are cooperating are combined in a pool of agents (which is a composite agent). A coordination object that is also included in the pool does the coordination. In figure 3.2 the symbol of coordination objects is given. The coordination objects can be divided in three types:

- Independent
- Cooperative
- Competitive

Independent: coordination allows two or more agent to be active at the same time. The agent that are active at the same time have control over different outputs and do not interfere with each other. An independent type coordination is the Parallel coordination object, which is also the default one.

Cooperative: coordination also allows two or more agent to be active at the same time. But now the agent can have control over the same output. Also the output of one agent can be the input of the other ones. Master-slave coordination object and Fuzzy addition coordination objects are cooperative type objects.

- Master-slave coordination: If one agent is active the other one is also active.
- Fuzzy addition coordination: The shared output is a function of the agent outputs that are active.

Competitive: coordination allows only one agent to have control over a certain output. Coordination object of this type are Fixed priority coordination, Sequential coordination and Cyclic coordination.

- Fixed priority coordination: From all the agents that wants to be active, the one with the highest priority is allowed to be active.
- Sequential coordination: the agents of the pool are activated after each other. That means that when one agent stops being active, the next one in the pool becomes active.

- Cyclic coordination: this coordination is almost the same as Sequential except that if the last one in the order stops being active, the first one becomes active again.

With these four coordination objects and the possibility for a hierarchical organization, complex controller problems can be solved in a partial manner. The MAC system for the Imotec manipulator will be described in the next section.



Fig. 3.2 Symbol of the coordination objects.

3.5 MAC system for the manipulator

An overview of the total controller agents for the Imotec manipulator will be given and their functions will be discussed. The entire controller is called the OverallController and has input connections to the buttons like START, STOP and EMERGENCY. Also the three encoder values z_1, z_2, z_3 and the end switches of the motors are acquired by the inputs sensors. The outputs of the OverallController are the control voltages that drive the amplifiers and the digital outputs signals. A schematic overview of the total control system is given in figure 3.3.

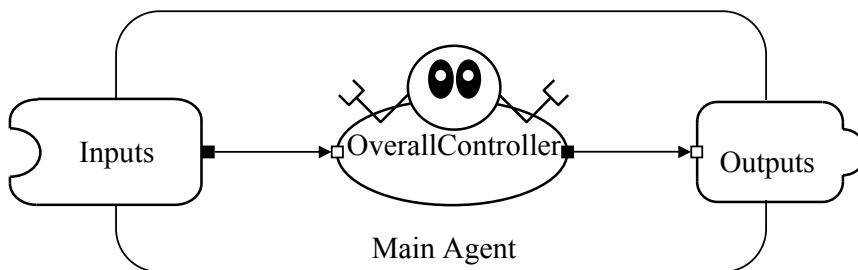


Fig. 3.3 The Main controller agent for the manipulator.

The OverallController consists of a pool with a Startup, Alarm, GuardedEmergency, and a GuardedStandard agent. The coordination is fixed priority, see figure 3.4. In the following sections the function of these agents will be discussed.

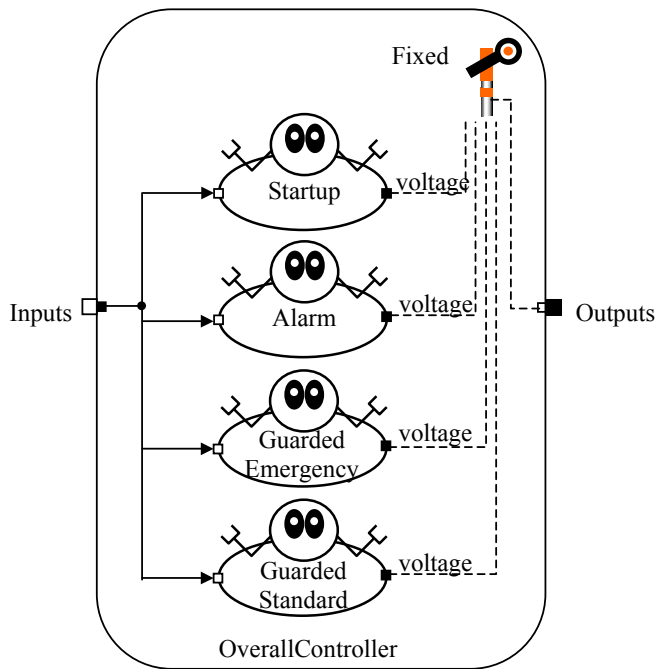


Fig. 3.4 Pool with agents of the OverallController system.

3.5.1 Startup agent

The first agent in priority order is the Startup agent. It becomes active when the ENABLE button is pressed. This button also activates the safety relay. When active, this agent performs the alignment and the homing of the manipulator. At the end of the homing procedure, the platform is positioned in the lowest centre point of the safe work area cylinder, see figure 3.5. This point is referred as the home location. The manipulator is now ready for operation and waiting to perform a specified motion.

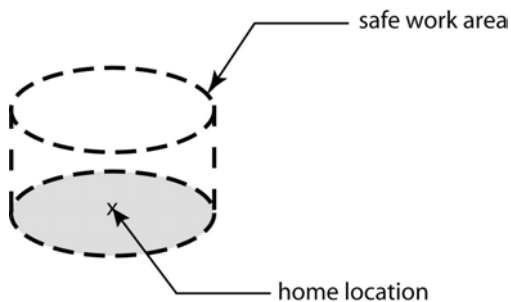


Fig. 3.5 Location of the home position.

3.5.2 Alarm agent

The Alarm agent is an elementary agent and becomes active when the manipulator approaches situations that are not allowed or when malfunctioning of the manipulator occurs. It is activated on the following conditions:

- The thermal resistors indicate that the linear motors heat up too much and digital input is set high.
- The platform exceeds its safe work area; approximation done according to section 2.2.1.
- Internally generated errors; e.g, unexpected startup position, aligning unsuccessful, encoder index pulse not found during homing procedure.

Also the end switches of the linear motors activate the alarm agent, but theoretically this will not occur, because the platform has to exceed its safe work area to reach the end switches. Then the agent is already activated by the condition of exceeding the safe work area. When the alarm agent becomes active, all the steering signals are set to zero and the translators will fall down towards the end dampers. There is a possibility that the translators fall down from a high position on the end dampers. The end dampers have been constructed so to withstand the collision that occurs when the translators fall down from the highest position, but this should preferably be prevented.

3.5.3 GuardedEmergency agent

In case of an emergency, the EMERGENCY button should be pressed. This makes the GuardedEmergency agent become active. The EMERGENCY button also directly activates the safety relay. The safety relay switches of the power supply with a time delay of three seconds. Then the platform and the translators of the motors will fall down because of gravity. The function of the GuardedEmergency agent is to bring the manipulator in a safe situation within three seconds. This safe situation is at a position where the translators are about 5 [cm] above their end dampers and the speed is about zero. Then the translators fall down over a very short distance and this will not cause any damage. During the positioning of the translators, the tracking error is monitored by an ErrorGuard. When any of the three tracking errors grow out of the bounds, this agent sets all the steering signals to zero.

3.5.4 GuardedStandard agent

The GuardedStandard agent consists of a Standard agent and an ErrorGuard in master slave coordination. As before, the ErrorGuard agent monitors the tracking error and sets the steering signals to zero in case of exceeding the bounds.

The Standard agent implements the control scheme shown in figure 3.6. This is done by combining a ModeSwitchController and a GravityCompensator (GC) via a fuzzy addition coordination. The gravity compensator currently simply generates a constant force to compensate for the non-variable parts of the gravity load. In the future this might be extended to a compensation scheme that depends on the actual manipulator position. The motivation for this

gravity compensator is that by using this scheme, mode switching will not induce transition responses due to the gravitational load.

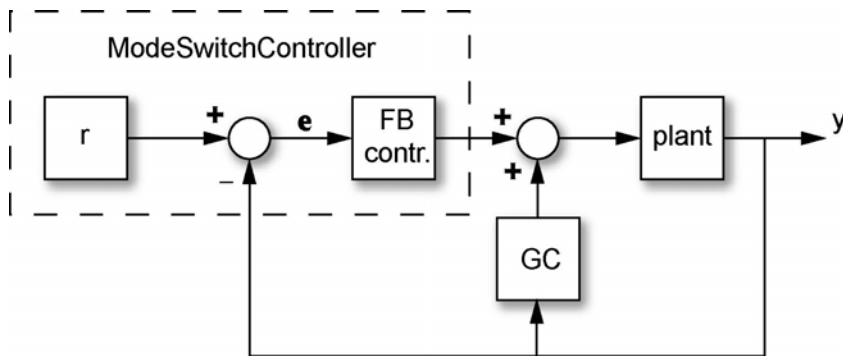


Fig. 3.6 Implementation of the Standard agent

3.5.5 ModeSwitchController agent

The ModeSwitchController agent is given in the figure 3.7. It consists of four agents in fixed priority coordination. The HoldZero agent is a controller agent with the lowest priority that keeps the platform at its home position. The Shutdown agent becomes active when the STOP button is pressed for four seconds. When active, it brings the translators of the linear motors to a position where they are just above their lower end stops and then it shuts down the program and switches off the power. When the START button is pressed, the Operate agent becomes active and the manipulator is then in operation mode. Two parallel working agents perform the operation mode, see figure 3.8. The PathFromFile agent reads the reference file samples into an array and presents every sample time a value to the PID agent. For settings of the PID controller see appendix D. The manipulator then performs the motion as specified in the sample file. In the case that the starting point of the reference path is not the same as the home point of the manipulator, the platform moves slowly to this starting point and then the motion is performed according to the reference path. This functionality is integrated in the PathFromFile agent.

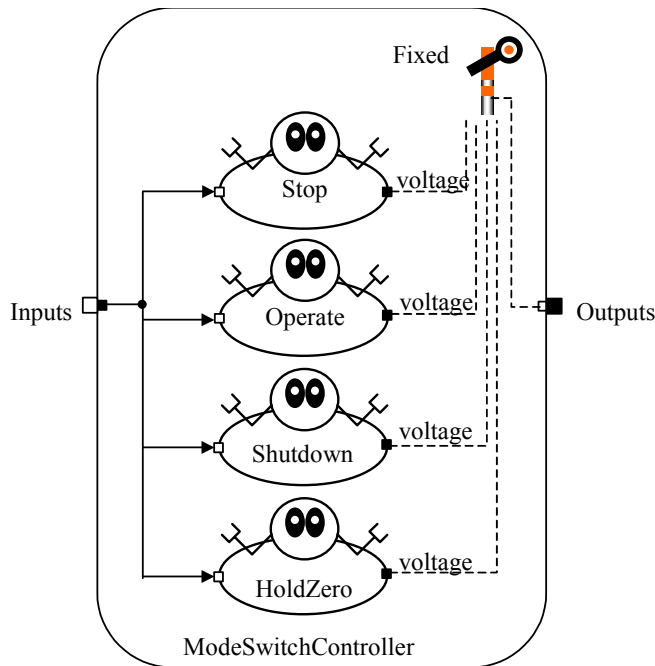
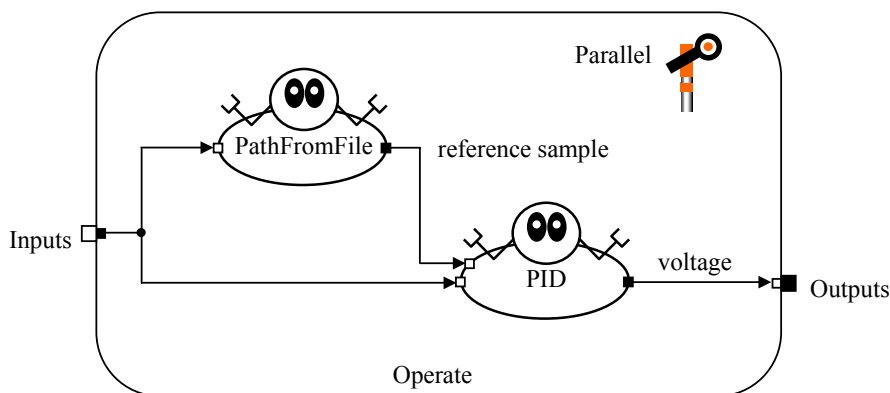


Fig. 3.7 Pool with agents of the ModeSwitchController

When the STOP button is pressed, the Stop agent, which has the highest priority, becomes active and stops the manipulator by bringing it back to its home position. The Stop agent pool is given in figure 3.9. It consists of a Brake agent and a GoSteadyAll agent. The brake agent is a speed control agent that brings the speed of the manipulator to zero. When the STOP button is pressed, the momentary speed is determined. This measured speed is used to generate an inverse desired speed step to zero. A low-pass filter filters this desired speed step and the output is the reference for the speed controller. By using a filter, a smooth reference is generated without much computational effort.

When the speed of the manipulator is almost zero, the Brake agent becomes inactive and the GoSteadyAll agent becomes active. This agent brings the manipulator to its home position after which the Stop agent becomes inactive. Like the Brake agent, also the GoSteadyAll agent works with a low-pass filter. Now the a desired position step is filtered to create a smooth reference path. After the home position is reached, the HoldZero agent automatically becomes active and keeps the manipulator at its home position. The manipulator is then ready and waiting to go in operation mode again.



22 Fig. 3.8 The Operate agent pool.

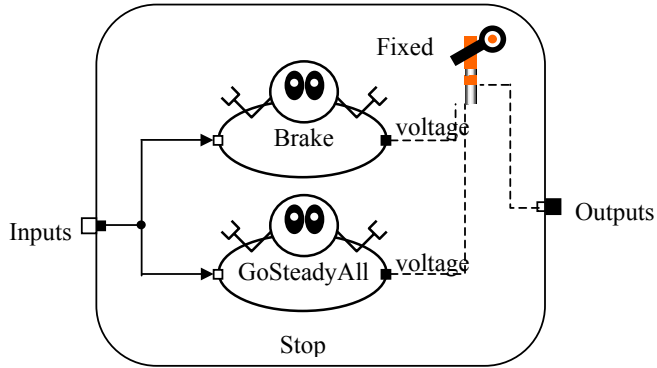


Fig. 3.9 The Stop agent pool.

3.6 Total overview of the MAC system

In figure 3.10 an overview is given of the total MAC system in hierarchical form. With the aid of IDITMAC created software can be compiled as an executable file to implement in the computing system of the manipulator. Also a dll file can be created for simulation in 20-Sim. It can be concluded that the agent based approach enables a convenient design process. The results of the simulation and the implementation will be discussed in chapter 5. The code for the total MAC system is given in appendix E.

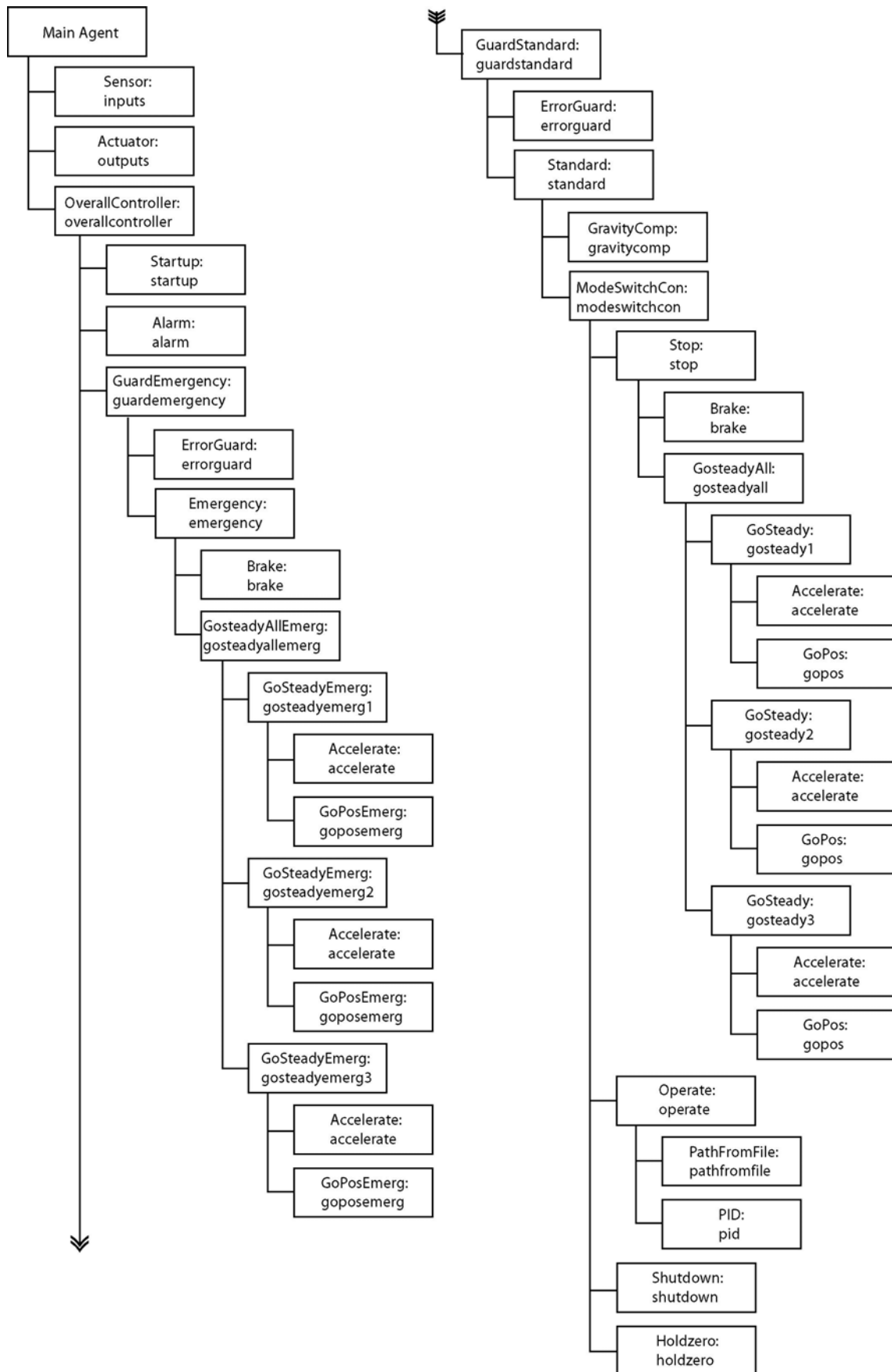


Fig. 3.10 Hierarchical overview of the total MAC system.

4 Path generation

4.1 Introduction

To perform motions with the manipulator, a motion profile input is needed by the PathFromFile agent. Motion profiles vary in shape and order. In one-dimensional manipulators, the shape of the motion profile is restricted to a straight line only, whereas in a multiple axes manipulator, much more complex shapes can be defined. Usually, profiles are defined as piecewise polynomials. The order of the profile gives the smoothness of the path. Higher order motion profiles are easier to follow by the end-effector, but require more complex computation and usually imply larger accelerations. This chapter will discuss path generation in general for multidimensional manipulators and a generally applicable tool for path generation will be developed.

4.2 Tools for path specification

For performing a motion, the controller of the actuator needs a sample of the path at each sample instant. This means that the path has to be presented to the controller as a list of reference points. It is a laborious and error-prone job to create this list manually, especially for complex trajectories. A tool on the market which allow us to give up begin points, end points, speed, acceleration etc. and creates the reference position samples from these parameters, is the Motion Assistant from National Instruments [www.ni.com/motion]. See figure 4.1 for a screndump of the tool.

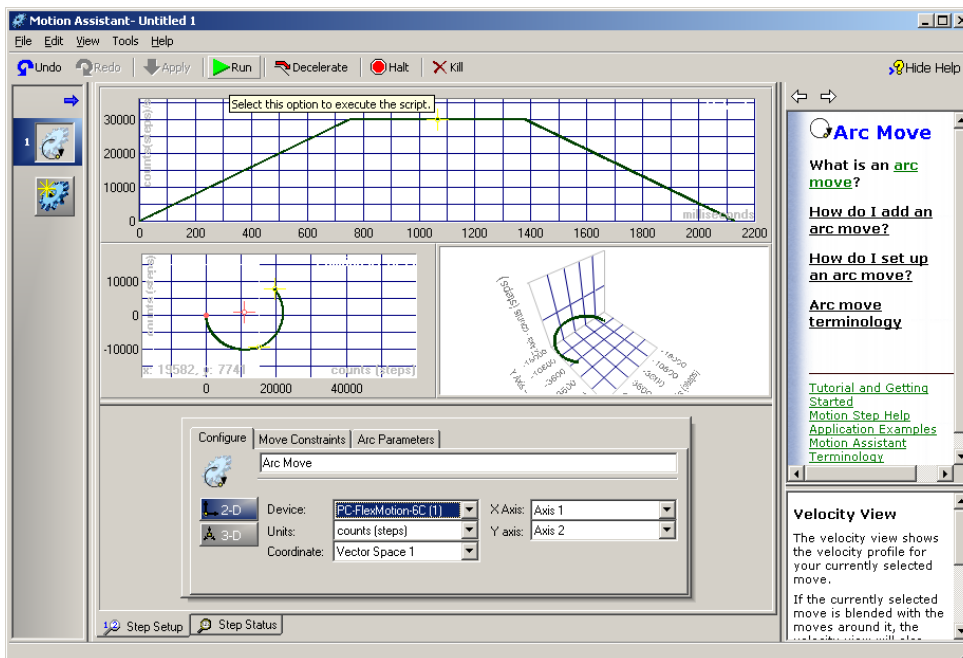


Fig. 4.1 Screenshot of National Instruments path specification tool.

The Motion Assistant allows specifying motions in a graphical manner. By clicking and dragging with the mouse, we can manipulate the shape of the motion. Types of shapes that are available in the tool are straight-line movements and arc movements for up to 3 degrees of freedom. For the order of the profile we have the option between second order and third order profiles, which are also called respectively trapezoidal and s-curve profiles (these terms refer to the shape of the speed curve). A major drawback of the Motion Assistant is that it works only with hardware interfaces from National Instruments. This conflicts with the desire for a generally applicable path specification tool. Therefore the Motion Assistant will not be used further. Based on concepts as present in the Motion Assistant, the choice has been made to develop a path generation tool, which is hardware independent. This will be discussed in the next section.

4.3 The Path Generator

The generally applicable tool, see figure 4.2, uses an XML format to store motion information in an structural way. See Appendix C for an introduction to XML. The tool is written in Borland C++ Builder. An advantage of this is that the programming environment allows us to work directly with XML documents. This is done with the component TXMLDocument. It has methods like AddChildNode, GetChildNode, GetNodeName for manipulating nodes in an XML document directly.

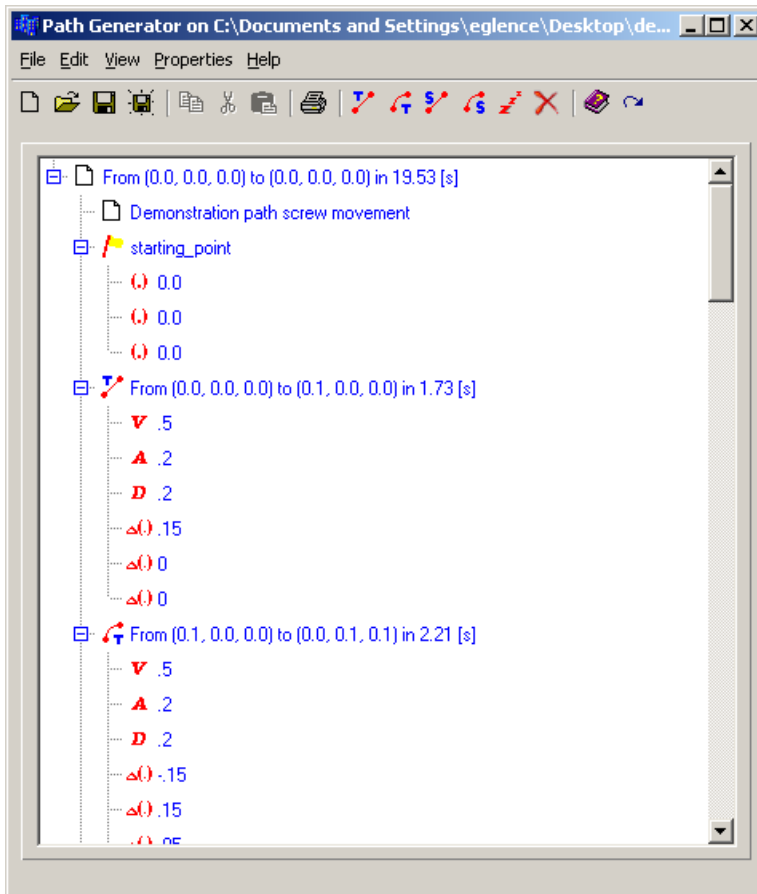


Fig. 4.2 The generally applicable path generation tool.

The tool creates reference points in three coordinates. For the Imotec manipulator these coordinates are interpreted as positions x,y,z of the end-effector. The reference points are transformed according the inverse kinematics of the manipulator. For the Imotec manipulator this is done with [2-1]. At the moment no clicking and dragging options are available to set the shape of the motion. Also checking the created path with a plot must be done externally. For this, free plotting programs like GNUPLOT can be used. Motions are built up of segments. Each segment is a motion from standstill to standstill. Segments with initial and final speed not equal to zero are considered, but a problem with this is that discontinuities in speed and acceleration will arise at the joint between two segments. This can be overcome by connecting the segments with a spline movement or a Bezier curve. A major drawback of this is that considerable computational effort is needed for control over the shape of the Bezier curve. The manipulator could exceed its workspace if the Bezier curve is not well defined. So for the moment we only consider motion segments from standstill to standstill.

Available motion shapes are straight line and arc shapes. The latter of course only in case of two or more dimensional manipulators. Profile types that can be chosen are trapezoidal and s-curve. It is desirable to have the opportunity to enter different values for the acceleration and the deceleration of a single segment. This option is included in the Path Generator. See table 4.1 for the functions of the Path Generator.

When the desired segments have been added, the parameters can be changed by double clicking and editing the relevant items in the tree. The parameters and their meanings can be found in table 4.2.








Button	Function
	Adds a straight motion segment with trapezoidal profile.
	Adds an arc motion segment with trapezoidal profile.
	Adds a straight motion segment with s-curve profile.
	Adds an arc motion segment with s-curve profile.
	Adds a wait in the motion
	Deletes a wait or a motion segment.
	Creates the file with reference samples

Table 4.1 Functions of the Path Generator tool.











Icon	Parameter
	Starting point coordinate for the motion.
	Distance coordinate of a motion segment.
	The maximum speed of the motion segment.
	The maximum acceleration of the motion segment.
	The maximum deceleration of the motion segment.
	The jerk of the motion segment with s-curve profile.
	The radius of an arc movement.
	Angle of an arc movement, more or less than π .
	Auxiliary coordinate point for plane of arc movement.

Table 4.2 Parameters of the Path Generator tool.

After the desired motion has been set up, the reference sample file can be built by clicking the  button. The user will be prompted to enter the sample frequency of the controller system. A sample file with extension .egl will be created. In the next sections, the algorithms for creating a sample file from the XML file will be discussed. An example of a saved XML path file is given in appendix C.

4.4 Trapezoidal profile algorithm

For creating a trapezoidal motion profile, the following parameters are needed. The maximum speed V_{\max} that is allowed to be reached, the acceleration A , deceleration D and the length of the stroke h . In figures 4.3 and 4.4 the position, speed and acceleration plots are given of the trapezoidal motion for two cases: when the maximum speed is reached within the given stroke h and when not.

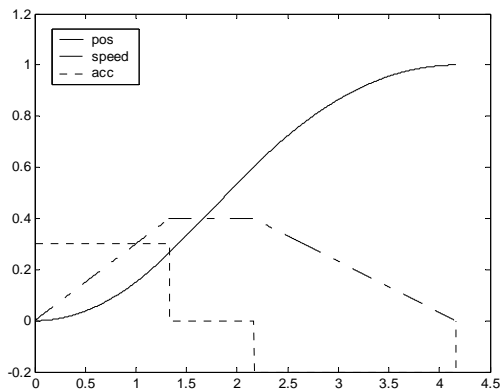


Fig. 4.3 Trapezoidal motion profile when the maximum speed is reached.

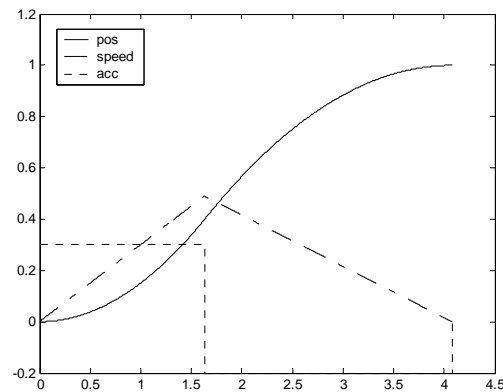


Fig. 4.4 Trapezoidal motion profile when the maximum speed cannot be reached.

In the algorithm, these two situations have to be distinguished. The algorithm is based on integrating the profile of the speed to a position; the speed profile is obtained by applying the given A and D . To take the right speed profile, a check must be performed to see if the maximum speed can be reached within the given stroke h . This is done by calculating the distance that would be travelled if the maximum speed would be just reached. If this distance, which is called h' , is larger than the given stroke h , the maximum speed will not be reached. With these tests, the following speed profiles are obtained and can be integrated, see figure 4.5.

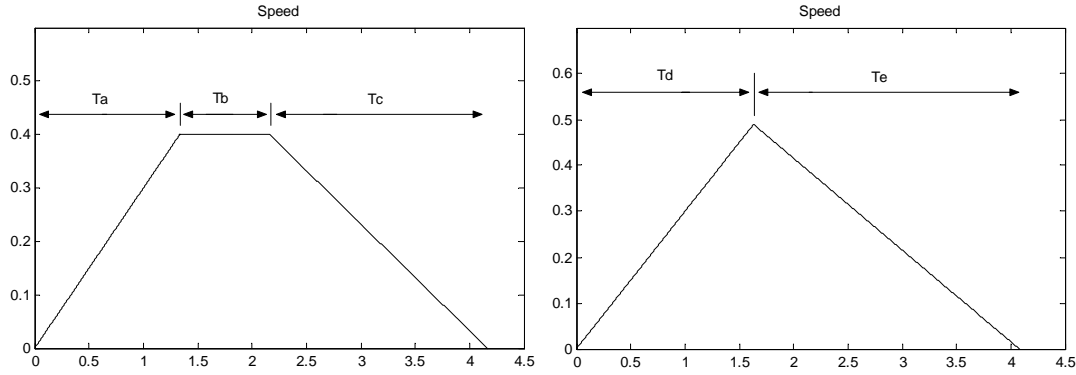


Fig. 4.5 Speed profiles that are integrated to a position profile.

To perform the actual integration, the times T_a till T_e need to be known. These can be calculated. First we take a look at the profile where the maximum speed is reached.

We introduce the times T_1 and T_2 which are respectively the times to reach V_{\max} with the given acceleration A and to return from V_{\max} to zero speed with the given deceleration D .

$$T_1 = \frac{V_{\max}}{A} \quad [4-1]$$

$$T_2 = \frac{V_{\max}}{D}$$

with this the mentioned h' becomes.

$$h' = \frac{(T_1 + T_2)V_{\max}}{2} \quad [4-2]$$

The times T_a , T_b and T_c can be calculated as:

$$T_a = T_1$$

$$T_b = \frac{(h - h')}{V_{\max}} \quad [4-3]$$

$$T_c = T_2$$

For the situation where V_{\max} is not reached the following holds:

$$h = \frac{AT_d^2 + DT_e^2}{2}, \quad [4-4]$$

$$T_d = \frac{D}{A}T_e \quad [4-5]$$

This yields

$$h = \frac{\frac{D^2}{A} + D}{2}T_e^2 \quad [4-6]$$

Now the value of T_e can be calculated as

$$T_e = \sqrt{\frac{2h}{\frac{D^2}{A} + D}} \quad [4-7]$$

With relation [4-5] also T_d is known and the integration can be performed.

4.5 S-curve profile algorithm

For the s-curve profile, an extra parameter is needed in addition to the trapezoidal profile parameters, namely the derivative of the acceleration/deceleration called jerk. For the acceleration A and deceleration D the same value for the jerk is taken. Only the maximum values of A and D can be different. Like the trapezoidal profile, also here there are several situations that have to be distinguished. Now not only a check for reaching the maximum speed has to be done, but also whether the maximum acceleration/deceleration can be reached within the given stroke h . In figure 4.6 a plot of the S-curve profile is given when the maximum speed, the maximum acceleration and the maximum deceleration are reached.

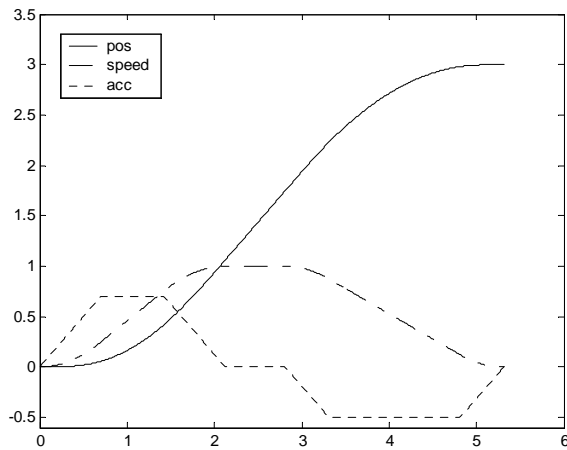


Fig. 4.6 S-curve motion profile with both maximum speed and acceleration reached.

The basis acceleration profile is given in figure 4.7. This can be integrated two times to obtain the desired position profile. Before the integration can be performed, the times T_a till T_g have to be known. If the maximum speed is not reached, T_d equals zero. If the maximum acceleration or deceleration is not reached, respectively T_b and T_f equal zero.

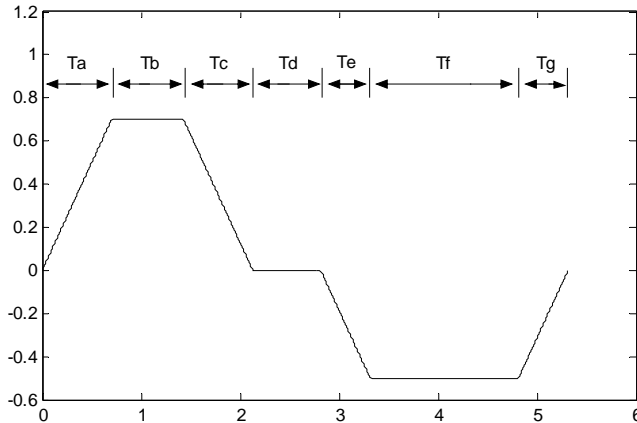


Fig. 4.7 Basis acceleration profile that is integrated to position.

First we distinguish whether or not the maximum speed is reached. The times T_a , T_b and T_c are calculated to reach the maximum speed with the given maximum acceleration and jerk. Then the times T_e , T_f and T_g can be calculated to return from this maximum speed with the given maximum deceleration and jerk. With the calculated times, the total covered distance h' can be calculated. If this distance is smaller than the stroke h , it is certain that the maximum speed is reached and T_d is larger than zero. The time T_d that the maximum speed holds on can be calculated as:

$$T_d = \frac{h - h'}{V_{\max}} \quad [4-8]$$

Then all the times are known and the integration can be performed. In the situation where the distance h' is larger than h , it is certain that the maximum speed will not be reached. Then the acceleration profile will have one of the four following shapes (figures 4.8).

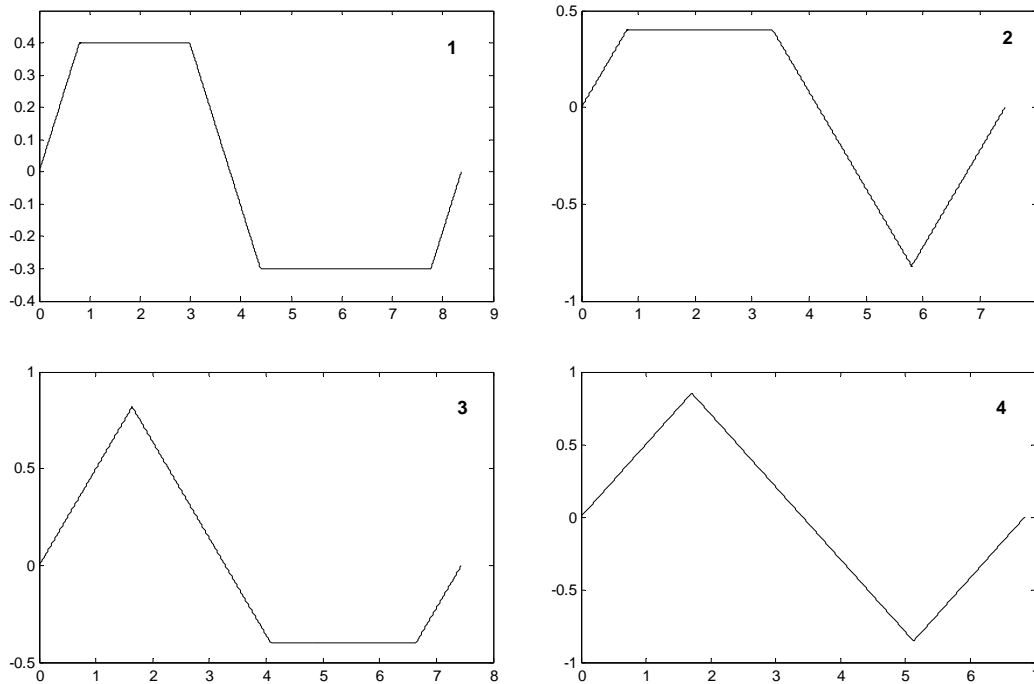


Fig. 4.8 Possible acceleration profiles when the maximum speed is not reached.

Profile 4 is solved easily because of symmetry. Profile 1 can be solved in the following way. First the distance h'' that is covered when the maximum acceleration is just reached is calculated. This distance h'' is smaller than the stroke h . A formula can be derived for the distance $(h - h')$ that is needed to equal stroke h as a function of ΔT_b . The order of ΔT_b is two in this function. That means that it can be solved with the ABC formula. Profile 2 and 3 are solved on the same manner as with profile 1, also here a formula can be derived for the distance that is needed to equal h as a function of ΔT_c . But here the order of ΔT_c is three in this function. This means that there is no analytical general solution. The function can be solved numerically by iteration. The formula's for $(h - h')$ are not given because of their large size and non-transparency.

4.6 Straight line movement

The straight line movement is a relatively simple movement. The trapezoidal or S-curve profile that is generated has to be assigned to the straight line movement. If the manipulator is a single axis manipulator then the transformation is not needed. For multiple axis manipulators like the Imotec manipulator the transformation goes as follows. First the straight line movement has to be decomposed in the distances along the dimensions, like Δx for the distance covered along the x-axis etc. Then the transformation becomes:

$$h = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \quad [4-9]$$

$$\begin{aligned}
 x &= pos \frac{\Delta x}{h} \\
 y &= pos \frac{\Delta y}{h} \\
 z &= pos \frac{\Delta z}{h}
 \end{aligned}
 \tag{4-10}$$

With pos the instantaneous position value of the trapezoidal or S-curve profile.

4.7 Arc movement

The arc movement is more complex to generate but also more difficult to enter/define by the user. In xy -manipulators the parameters needed are a beginpoint, endpoint and the radius of the arc. However this is not sufficient to define the arc motion unambiguously. Figure 4.9 shows the possible arc motions with only the parameters above given.

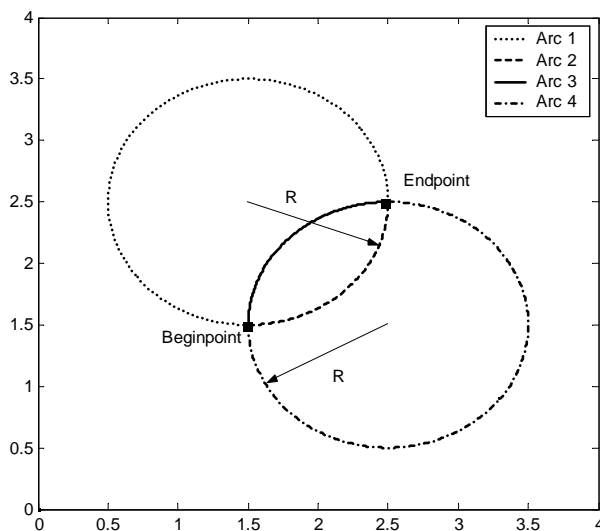


Fig. 4.9 Possible arc movements when only a beginpoint, endpoint and radius are entered in a two dimensional manipulator.

From the figure can be seen that there are four possible arc movements: two arcs that make a smaller angle than π and two arcs that make a larger angle than π . Defining that the arc motion is clockwise from beginpoint to endpoint for the smaller angle arcs and counter clockwise for the large angle arcs, makes the method more unambiguous. This means that in figure 4.9 only arc 3 and 4 are possible. To complete the method also a choice between arc 3 and arc 4 has to be made. This is done by entering an extra parameter, which defines whether the angle of the arc is smaller or larger than π .

In three dimensional manipulators the arc is not restricted to the xy -plane. An extra parameter is needed to define the plane of the arc. There are various ways of defining this plane. For instance, an option is to first define the arc in the xy -plane as with two-dimensional manipulators. Then by giving up two angles the arc can be placed in the desired plane. This is what is done in the Motion Assistant from National Instruments, See figure 4.10.

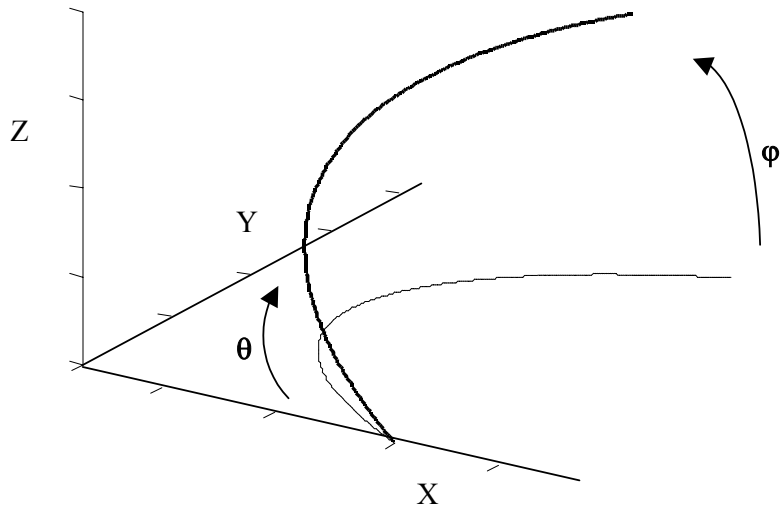


Fig. 4.10 A method for defining arc motions in three dimensional manipulators, first the arc is defined in the xy -plane and then rotated and tilted to the desired plane.

A major drawback of this method is that the endpoint of the arc motion cannot be entered directly by the user. This is highly desirable when defining a motion. A better method which allows entrance of the endpoint is the following method. A third auxiliary point is entered which defines the plane together with the begin and endpoint. The auxiliary point is given up with respect to the beginpoint. This method is used in the developed Path Generator tool.

4.8 Assigning the sample reference points to the arc movement

To create the actual arc movement, the trapezoidal or S-curve profile has to be assigned to the motions along the axes of the manipulator. In this section the procedure for a three dimensional one like the Imotec manipulator will be discussed. A two dimensional transformation can be derived easily from the three dimensional one. In figure 4.11, some vectors are defined that are needed for the conversion. An arc movement can be described as the rotation of vector e in the plane of the arc. This vector rotates from the beginpoint B_p to endpoint E_p , with a speed conform the motion profile that is used (trapezoidal or S-curve). A_p is the auxiliary point of the arc motion that defines the plane of the arc movement.

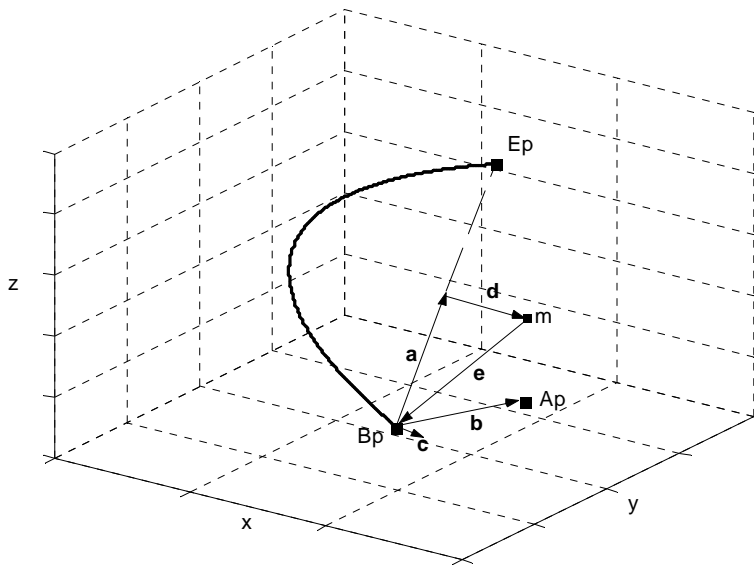


Fig. 4.11 Vectors that are defined to transform the arc parameters in to the sample file.

A_p is entered as a point with respect to B_p and therefore equals vector b . The centre point m of the arc has to be determined to create the rotation vector e . The point m can be found on the following way. First the vector a is found as:

$$a = \frac{E_p - B_p}{2} \quad [4-11]$$

The normal on the plane of the arc motion, vector c , is found by:

$$c = b \times a \quad [4-12]$$

We define a new vector g , which is aligned with d , but has not the same length as d .

$$g = a \times c \quad [4-13]$$

Then the vector d can be found as

$$d = \frac{1}{|g|} \cdot \sqrt{R^2 - |a|^2} \cdot g \quad [4-14]$$

With R , the known radius of the arc motion. And finally for the point m :

$$m = B_p + a + d \quad [4-15]$$

With the centre point m of the rotation and the motion profile known, the actual rotation of vector e can be performed to obtain the sample points along the arc. The rotation vector e is found by:

$$e = B_p - m \quad [4-16]$$

The rotation is done by placing the plane of the arc in the xy -plane, rotating about the z -axis over a variable angle β and then transforming back. The transformation matrix to place the plane of the arc in the xy -plane is called R_c and the rotation matrix in the xy -plane is called $R_{(\beta)}$.

The actual sample values s of the arc motion is found by:

$$s = m + R_c^{-1} R_{(\beta)} R_c e \quad [4-17]$$

R_c follows from B_p , E_p and A_p . We need to find the variable angle β for rotating e about the z axis with the rotation matrix $R_{(\beta)}$.

The overall rotation angle α that vector e makes, is found by the inner product of the vectors e and d .

$$\alpha = 2 \cos^{-1} \left(\frac{-d \cdot e}{|d||e|} \right) \quad [4-18]$$

With the known radius R , the angle α is related to the motion stroke h by:

$$h = \begin{cases} \alpha R & \text{angle} < \pi \\ (2\pi - \alpha)R & \text{angle} \geq \pi \end{cases} \quad [4-19]$$

Take pos to be the instantaneous values of the trapezoidal or S-curve profile, running from zero to h . The values of pos can be transformed back in the variable angle β by:

$$\beta = \begin{cases} \frac{-pos}{R} & \text{angle} < \pi \\ \frac{pos}{R} & \text{angle} \geq \pi \end{cases} \quad [4-20]$$

As a result, β will go from zero to α [rad] with a trapezoidal or S-curve motion profile and the rotation can be done.

5 Simulations and experiments

5.1 Introduction

In this chapter the developed path generator and safe controller system will be tested. Both simulations and actual experiments are carried out. First a demonstrator path is made with the Path Generator tool. This path will be used to perform motions with the manipulator. The response on emergency and alarm situations will be tested. Also the response on a too large tracking error which is monitored by the ErrorGuard agent is verified. Because of time constraints, the experiments with LFFC have not been carried out.

5.2 Design of demonstrator path

First a path is made that will operate the manipulator in its safe work area. Therefore an alarm situation is not expected to occur and the manipulator should operate normally. The specified motion is a spiral with a radius of 0.15 [m] that curls from zero height to 0.25 [m] and then returns to the home position in a straight line. The motion starts in the home position, see figure 5.1 for a plot of the motion. The maximum speed is 0.2 [m/s] and the maximum acceleration 0.5 [m/s²]. Later on, this path will be modified such that it exceeds the safe work area and alarm situations should arise.

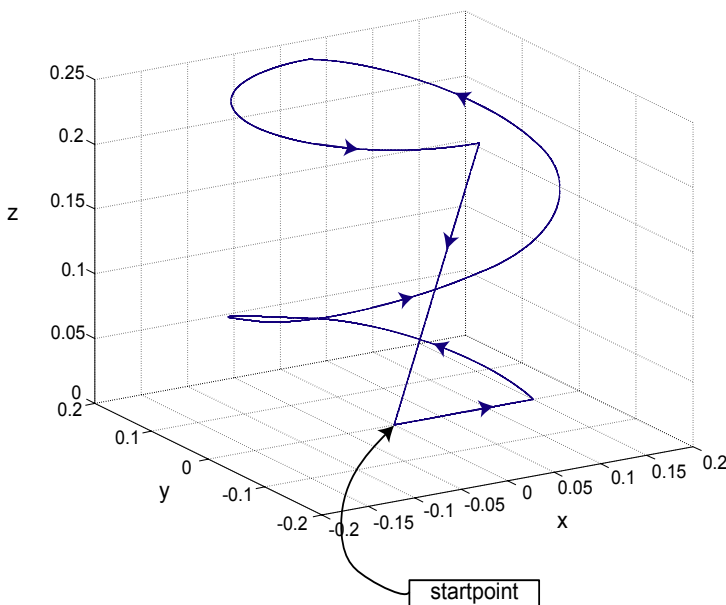


Fig. 5.1 Plot of the path created with the Path Generator tool that operates the manipulator in its safe work area.

5.3 The 20-Sim model of the manipulator.

A 20-Sim model of the manipulator is available that can be used for verifying the developed software in simulation (De Vries, De Kruif). An overview of the top level of this model is given in figure 5.2. The plant model consists of the following parts:

- 3 *linear motors* that operate in z -direction. These submodels primarily consist of a modulated source of force and a translator mass;
- 3 *arms* with ball joints on both ends. The arms implement the kinematics of the robot. In order to allow for the use of explicit integration methods, they also contain spring- and damping effects. I.e., although these effects could also be motivated physically, they have been included for simulation purposes and the corresponding parameter values are chosen from this perspective;
- a *platform* (plateau). This plateau has mass and can move with three degrees of freedom (x, y, z).

For evaluation purposes, the model has been extended with a multi-agent controller and with signal sources that emulate the digital hardware inputs. The input buttons are implemented as pulse generators. The controller is a sub model that contains the dll created with the IDITMAC program. The plant model has not been validated and hence is not competent for tuning the PID settings of the controller. But this is not a problem; the goal of the used model is to verify the functionality of the created software and not to optimize the controller for small tracking error.

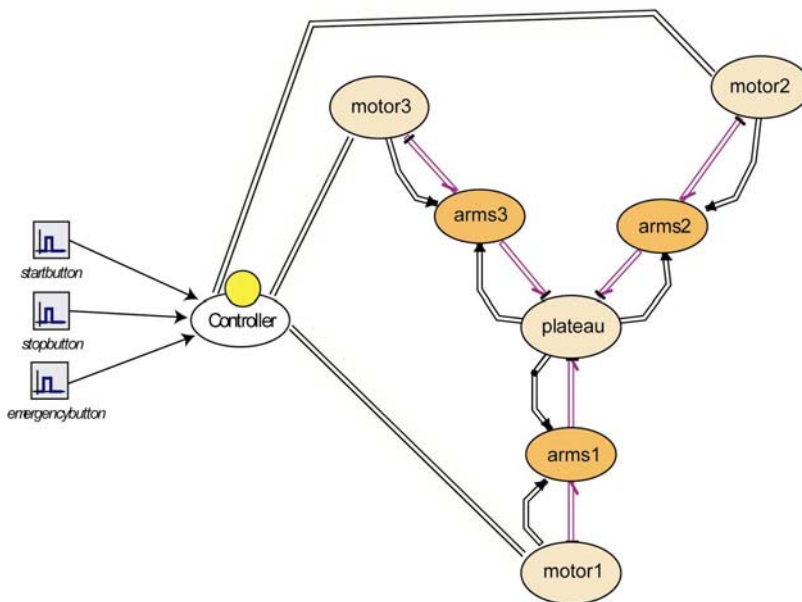


Fig. 5.2 20-Sim model of the Imotec manipulator.

5.4 Verifying the standard mode in simulation

The first simulation that is done is to test the normal operation mode of the manipulator. That means that the manipulator is waiting for the START button to be pressed. When the START button is pressed, the manipulator has to perform the movement as specified in figure 5.1. When the STOP button is pressed before the end of the specified motion, the manipulator must brake and return to its home position. First the results are given of the case where the STOP button is not pressed and the motion is performed till the end, see figure 5.3.

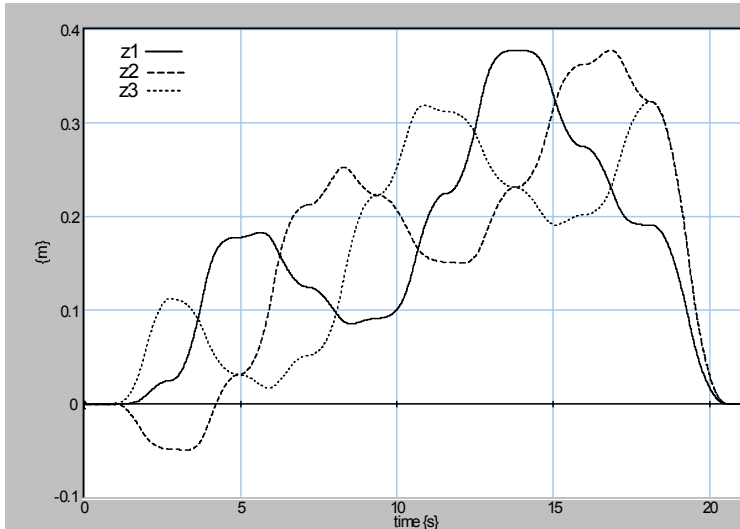


Fig. 5.3a Simulation of the motion given in figure 5.1 for z_1 , z_2 and z_3 ; the START button is pressed at $t=1$ sec. The STOP button is not pressed and the motion is performed till the end.

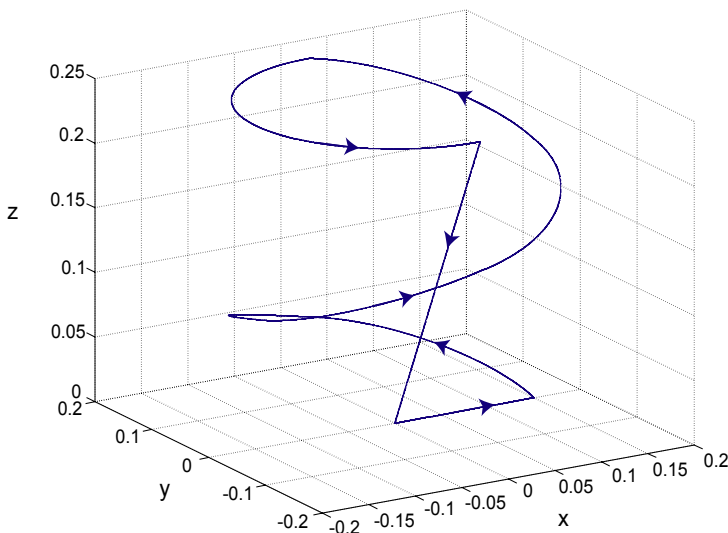


Fig. 5.3b 3D simulation of the motion given in figure 5.1 for x , y and z , the START button is pressed at $t=1$ sec. The STOP button is not pressed and the motion is performed till the end.

From the plots can be seen that the manipulator behaves as expected. In figure 5.3a it can be seen that the platform is held in its home position until the START button is pressed at $t = 1$ sec. Then the motion is performed. Notice that z_2 becomes negative first. The fact that the translators are not at their lowest point when the platform is in its home position explains this. z_1 , z_2 and z_3 can vary maximally between -0.08 [m] and 0.45 [m]. Now the test will be continued with the case that the STOP button is pressed before the end of the specified motion. These results are given in figure 5.4.

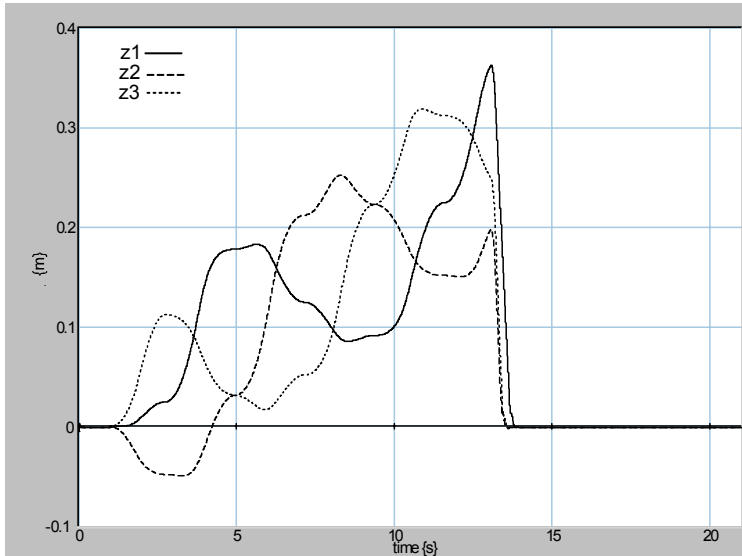


Fig. 5.4a Simulation of the motion given in figure 5.1 for z_1 , z_2 and z_3 , the START button is pressed at $t=1$ sec. The STOP button is pressed at $t=13$ sec. The motion is stopped and the manipulator moves to its home position.

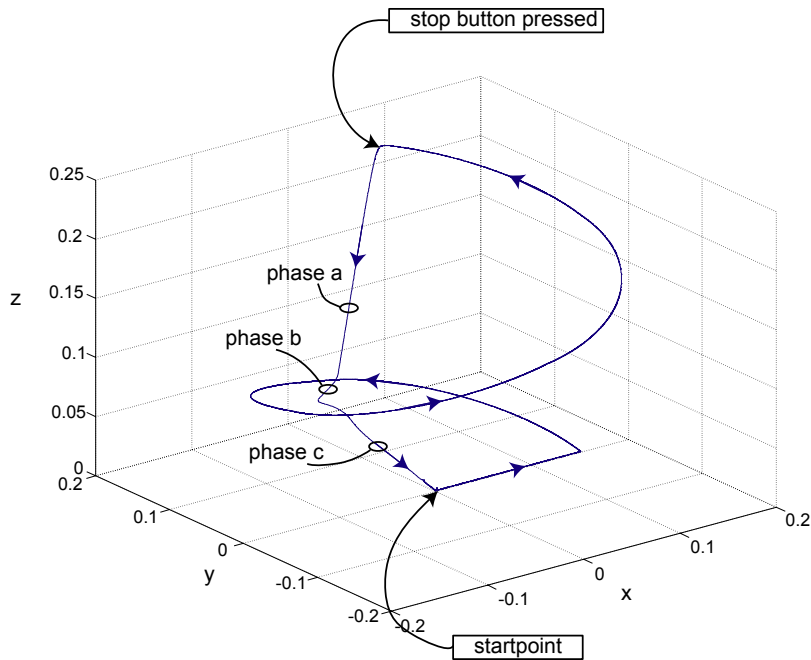


Fig. 5.4b 3D simulation of the motion given in figure 5.1 for x , y and z , the START button is pressed at $t=1$ sec. The STOP button is pressed at $t=13$ sec.

Again from the plots of figure 5.4 can be seen that the manipulator behaves as expected. When the STOP button is pressed at $t = 13$ sec. the manipulator brakes and moves towards its home position, then it is ready to perform the motion again when the START button is pressed. Fig 5.4b shows the motion of the platform. Due to the kinematics, the shape of the motion after the stop is not straight, but rather arbitrary. One can recognize three phases:

- a: all the translators move towards their zero point.
- b: translator 2 has reached its zero point and translator 1 and 3 are still moving.
- c: only translator 1 is still moving towards its zero point.

The followed path of the platform is arbitrary because the speeding down and positioning of the translators towards their home positions is done independent from each other.

5.5 Verifying the ErrorGuard

In case of too large tracking errors, all the steering signals should be set to zero by the ErrorGuard agent. This simulation is given in figure 5.5. At $t = 4.4$ sec., the value of encoder 3 is set to zero to emulate an error. This results in a large tracking error and therefore the ErrorGuard agent should become active.

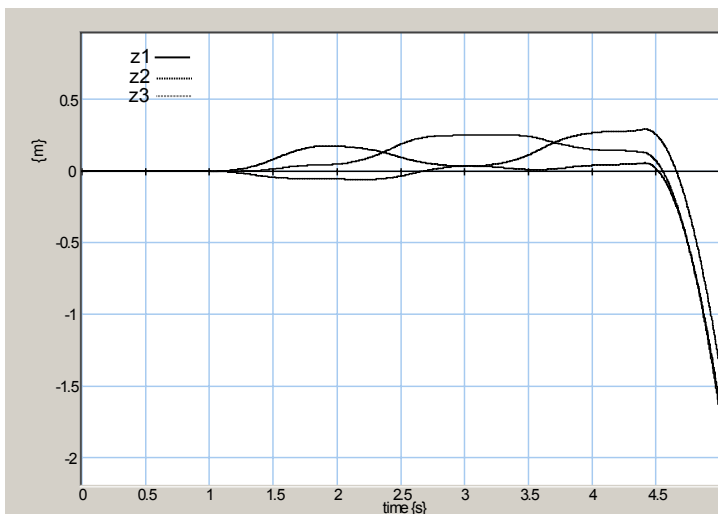


Fig. 5.5 Simulation of the ErrorGuard agent. An encoder error is emulated at $t = 4.4$ sec. The steering signals are set to zero then and the translator fall down towards the end dampers

In the plots can be seen that. We note that when the encoder value 3 is set to zero, the alarm agent (which has a higher priority) also can become active. The alarm agent uses the encoder reading to approximate the position of the platform. A bad encoder reading can result in determining that the platform is outside its safe work area. However, the response of the Alarm agent is the same as the response of the ErrorGuard agent (setting all the steering signals to zero) and therefore this has no consequences.

5.6 Verifying the emergency situation in simulation

In emergency situations the EMERGENCY button is pressed. The manipulator should react on this by bringing the platform to a safe position first and then switching off the power. In simulation, switching off the power is not taken in account. The simulation of the emergency situation is given in the plots of figure 5.6.

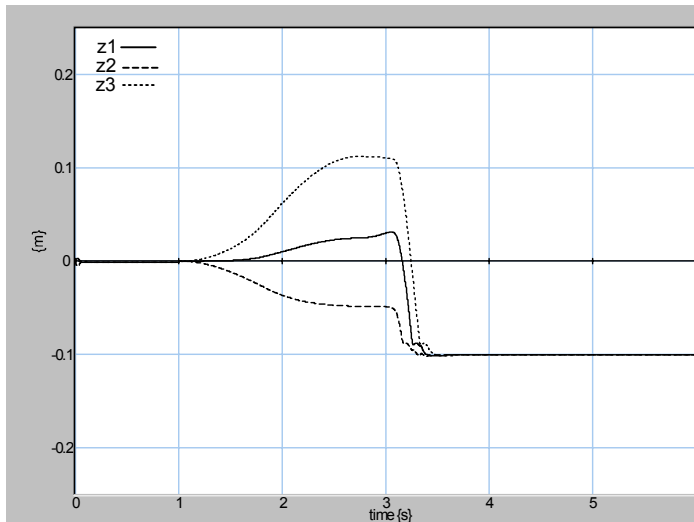


Fig. 5.6a Simulation of an emergency situation during the motion, the EMERGENCY button is pressed at $t=3$ sec. The translators move down to a safe position just above the lower end stops and wait until the power is cut off.

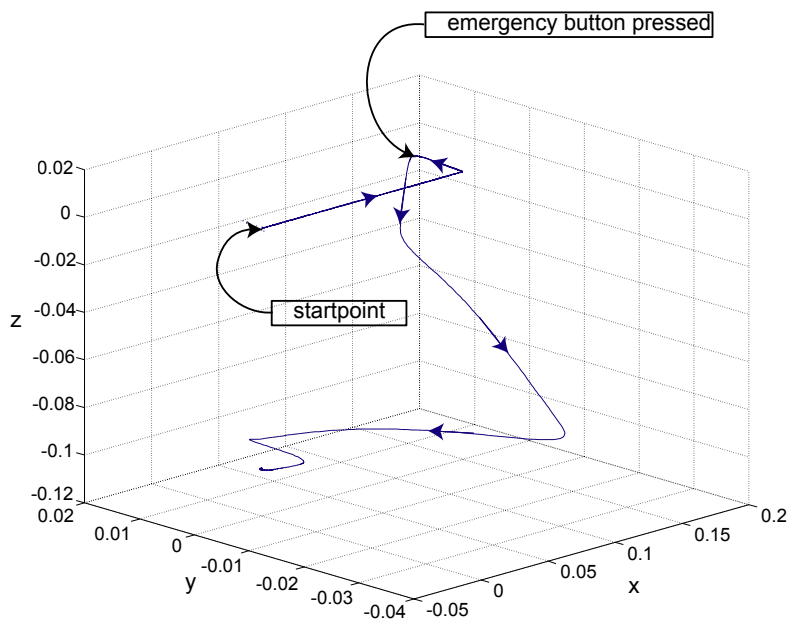


Fig. 5.6b 3D simulation of an emergency situation during the motion, the EMERGENCY button is pressed at $t=3$ sec.

From the plot in figure 5.6a, we see that the emergency mode works properly. At $t = 3$ sec. the EMERGENCY button is pressed. The speed of the translators is reduced and then they move to the lower safe position where $z_1 = z_2 = z_3 = -0.1$. Figure 5.6b shows that the platform moves towards the safe position. The followed path to get there looks a bit strange. The reason for this is the same as with the stop mode: the translators are slowed down and positioned independent from each other.

5.7 Verifying the alarm mode in simulation

To verify the alarm mode of the controller, first the reference path is modified such that the manipulator will exceed its safe work area. At the moment that this happens, the alarm mode should become active and the steering signals must be set to zero. The reference path that is used causes the platform to move to the point $\{0.3, 0.1, 0.0\}$. This point lies outside the safe work area and therefore the alarm mode becomes active. The results can be seen in the plots of figure 5.7.

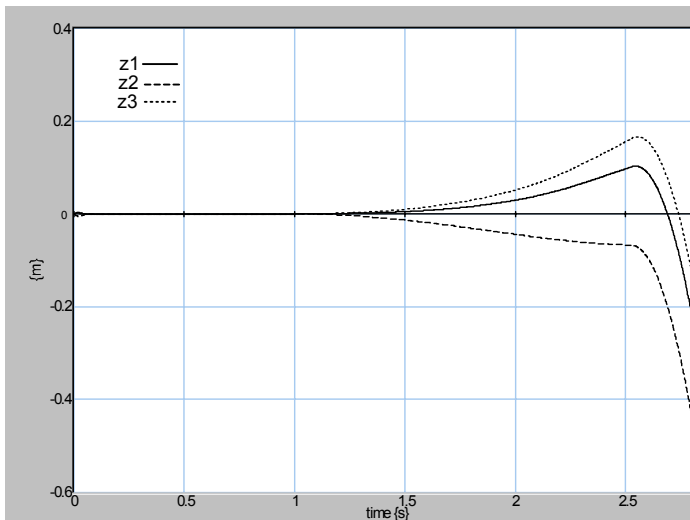


Fig. 5.7a Simulation of an alarm situation during the motion; the platform is forced to operate beyond its limits. At the moment that the platform reaches its limit, the steering signals are set to zero and the translators move down towards the end stops.

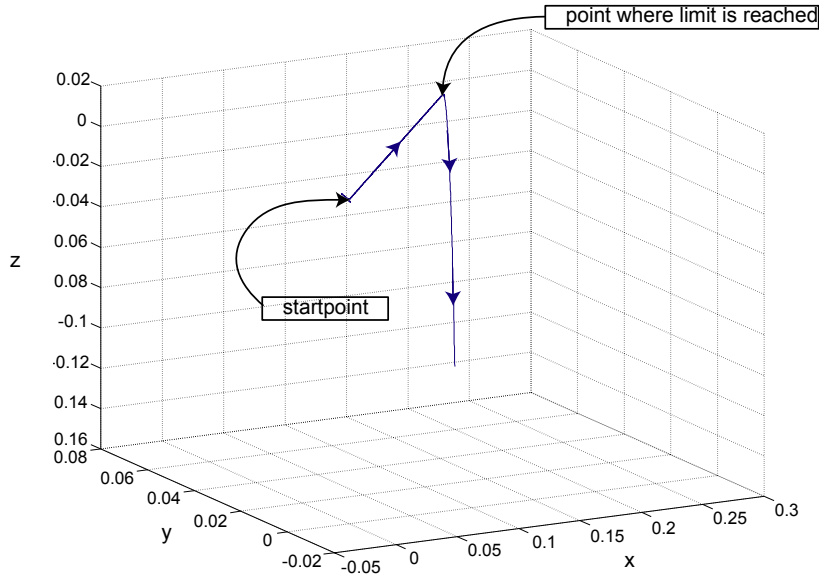


Fig. 5.7b 3D simulation of an alarm situation during the motion. When the platform reaches its limits, the alarm mode becomes active and switches off the power. The translators then move to the end stops.

From the simulation plots of figure 5.7, we see that the alarm mode works correctly for exceeding the safe work area. The steering signals are set to zero when the platform reaches its limits. The translators then move down due to gravity. In the plot is seen that the translators continue falling down. In the real system the end dampers stop them, but in the model this has not been incorporated.

5.8 Experimental results

The designed controller system has largely been implemented on the computing system of the manipulator. This means that the hierarchy of fig 3.10 is for a large part also realized in the actual real time control system. Compared to the simulated control system, the following differences are present:

- the StartupController, which was not included in simulations, has been implemented as a sequentially coordinated composite agent consisting of: a CheckStartAgent, an AlignAgent, a HomingAgent and a GoHomeAgent.
- the Alarm and GuardEmergency agents have not yet been included
- the Stop agent and the Shutdown agent have not yet been added.

The following controller settings are used which gave satisfactory results.

$$K = 89290 \text{ [N/m]}$$

$$T_d = 0.01264 \text{ [s]}$$

$$T_i = 0.2 \text{ [s]}$$

The motion according to fig 5.1 is performed with the manipulator. For safety reasons the path is slightly modified to lower accelerations and speeds. Therefore the motion time is larger. The results of performing the motion with the actual manipulator are given in the figure 5.8 till 5.10.

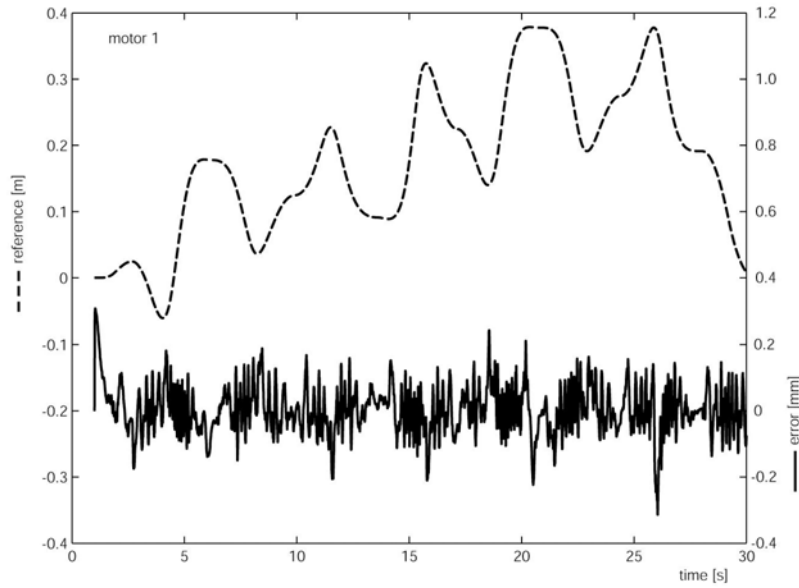


Fig. 5.8 Result of motor 1 with actual manipulator motion.

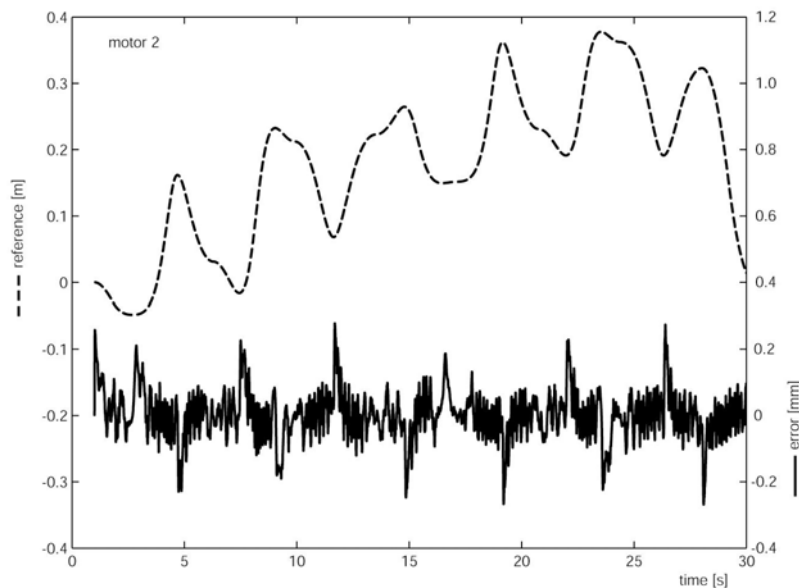


Fig. 5.9 Result of motor 2 with actual manipulator motion.

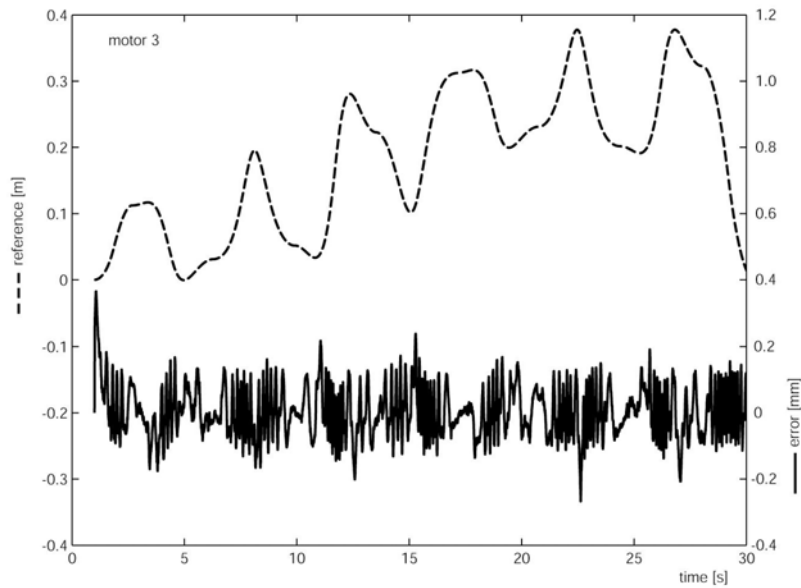


Fig. 5.10 Result of motor 3 with actual manipulator motion.

From the plots can be seen that the tracking error has peaks of somewhat larger than $250 \text{ } [\mu\text{m}]$. These peaks coincides with the moments where a velocity reversal occurs with the translators. This large peak can be explained by the phenomena of stiction. If we disregard the peaks, the tracking error is about $100 \text{ } [\mu\text{m}]$. This is conform expectations.

6 Conclusions and recommendations

6.1 Conclusions

The aim of the assignment was to design and realize a safe controller system for a parallel manipulator including the setpoint generation. The safe controller is designed and implemented according to the agent-based approach as introduced by [Van Breemen, 2000]. For doing this the tool IDITMAC is used. The tool provides a convenient design process and the designed software can easily be incorporated with the modelling and simulation environment 20-Sim for verification. The designed safe controller has been tested in simulation and the correct operation of the various modes (Standard, Alarm and Emergency) is verified. The obtained structure of the safe controller system is also applicable for other manipulators; only the conditions for agent activity will vary.

A safety analysis of the manipulator is done and has resulted in the following crucial points:

- Judging tracking error
- Judging platform position for not exceeding safe work area.
- Hardware safety circuitry

Furthermore, performing the commutation algorithm inside the computer system and not in the amplifiers makes the manipulator inherent safe.

The designed controller is largely implemented in the computer system of the Imotec manipulator. A slow, large stroke movement is performed which resulted in a maximum tracking error of 250 [μm]. This peak occurs at moments of velocity reversal. Therefore, the error peak can be explained as a result of stiction.

Because of time constraints, the experiments with Learning Feed Forward Control have not been carried out, but it is expected that adding the LFFC to the feedback loop will give a large improvement considering the stiction.

A general path specification tool is developed for defining reference paths for *xyz*-manipulators. It allows giving up a motion in segments. Supported motion profiles are trapezoidal and S-curve. For the moment, the shape of the reference paths can be straight line and arc movements. The defined path is stored in XML format for which a schema is available. The tool also creates a file with reference sample points as setpoint for the feedback controller.

6.2 Recommendations

In alarm situations of the safe controller, the steering signals are set to zero. This causes the translators to fall down on the end dampers (This also happens in power cut-off situations). It would be better to brake dynamically in those situations. The coils of the translators can be short circuited by means of a brake relay. This would result in a slow movement of the translator towards the end dampers.

The approximation of the platform position has a too large variation; the error that is made depends too much on the angle ϕ in cylindrical coordinates. A better approximation should be made that also takes ϕ in account.

The Function Approximator has to be added to the feedback controller and experiments must be done .

The path generator tool developed in this thesis provides basic facilities for path specification of manipulators. The following improvements can be made to the tool:

- Motion segments are currently from standstill to standstill. It is desirable that segments can start and end with a speed unequal to zero. This allows smooth continuous movements.
- Currently, only arc and straight line movements are supported. This should be extended with elliptical and spiral movements.
- Copy, cut and paste functions should be added to the tool for easy editing of path specifications.
- Checking the specified path by means of a 3d plot should occur within the tool itself.

Appendix A

Working principle and drive of linear motor.

A.1 Working principle of linear motor

The brushless permanent-magnet motor configuration consists of a base plate (stator), covered with permanent magnets, and a sliding part (translator) that holds the electric coils and their iron cores, see figure A.1. By applying a three-phase current to the coils, a sequence of attracting and repelling forces between the poles and the permanent magnets will be generated. An incremental linear encoder measures the position of the translator.

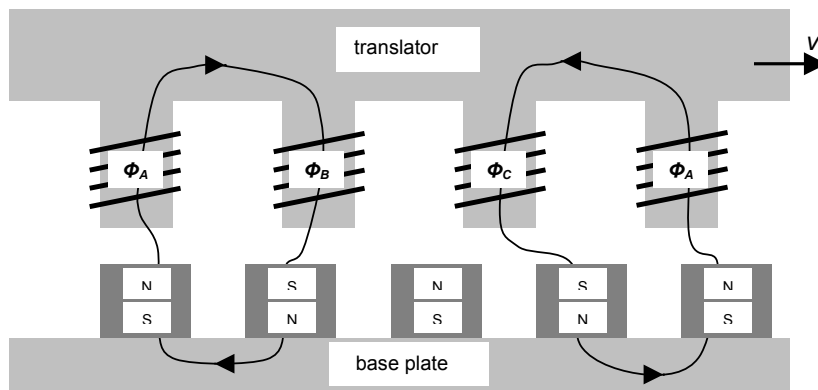


Figure A.1 Working principle of the linear motor. The indicated lines are the flux lines of the permanent magnets.

To increase the efficiency of the motor, iron cores are placed in the coils. These iron cores have a magnetic interaction with the permanent magnets on the base plate. This interaction, regardless whether there is a current in the coils or not, results in a force that tries to move the translator into stable detent positions. This phenomenon is known as cogging. Because of the spatial distribution of the magnets with a period of 12 [mm] a stable detent position is also found every 12 [mm].

A.2 Drive of linear motor

To obtain the right sequence of the attracting and repelling forces for the movement commutation is needed. There are several commutation methods, like trapezoidal and sine wave. The best performance is achieved with sine wave commutation, which is also the most computing effort costing. The currents through the individual coils depends on the needed force

for the movement and the relative position of the coil with respect to the magnets. In figure A.2 the principle of commutation is explained. In the figure only one coil is given. Because of the three-phase configuration of the coils the current through the other coils are 120° shifted. The colour of the coil gives the amount of current and the N or S symbol gives the pole, which depends on the sign of the current. Darker colours represent larger currents. The direction of the movement can be changed by applying a phase shift of 180° to the current. Because of the sine function this is the same as changing the sign of the reference current.

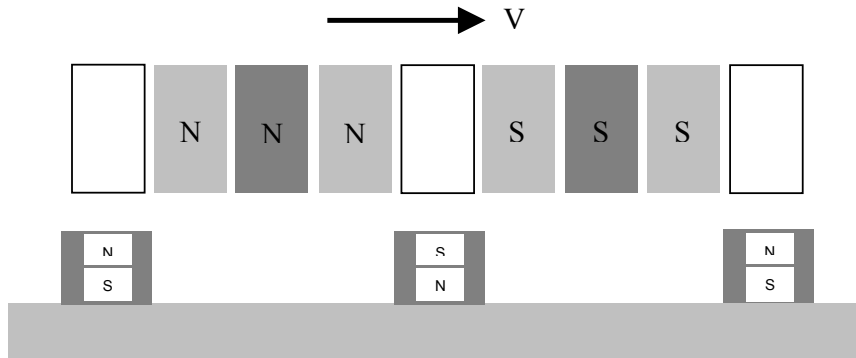


Figure A.2 Principle of commutation. The darkness of the coil represents the amount of current through it.

The commutation can be written out in formulas:

$$I_1 = I_{ref} \sin(Pos / 0.024 * 2\pi)$$

$$I_2 = I_{ref} \sin(Pos / 0.024 * 2\pi - \frac{2\pi}{3})$$

$$I_3 = I_{ref} \sin(Pos / 0.024 * 2\pi - \frac{4\pi}{3})$$

With Pos as the absolute position of the translators, which starts with zero exactly when coil number 1 is above a North pole magnet on the base plate. There are amplifiers on the market, which can determine the relative position of the coils with respect to the magnets by measuring the magnetic field. This is done by Hall sensors in the translator. The Tecnotion linear motor that is used for the manipulator doesn't have any Hall sensors. This means that before the commutation can start, a procedure has to be executed to bring the translator exactly with coil number one above a North pole. Which North pole doesn't matter, as long it is a North pole on the base plate. This procedure is called aligning. A feasible aligning method that can be performed with the used amplifiers is the so called "wake and shake" method. This goes as follows. Coil number one is fed with $0.5 I_{nom}$ and coil number two with $-0.25 I_{nom}$. In a three-phase system the three currents add up to a sum of zero, therefore a current of $-0.25 I_{nom}$ is also fed to coil number three by the amplifier. Coil number one has become a "strong" South pole and coil number two and three "weak" North poles. This causes a force that drives the translator exactly above the nearest North pole with coil number one. By holding on the currents for about half a second, the translator is aligned. With this position known the encoder can be reset and the commutation can start. The maximum displacement during aligning is 12 [mm].

Appendix B

Hardware overview of the manipulator

B.1 The electronic components

In figure B.1 a block schematic overview of the most important hardware components is given. The components will be treated in more detail.

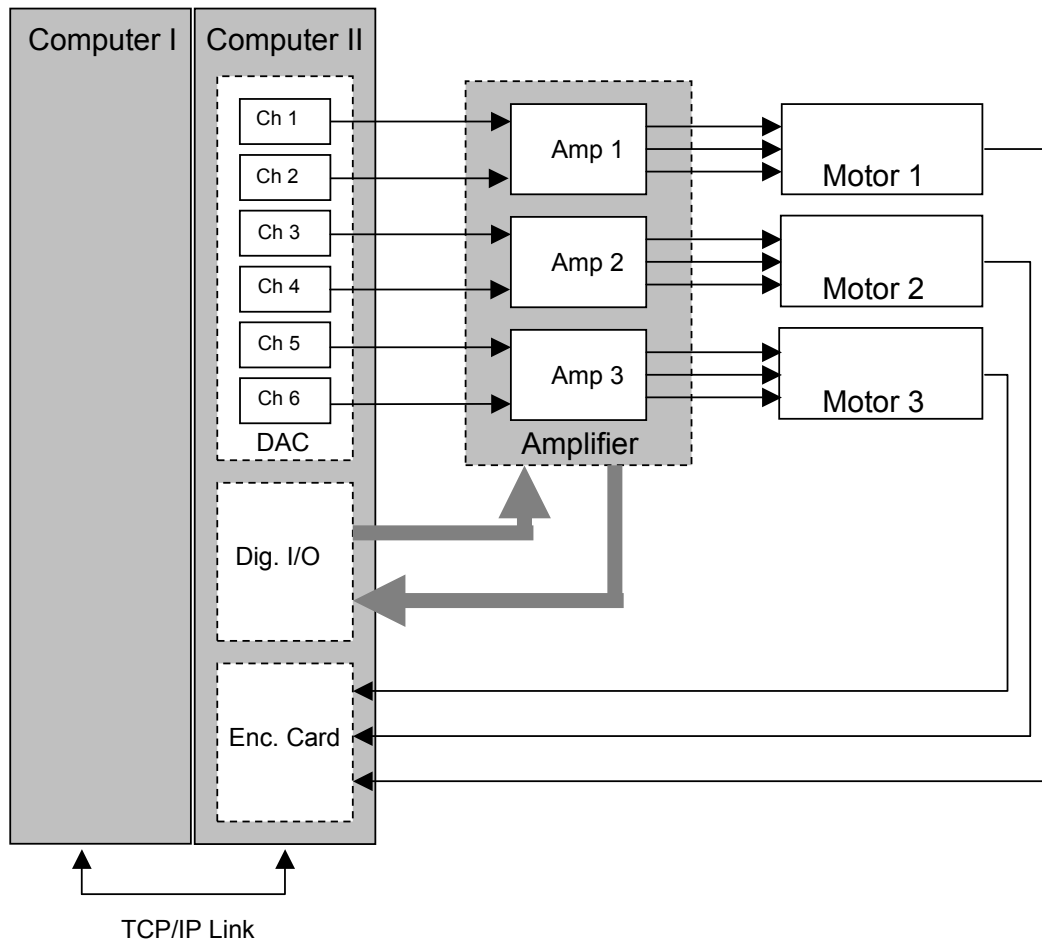


Fig. B.1 Overview of the electronic components.

B.2 The computer system

The computer system of the manipulator consists of two identical industrial pc's (IPC's) in one housing. The computers are linked via a TCP/IP Link. System I, a Windows NT machine, will

be used for running the Graphical User Interface (GUI) and system II for performing “realtime” control of the manipulator. System II is running under DOS. This is certainly not a realtime operating system but has proven to be a stable platform. The DOS machine contains three PC cards, A DAC plus some digital input, a digital I/O and an encoder card.

B.3 The amplifier

The used amplifiers are three TBL250/10 series in a 19” rack from MTS Automation. The amplifiers are relatively simple without any commutation logics or intelligence. This means that commutation to drive the motors must be performed in software. As an input reference for the three-phase output currents, only I_u and I_v are needed. The third reference current is calculated internally because the sum of the currents must be zero in a three-phase system. This saves out one DAC channel per motor. The amplifiers have a current-voltage ratio of 2. A maximum of ± 10 V reference results in a max output current of ± 20 A. The linear motors have a force-current ratio of 39. The force-voltage ratio is therefore 78.

B.4 The DAC card

The DAC card is an 8 channel 14-bit analog output board from ICP DAS. Only 6 of the 8 channels are used. It has a output modes of ± 10 Volt and ± 5 Volt. For the manipulator the ± 5 Volt mode is used. This results in an maximum output current for the amplifier of ± 10 Ampere, which is feasible for the motors.

B.5 The digital I/O card

The digital I/O card is from the manufacturer Isolation. It has 16 optically isolated inputs and 16 relay outputs. The input voltage is 5-24 volt AC or DC. The I/O card is used to enable the amplifiers, read in the operations button like ENABLE, START and STOP.

B.6 The Encoder card and the encoders.

The PCL-833 encoder card from Advantech reads in the 3 linear encoders of the motors. The encoders are from the manufacturer Numerik Jena and have a grating period of 20 [μm]. After the quadrature and the 5X interpolation factor this results in a resolution of 1.0 [μm].

Appendix C

An introduction to XML

C.1 The XML format

Extensible Markup Language (XML), defined by the World Wide Web Consortium (W3C), is a universal language for describing and exchanging data on the web. It is emerging as a new way to store and communicate data. Even though its primary application is as the future of the World Wide Web, it can be used in a variety of situations to structure digital data. XML is a powerful language that enables a user to store and communicate semi-structure data. XML, like HTML, is based on tags, and represents documents as trees of element. It also has two sorts of element: empty and non-empty elements. Moreover, the XML specification defines precise rules that make document parsing simple. XML specification defines two types of XML document: valid documents and well-formed documents. In what follows we describe some of the conditions that well-formed documents must fulfil.

The document instance must conform to the grammar of XML documents. In particular, some markup constructs are only allowed in specific places.

No attribute may appear more than once in the same start tag.

Attributes must be declared without ambiguity, notably attribute values must be enclosed between two similar quotation marks.

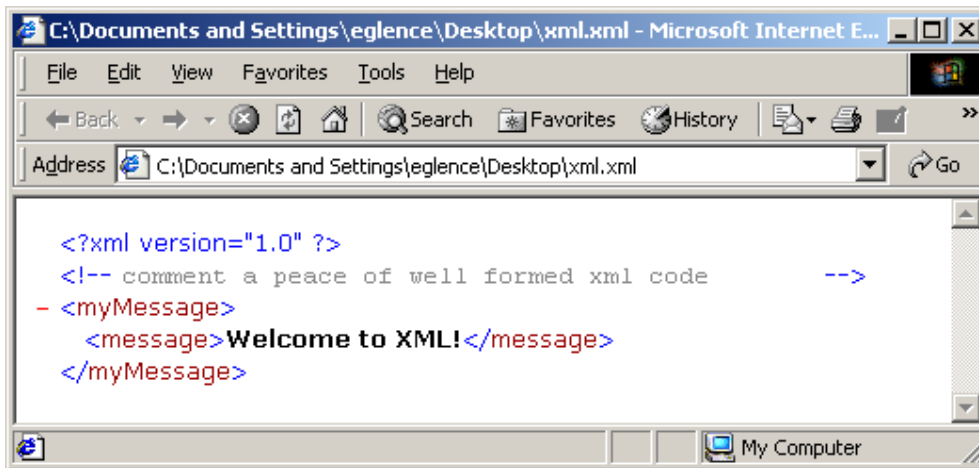
Non-empty element tags must be properly nested: any non-empty element must be closed by its end tag before its ancestors.

Empty element tags must contain a slash '/' just before the end bracket.

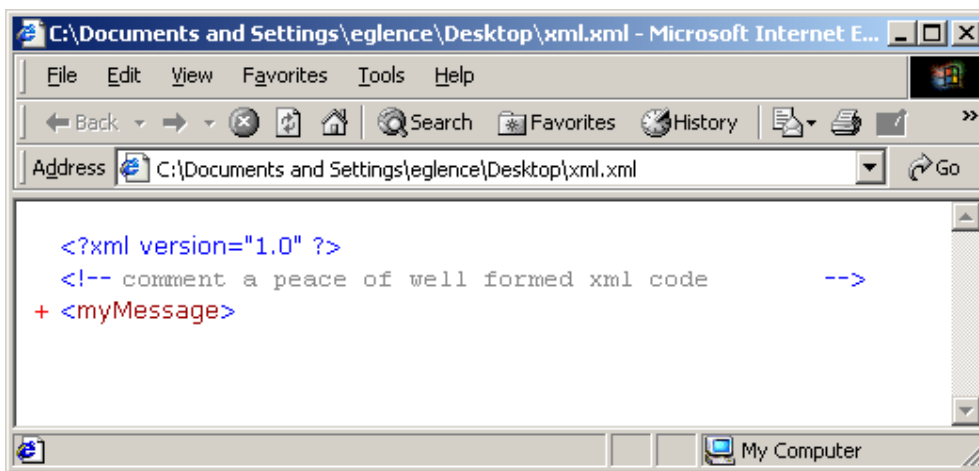
This is how well formed XML looks like:

```
<?xml version="1.0" ?>  
<!--comment a peace of well formed xml code -->  
<myMessage>  
    <message>Welcome to XML!</message>  
</myMessage>
```

The same code viewed with Internet Explorer:



```
<?xml version="1.0" ?>
<!-- comment a peace of well formed xml code -->
- <myMessage>
  <message>Welcome to XML!</message>
</myMessage>
```



```
<?xml version="1.0" ?>
<!-- comment a peace of well formed xml code -->
+ <myMessage>
```

C.2 XML Schemas

Schemas are to define an XML document's structure, but an XML document is not required to have a corresponding schema. However, schemas are often recommended to ensure document conformity. Schemas specify an XML document's structure and are themselves also defined using XML.

An XML document that conforms to a schema document is valid and a document that does not conform is invalid. Now, there are two major types of schema models, one created by Microsoft and the other by W3C. In this thesis, we use the W3C XML Schemas to describe the structure and element content of our XML documents for storing path information.

C.3 Structure of the XML path file

The XML path file that is stored and converted to a reference sample file has a specified structure. It could be modified directly in a text editor, but it is recommended to do this only within the Path Generator tool. The path file is validated against his schema “move.xsd”. There is no limit to the number of motion segments and waits that can be added to a motion. Beneath an example of a stored reference path is given in xml format.

```
<?xml version="1.0"?>
<!-- Created with XML Path Generator Imotec bv -->
<!-- Created on 5-3-2003 at 21:02:29 -->
<move xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="move.xsd">
  <description>Demonstrator path for Imotec manipulator</description>
  <starting_point dim="3">
    <coord>0.0</coord>
    <coord>0.0</coord>
    <coord>0.0</coord>
  </starting_point>
  <composite_path>
    <path>
      <segment>
        <motion>
          <scurve>
            <max_v>.5</max_v>
            <max_a>.2</max_a>
            <max_d>.2</max_d>
            <max_jerk>100</max_jerk>
          </scurve>
        </motion>
        <shape>
          <straight>
            <distance dim="3">
              <coord>.15</coord>
              <coord>0</coord>
              <coord>0</coord>
            </distance>
          </straight>
        </shape>
      </segment>
    </path>
    <wait>0.75</wait>
    <path>
      <segment>
        <motion>
          <trapezoid>
            <max_v>.5</max_v>
            <max_a>.2</max_a>
            <max_d>.2</max_d>
          </trapezoid>
        </motion>
        <shape>
          <arc>
            <distance dim="3">
              <coord>-.15</coord>
              <coord>.15</coord>
              <coord>.05</coord>
            </distance>
            <angle>less than_pi</angle>
            <radius>.15</radius>
            <aux_point dim="3">
              <coord>-.15</coord>
              <coord>0</coord>
              <coord>.025</coord>
            </aux_point>
          </arc>
        </shape>
      </segment>
    </path>
  </composite_path>
</move>
```


Appendix D

Controller settings

D.1 Identifying the plant model

For obtaining a plant model to come to a controller design, the manipulator is divided in to three identical parts. Solving the design for one part gives a solution to the total control problem of the manipulator. Such a part that is used as a model consists of one motor attached to the plateau by the arms. This results in a fourth order plant model. It is expected that the dominant stiffness of this model is located in the arms and joints. In figure D.1 an iconic diagram of the model is given.

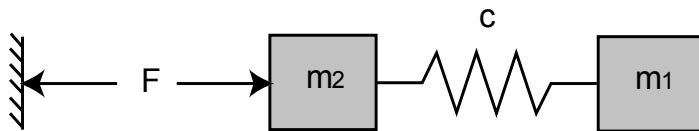


Fig. D.1 Iconic diagram of the model that is used for controller design.

This type of model is called a Flexible Mechanism. The following parameters are given:

Mass of the end-effector with load: $m_1 = 1 \dots 6$ [kg]

Mass of the motor: $m_2 = 2$ [kg]

Total mass $m = 8$ [kg]

The value of the spring constant c for the arms and joints are not known because of the complex construction. The position measurement is done on the motor. This leads to a concept AR transfer function. In the next section a PID feedback compensator will be designed for the model with unknown parameter c .

D.2 PID compensator design

For designing the PID compensator, the procedure as described in [Coelingh, 2000] is followed.

The steps for this procedure are:

1. Determine the desired bandwidth ω_b .
2. Determine the total mass to be displaced.
3. Determine the amount of phase lead by choosing β , a practical value for β is 0.1 which gives a phase lead of 55° .
4. Determine the value of the derivative time constant T_d according to:

$$T_d = \frac{1}{2\omega_b\sqrt{\beta}} \quad [\text{D-1}]$$

5. Determine the proportional gain K to obtain the desired bandwidth ω_b by:

$$K = 2\omega_b^2 m \left(\frac{\omega_b^2 T_d^2 \beta + 1}{\omega_b^2 T_d^2 + 1} \right) \quad [\text{D-2}]$$

6. Determine the integral time constant T_i in order to obtain a desired gain at low frequencies. Avoid interfering with the derivative and proportional action.
7. Determine the high frequency roll-off time constant T_h to suppress the disturbance of noise by:

$$T_h < \beta T_d \quad [\text{D-3}]$$

The desired bandwidth ω_b cannot be specified well due to uncertainty in the spring constant of the arms. Therefore, a number of controller settings are calculated for different bandwidths. These can be implemented in the actual controller, starting with the lower bandwidth parameters and improving to higher bandwidths without oscillating. In table D.1 the parameters are given for the different bandwidths.

ω_b [rad/s]	K [N/m]	T_d [s]	T_i [s]	T_h [s]
50	14286	0.0316	0.4	$18.6 \cdot 10^{-5}$
100	57143	0.0158	0.2	$9.5 \cdot 10^{-5}$
150	128570	0.0105	0.2	$6.3 \cdot 10^{-5}$
200	228570	0.0079	0.1	$4.7 \cdot 10^{-5}$
250	357140	0.0063	0.1	$3.8 \cdot 10^{-5}$

Table D.1 Parameters of the controller settings for different desired bandwidths.

Appendix E

Total MAC system XML source code

E.1 The Main agent XML code

```
<?xml version="1.0"?>
<mac xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="MacSchema.xsd">
  <name>TriPod2</name>
  <interface>
    <ports>
      <input>
        <type>TwenteSensor</type>
        <name>encoder1</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>encoder2</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>encoder3</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>stopbutton</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>startbutton</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>emergencybutton</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>s1up</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>s1down</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>s2up</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>s2down</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>s3up</name>
      </input>
      <input>
        <type>TwenteSensor</type>
        <name>s3down</name>
      </input>
      <output>
        <type>TwenteActuator</type>
        <name>voltage1</name>
      </output>
      <output>
        <type>TwenteActuator</type>
        <name>voltage2</name>
      </output>
    </ports>
  </interface>
</mac>
```

```

        <output>
          <type>TwenteActuator</type>
          <name>voltage3</name>
        </output>
      </ports>
    </interface>
    <implementation>
      <composite>
        <cagency>
          <cagent>
            <type>OverallController</type>
            <name>overallController</name>
          </cagent>
        </cagency>
        <connections>
          <connection>
            <from>stopbutton.output</from>
            <to>overallController.stopbutton</to>
          </connection>
          <connection>
            <from>startbutton.output</from>
            <to>overallController.startbutton</to>
          </connection>
          <connection>
            <from>emergencybutton.output</from>
            <to>overallController.emergencybutton</to>
          </connection>
          <connection>
            <from>encoder3.output</from>
            <to>overallController.Z3</to>
          </connection>
          <connection>
            <from>slup.output</from>
            <to>overallController.slup</to>
          </connection>
          <connection>
            <from>s1down.output</from>
            <to>overallController.s1down</to>
          </connection>
          <connection>
            <from>s2up.output</from>
            <to>overallController.s2up</to>
          </connection>
          <connection>
            <from>s2down.output</from>
            <to>overallController.s2down</to>
          </connection>
          <connection>
            <from>s3up.output</from>
            <to>overallController.s3up</to>
          </connection>
          <connection>
            <from>s3down.output</from>
            <to>overallController.s3down</to>
          </connection>
          <connection>
            <from>overallController.voltage1</from>
            <to>voltage1.input</to>
          </connection>
          <connection>
            <from>overallController.voltage2</from>
            <to>voltage2.input</to>
          </connection>
          <connection>
            <from>overallController.voltage3</from>
            <to>voltage3.input</to>
          </connection>
          <connection>
            <from>encoder1.output</from>
            <to>overallController.Z1</to>
          </connection>
          <connection>
            <from>encoder2.output</from>
            <to>overallController.Z2</to>
          </connection>
        </connections>
      </composite>
    </implementation>
  </interface>

```

```

        </implementation>
    </mac>

```

E.2 The OverallAgent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>OverallController</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <input>
                <type>real</type>
                <name>s1up</name>
            </input>
            <input>
                <type>real</type>
                <name>s1down</name>
            </input>
            <input>
                <type>real</type>
                <name>s2up</name>
            </input>
            <input>
                <type>real</type>
                <name>s2down</name>
            </input>
            <input>
                <type>real</type>
                <name>s3up</name>
            </input>
            <input>
                <type>real</type>
                <name>s3down</name>
            </input>
            <input>
                <type>real</type>
                <name>emergencybutton</name>
            </input>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>startbutton</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage1</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage2</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage3</name>
            </output>
        </ports>
    </interface>
</cagentclass>

```

```

</interface>
<implementation>
  <composite>
    <agency>
<!-- Insert Startup agent here -->
      <cagent>
        <type>Alarm</type>
        <name>alarm</name>
      </cagent>
      <cagent>
        <type>GuardedEmergency</type>
        <name>emergency</name>
      </cagent>
      <cagent>
        <type>GuardedStandard</type>
        <name>standard</name>
      </cagent>
    </agency>
    <coordination>
      <class>FixedPriorityCoordinator</class>
      <name>fixedPriorityM01</name>
    </coordination>
    <connections>
      <connection>
        <from>Z1</from>
        <to>alarm.Z1</to>
      </connection>
      <connection>
        <from>Z2</from>
        <to>alarm.Z2</to>
      </connection>
      <connection>
        <from>Z3</from>
        <to>alarm.Z3</to>
      </connection>
      <connection>
        <from>alarm.voltage1</from>
        <to>voltage1</to>
      </connection>
      <connection>
        <from>alarm.voltage2</from>
        <to>voltage2</to>
      </connection>
      <connection>
        <from>alarm.voltage3</from>
        <to>voltage3</to>
      </connection>
      <connection>
        <from>emergencybutton</from>
        <to>emergency.emergencybutton</to>
      </connection>
      <connection>
        <from>slup</from>
        <to>alarm.slup</to>
      </connection>
      <connection>
        <from>s1down</from>
        <to>alarm.s1down</to>
      </connection>
      <connection>
        <from>s2up</from>
        <to>alarm.s2up</to>
      </connection>
      <connection>
        <from>s2down</from>
        <to>alarm.s2down</to>
      </connection>
      <connection>
        <from>s3up</from>
        <to>alarm.s3up</to>
      </connection>
      <connection>
        <from>s3down</from>
        <to>alarm.s3down</to>
      </connection>
      <connection>
        <from>Z1</from>

```



```

        <to>emergency.Z1</to>
    </connection>
</connection>
<connection>
    <from>Z2</from>
    <to>emergency.Z2</to>
</connection>
<connection>
    <from>Z3</from>
    <to>emergency.Z3</to>
</connection>
<connection>
    <from>emergency.voltage1</from>
    <to>voltage1</to>
</connection>
<connection>
    <from>emergency.voltage2</from>
    <to>voltage2</to>
</connection>
<connection>
    <from>emergency.voltage3</from>
    <to>voltage3</to>
</connection>
<connection>
    <from>Z1</from>
    <to>standard.Z1</to>
</connection>
<connection>
    <from>Z2</from>
    <to>standard.Z2</to>
</connection>
<connection>
    <from>Z3</from>
    <to>standard.Z3</to>
</connection>
<connection>
    <from>stopbutton</from>
    <to>standard.stopbutton</to>
</connection>
<connection>
    <from>startbutton</from>
    <to>standard.startbutton</to>
</connection>
<connection>
    <from>standard.voltage1</from>
    <to>voltage1</to>
</connection>
<connection>
    <from>standard.voltage2</from>
    <to>voltage2</to>
</connection>
<connection>
    <from>standard.voltage3</from>
    <to>voltage3</to>
</connection>
</connections>
</composite>
</implementation>
</cagentclass>

```

E.3 The Alarm agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>Alarm</name>
    <include><![CDATA[<cmath>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
        </ports>
    </interface>
</cagentclass>

```

```

</input>
<input>
  <type>real</type>
  <name>Z2</name>
</input>
<input>
  <type>real</type>
  <name>Z3</name>
</input>
<input>
  <type>real</type>
  <name>s1up</name>
</input>
<input>
  <type>real</type>
  <name>s1down</name>
</input>
<input>
  <type>real</type>
  <name>s2up</name>
</input>
<input>
  <type>real</type>
  <name>s2down</name>
</input>
<input>
  <type>real</type>
  <name>s3up</name>
</input>
<input>
  <type>real</type>
  <name>s3down</name>
</input>
<output>
  <type>real</type>
  <name>voltage1</name>
</output>
<output>
  <type>real</type>
  <name>voltage2</name>
</output>
<output>
  <type>real</type>
  <name>voltage3</name>
</output>
</ports>
<parameterdefs>
  <parameterdef>
    <type>real</type>
    <name>Tm</name>
    <defaultvalue>1.0</defaultvalue>
  </parameterdef>
  <parameterdef>
    <type>real</type>
    <name>Td</name>
    <defaultvalue>2.0</defaultvalue>
  </parameterdef>
  <parameterdef>
    <type>real</type>
    <name>Ho</name>
    <defaultvalue>0.0</defaultvalue>
  </parameterdef>
  <parameterdef>
    <type>real</type>
    <name>Hm</name>
    <defaultvalue>1.0</defaultvalue>
  </parameterdef>
  <parameterdef>
    <type>real</type>
    <name>samplingtime</name>
    <defaultvalue>0.001</defaultvalue>
  </parameterdef>
</parameterdefs>
</interface>
<implementation>
  <elementary>
    <states>

```

```

        <state>
            <type>boolean</type>
            <name>goAktiv</name>
        </state>
        <state>
            <type>real</type>
            <name>zcalc</name>
        </state>
        <state>
            <type>real</type>
            <name>radc</name>
        </state>
    </states>
    <start><![CDATA[
{
    zcalc = 0.0;
    radc = 0.0;
    goAktiv = false;
}
]]></start>
    <initialize><![CDATA[
{
    voltage1 = 0.0;
    voltage2 = 0.0;
    voltage3 = 0.0;
}
]]></initialize>
    <activation><![CDATA[
{
    if (goAktiv)
        return 1.0;
    else
        return 0.0;
}
]]></activation>
    <calculate><![CDATA[
{
    voltage1 = 0.0;
    voltage2 = 0.0;
    voltage3 = 0.0;
}
]]></calculate>
    <update><![CDATA[
{
    radc = 0.82 * sqrt(Z1*Z1 - Z1*Z2 - Z1*Z3 + Z2*Z2 - Z2*Z3 + Z3*Z3);

    zcalc = (Z1 + Z2 + Z3)/3.0 + 1.355*( 0.202681 - radc*radc) - 0.301201
    if(radc>0.17 || zcalc>0.234)
        goAktiv = true;
}
]]></update>
</elementary>
</implementation>
</cagentclass>

```

E.4 The GuardedEmergency agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>GuardedEmergency</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>emergencybutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
        </ports>
    </interface>
</cagentclass>

```

```

        </input>
        <input>
            <type>real</type>
            <name>Z2</name>
        </input>
        <input>
            <type>real</type>
            <name>Z3</name>
        </input>
        <output>
            <type>real</type>
            <name>voltage1</name>
        </output>
        <output>
            <type>real</type>
            <name>voltage2</name>
        </output>
        <output>
            <type>real</type>
            <name>voltage3</name>
        </output>
    </ports>
</interface>
<implementation>
    <composite>
        <cagency>
            <cagent>
                <type>Emergency</type>
                <name>em1</name>
            </cagent>
            <cagent>
                <type>ErrorGuard</type>
                <name>eg</name>
            </cagent>
        </cagency>
        <coordination>
            <class>MasterSlaveCoordinator</class>
            <name>c1</name>
        </coordination>
        <connections>
            <connection>
                <from>emergencybutton</from>
                <to>em1.emergencybutton</to>
            </connection>
            <connection>
                <from>Z1</from>
                <to>em1.Z1</to>
            </connection>
            <connection>
                <from>Z2</from>
                <to>em1.Z2</to>
            </connection>
            <connection>
                <from>Z3</from>
                <to>em1.Z3</to>
            </connection>
            <connection>
                <from>em1.voltage1</from>
                <to>eg.voltageIn1</to>
            </connection>
            <connection>
                <from>em1.voltage2</from>
                <to>eg.voltageIn2</to>
            </connection>
            <connection>
                <from>em1.voltage3</from>
                <to>eg.voltageIn3</to>
            </connection>
            <connection>
                <from>em1.error1</from>
                <to>eg.error1</to>
            </connection>
            <connection>
                <from>em1.error2</from>
                <to>eg.error2</to>
            </connection>
        </connections>
    </composite>
</implementation>
</class>

```

```

        <from>em1.error3</from>
        <to>eg.error3</to>
    </connection>
    <connection>
        <from>eg.voltageOut1</from>
        <to>voltage1</to>
    </connection>
    <connection>
        <from>eg.voltageOut2</from>
        <to>voltage2</to>
    </connection>
    <connection>
        <from>eg.voltageOut3</from>
        <to>voltage3</to>
    </connection>
</connections>
</composite>
</implementation>
</cagentclass>

```

E.5 The ErrorGuard agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
  <name>ErrorGuard</name>
  <include><![CDATA[<math>]]></include>
  <interface>
    <ports>
      <input>
        <type>real</type>
        <name>voltageIn1</name>
      </input>
      <input>
        <type>real</type>
        <name>voltageIn2</name>
      </input>
      <input>
        <type>real</type>
        <name>voltageIn3</name>
      </input>
      <input>
        <type>real</type>
        <name>error1</name>
      </input>
      <input>
        <type>real</type>
        <name>error2</name>
      </input>
      <input>
        <type>real</type>
        <name>error3</name>
      </input>
      <output>
        <type>real</type>
        <name>voltageOut1</name>
      </output>
      <output>
        <type>real</type>
        <name>voltageOut2</name>
      </output>
      <output>
        <type>real</type>
        <name>voltageOut3</name>
      </output>
    </ports>
    <parameterdefs>
      <parameterdef>
        <type>real</type>
        <name>bound</name>
        <defaultvalue>0.005</defaultvalue>
      </parameterdef>

```

```

        </parameterdefs>
</interface>
<implementation>
  <elementary>
    <states>
      <state>
        <type>real</type>
        <name>factor</name>
      </state>
    </states>
    <start><![CDATA[
{
  factor = 1.0;
}
]]></start>
    <activation><![CDATA[
{
  return 1.0;
}
]]></activation>
    <calculate><![CDATA[ {
if (error1>=bound)
  factor = 0.0;
if (error2>=bound)
  factor = 0.0;
if (error3>=bound)
  factor = 0.0;

voltageOut1 = factor * voltageIn1;
voltageOut2 = factor * voltageIn2;
voltageOut3 = factor * voltageIn3;
}
]]></calculate>
  </elementary>
</implementation>
</cagentclass>

```

E.6 The Emergency agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
  <name>Emergency</name>
  <interface>
    <ports>
      <input>
        <type>real</type>
        <name>emergencybutton</name>
      </input>
      <input>
        <type>real</type>
        <name>Z1</name>
      </input>
      <input>
        <type>real</type>
        <name>Z2</name>
      </input>
      <input>
        <type>real</type>
        <name>Z3</name>
      </input>
      <output>
        <type>real</type>
        <name>voltage1</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage2</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage3</name>
      </output>
    </ports>
  </interface>
</cagentclass>

```

```

        <output>
          <type>real</type>
          <name>error1</name>
        </output>
        <output>
          <type>real</type>
          <name>error2</name>
        </output>
        <output>
          <type>real</type>
          <name>error3</name>
        </output>
      </ports>
    </interface>
    <implementation>
      <composite>
        <cagency>
          <cagent>
            <type>Brake</type>
            <name>brakeEmerg</name>
            <parameters>
              <parameter>
                <name>sampletime</name>
                <value>0.001</value>
              </parameter>
              <parameter>
                <name>K</name>
                <value>5.0</value>
              </parameter>
              <parameter>
                <name>BW</name>
                <value>5.0</value>
              </parameter>
            </parameters>
          </cagent>
          <cagent>
            <type>GoSteadyAllEmerg</type>
            <name>goSteadyAllEmerg</name>
          </cagent>
        </cagency>
      </composite>
      <coordination>
        <class>FixedPriorityCoordinator</class>
        <name>fixedPriorityM06</name>
      </coordination>
      <connections>
        <connection>
          <from>emergencybutton</from>
          <to>brakeEmerg.activationinput</to>
        </connection>
        <connection>
          <from>Z1</from>
          <to>brakeEmerg.Z1</to>
        </connection>
        <connection>
          <from>Z2</from>
          <to>brakeEmerg.Z2</to>
        </connection>
        <connection>
          <from>Z3</from>
          <to>brakeEmerg.Z3</to>
        </connection>
        <connection>
          <from>brakeEmerg.voltage1</from>
          <to>voltage1</to>
        </connection>
        <connection>
          <from>brakeEmerg.voltage2</from>
          <to>voltage2</to>
        </connection>
        <connection>
          <from>brakeEmerg.voltage3</from>
          <to>voltage3</to>
        </connection>
        <connection>
          <from>brakeEmerg.error1</from>
          <to>error1</to>
        </connection>
      </connections>
    </implementation>
  </cmodel>

```

```

        <connection>
            <from>brakeEmerg.error2</from>
            <to>error2</to>
        </connection>
        <connection>
            <from>brakeEmerg.error3</from>
            <to>error3</to>
        </connection>
        <connection>
            <from>emergencybutton</from>
            <to>goSteadyAllEmerg.emergencybutton</to>
        </connection>
        <connection>
            <from>Z1</from>
            <to>goSteadyAllEmerg.Z1</to>
        </connection>
        <connection>
            <from>Z2</from>
            <to>goSteadyAllEmerg.Z2</to>
        </connection>
        <connection>
            <from>Z3</from>
            <to>goSteadyAllEmerg.Z3</to>
        </connection>
        <connection>
            <from>goSteadyAllEmerg.voltage1</from>
            <to>voltage1</to>
        </connection>
        <connection>
            <from>goSteadyAllEmerg.voltage2</from>
            <to>voltage2</to>
        </connection>
        <connection>
            <from>goSteadyAllEmerg.voltage3</from>
            <to>voltage3</to>
        </connection>
        <connection>
            <from>goSteadyAllEmerg.error1</from>
            <to>error1</to>
        </connection>
        <connection>
            <from>goSteadyAllEmerg.error2</from>
            <to>error2</to>
        </connection>
        <connection>
            <from>goSteadyAllEmerg.error3</from>
            <to>error3</to>
        </connection>
    </connections>
</composite>
</implementation>
</cagentclass>

```

E.7 The Brake agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>Brake</name>
    <include><![CDATA[<cmath>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>activationinput</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>

```



```

        <name>Z2</name>
    </input>
    <input>
        <type>real</type>
        <name>Z3</name>
    </input>
    <output>
        <type>real</type>
        <name>voltage1</name>
    </output>
    <output>
        <type>real</type>
        <name>voltage2</name>
    </output>
    <output>
        <type>real</type>
        <name>voltage3</name>
    </output>
    <output>
        <type>real</type>
        <name>error1</name>
    </output>
    <output>
        <type>real</type>
        <name>error2</name>
    </output>
    <output>
        <type>real</type>
        <name>error3</name>
    </output>
</ports>
<parameterdefs>
    <parameterdef>
        <type>real</type>
        <name>sampletime</name>
        <defaultvalue>0.001</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>K</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>BW</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
</parameterdefs>
</interface>
<implementation>
    <elementary>
        <states>
            <state>
                <type>boolean</type>
                <name>isActive</name>
            </state>
            <state>
                <type>real</type>
                <name>previousZ1</name>
            </state>
            <state>
                <type>real</type>
                <name>previousZ2</name>
            </state>
            <state>
                <type>real</type>
                <name>previousZ3</name>
            </state>
            <state>
                <type>real</type>
                <name>speed1</name>
            </state>
            <state>
                <type>real</type>
                <name>speed2</name>
            </state>
        </states>
    </elementary>
</implementation>

```

```

        <type>real</type>
        <name>speed3</name>
    </state>
    <state>
        <type>real</type>
        <name>refspeed1</name>
    </state>
    <state>
        <type>real</type>
        <name>refspeed2</name>
    </state>
    <state>
        <type>real</type>
        <name>refspeed3</name>
    </state>
    <state>
        <type>real</type>
        <name>speedstep1</name>
    </state>
    <state>
        <type>real</type>
        <name>speedstep2</name>
    </state>
    <state>
        <type>real</type>
        <name>speedstep3</name>
    </state>
    <state>
        <type>real</type>
        <name>s1y1</name>
    </state>
    <state>
        <type>real</type>
        <name>s2y1</name>
    </state>
    <state>
        <type>real</type>
        <name>s1y2</name>
    </state>
    <state>
        <type>real</type>
        <name>s2y2</name>
    </state>
    <state>
        <type>real</type>
        <name>s1y3</name>
    </state>
    <state>
        <type>real</type>
        <name>s2y3</name>
    </state>
    <state>
        <type>real</type>
        <name>BWrad</name>
    </state>
    <state>
        <type>real</type>
        <name>y1</name>
    </state>
    <state>
        <type>real</type>
        <name>y2</name>
    </state>
    <state>
        <type>real</type>
        <name>y3</name>
    </state>
</states>
<start><![CDATA[
{
    BWrad = BW*2*3.1415926536;
    isActive = false;
}
]]></start>
<initialize><![CDATA[
{
    speedstep1 = (Z1 - previousZ1) / sampletime;

```

```

speedstep2 = (Z2 - previousZ2) / sampletime;
speedstep3 = (Z3 - previousZ3) / sampletime;

y1 = 0;
y2 = 0;
y3 = 0;

voltage1 = 0.0;
voltage2 = 0.0;
voltage3 = 0.0;
}
        ]]></initialize>
        <finalize><![CDATA[
{
    speed1 = 0.0;
    speed2 = 0.0;
    speed3 = 0.0;
}
        ]]></finalize>
        <activation><![CDATA[
{
    if (isAktive)
        return 1.0;
    else
        return 0.0;
}
        ]]></activation>
        <calculate><![CDATA[
{
    s2y1 = BWrads*BWrads*(speedstep1 - y1) - 1.4142*BWrads*s1y1;
    s1y1 = s1y1 + s2y1*sampletime;
    y1 = y1 + s1y1*sampletime;

    s2y2 = BWrads*BWrads*(speedstep2 - y2) - 1.4142*BWrads*s1y2;
    s1y2 = s1y2 + s2y2*sampletime;
    y2 = y2 + s1y2*sampletime;

    s2y3 = BWrads*BWrads*(speedstep3 - y3) - 1.4142*BWrads*s1y3;
    s1y3 = s1y3 + s2y3*sampletime;
    y3 = y3 + s1y3*sampletime;

    refspeed1 = speedstep1 - y1;
    refspeed2 = speedstep2 - y2;
    refspeed3 = speedstep3 - y3;

    speed1 = (Z1 - previousZ1) / sampletime;
    speed2 = (Z2 - previousZ2) / sampletime;
    speed3 = (Z3 - previousZ3) / sampletime;

    error1 = refspeed1 - speed1;
    error2 = refspeed2 - speed2;
    error3 = refspeed3 - speed3;

    voltage1 = error1 * K;
    voltage2 = error2 * K;
    voltage3 = error3 * K;

    if (fabs(speed1)< 0.02 && fabs(speed2)<0.02 && fabs(speed2)<0.02)
        isAktive = false;
}
        ]]></calculate>
        <update><![CDATA[
{
    if (activationinput == 1.0)
        isAktive = true;

    speed1 = (Z1 - previousZ1) / sampletime;
    speed2 = (Z2 - previousZ2) / sampletime;
    speed3 = (Z3 - previousZ3) / sampletime;

    previousZ1 = Z1;
    previousZ2 = Z2;
    previousZ3 = Z3;
}
        ]]></update>
</elementary>

```

```

    </implementation>
</cagentclass>

```

E.8 The GosteadyAllEmerg agent code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
  <name>GoSteadyAllEmerg</name>
  <interface>
    <ports>
      <input>
        <type>real</type>
        <name>emergencybutton</name>
      </input>
      <input>
        <type>real</type>
        <name>Z1</name>
      </input>
      <input>
        <type>real</type>
        <name>Z2</name>
      </input>
      <input>
        <type>real</type>
        <name>Z3</name>
      </input>
      <output>
        <type>real</type>
        <name>voltage1</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage2</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage3</name>
      </output>
      <output>
        <type>real</type>
        <name>error1</name>
      </output>
      <output>
        <type>real</type>
        <name>error2</name>
      </output>
      <output>
        <type>real</type>
        <name>error3</name>
      </output>
    </ports>
  </interface>
  <implementation>
    <composite>
      <cagency>
        <cagent>
          <type>GoSteadyEmerg</type>
          <name>goSteady1Emerg</name>
        </cagent>
        <cagent>
          <type>GoSteadyEmerg</type>
          <name>goSteady2Emerg</name>
        </cagent>
        <cagent>
          <type>GoSteadyEmerg</type>
          <name>goSteady3Emerg</name>
        </cagent>
      </cagency>
      <connections>
        <connection>
          <from>emergencybutton</from>
          <to>goSteady1Emerg.emergencybutton</to>

```

```

        </connection>
        <connection>
            <from>Z1</from>
            <to>goSteady1Emerg.Z</to>
        </connection>
        <connection>
            <from>emergencybutton</from>
            <to>goSteady2Emerg.emergencybutton</to>
        </connection>
        <connection>
            <from>Z2</from>
            <to>goSteady2Emerg.Z</to>
        </connection>
        <connection>
            <from>emergencybutton</from>
            <to>goSteady3Emerg.emergencybutton</to>
        </connection>
        <connection>
            <from>Z3</from>
            <to>goSteady3Emerg.Z</to>
        </connection>
        <connection>
            <from>goSteady1Emerg.voltage</from>
            <to>voltage1</to>
        </connection>
        <connection>
            <from>goSteady2Emerg.voltage</from>
            <to>voltage2</to>
        </connection>
        <connection>
            <from>goSteady3Emerg.voltage</from>
            <to>voltage3</to>
        </connection>
        <connection>
            <from>goSteady1Emerg.error</from>
            <to>error1</to>
        </connection>
        <connection>
            <from>goSteady2Emerg.error</from>
            <to>error2</to>
        </connection>
        <connection>
            <from>goSteady3Emerg.error</from>
            <to>error3</to>
        </connection>
    </connections>
</composite>
</implementation>
</cagentclass>

```

E.9 The GosteadyEmerg agent code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>GoSteadyEmerg</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>emergencybutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage</name>
            </output>
            <output>
                <type>real</type>
                <name>error</name>
            </output>
        </ports>
    </interface>
</cagentclass>

```

```

        </output>
    </ports>
</interface>
<implementation>
    <composite>
        <cagency>
            <cagent>
                <type>Accelerate</type>
                <name>accelerate</name>
                <parameters>
                    <parameter>
                        <name>endspeed</name>
                        <value>1.0</value>
                    </parameter>
                    <parameter>
                        <name>endposZ</name>
                        <value>-0.1</value>
                    </parameter>
                    <parameter>
                        <name>sampletime</name>
                        <value>0.001</value>
                    </parameter>
                    <parameter>
                        <name>K</name>
                        <value>5.0</value>
                    </parameter>
                    <parameter>
                        <name>BW</name>
                        <value>3.0</value>
                    </parameter>
                </parameters>
            </cagent>
            <cagent>
                <type>GoPosEmerg</type>
                <name>goPosEmerg</name>
                <parameters>
                    <parameter>
                        <name>K</name>
                        <value>80</value>
                    </parameter>
                    <parameter>
                        <name>Td</name>
                        <value>0.02</value>
                    </parameter>
                    <parameter>
                        <name>N</name>
                        <value>10.0</value>
                    </parameter>
                    <parameter>
                        <name>Ti</name>
                        <value>1000</value>
                    </parameter>
                    <parameter>
                        <name>minimum</name>
                        <value>-5</value>
                    </parameter>
                    <parameter>
                        <name>maximum</name>
                        <value>5</value>
                    </parameter>
                    <parameter>
                        <name>MV_scale</name>
                        <value>1.0</value>
                    </parameter>
                    <parameter>
                        <name>output_scale</name>
                        <value>1.0</value>
                    </parameter>
                    <parameter>
                        <name>sampletime</name>
                        <value>0.001</value>
                    </parameter>
                    <parameter>
                        <name>BW</name>
                        <value>3.0</value>
                    </parameter>
                </parameters>
            </cagent>
        </cagency>
    </composite>
</implementation>
</interface>

```

```

                <name>EndposZ</name>
                <value>-0.1</value>
            </parameter>
        </parameters>
    </cagent>
</cagency>
<coordination>
    <class>FixedPriorityCoordinator</class>
    <name>fixedPriorityM04</name>
</coordination>
<connections>
    <connection>
        <from>emergencybutton</from>
        <to>accelerate.activationinput</to>
    </connection>
    <connection>
        <from>Z</from>
        <to>accelerate.Z</to>
    </connection>
    <connection>
        <from>accelerate.voltage</from>
        <to>voltage</to>
    </connection>
    <connection>
        <from>accelerate.error</from>
        <to>error</to>
    </connection>
    <connection>
        <from>emergencybutton</from>
        <to>goPosEmerg.emergencybutton</to>
    </connection>
    <connection>
        <from>Z</from>
        <to>goPosEmerg.Z</to>
    </connection>
    <connection>
        <from>goPosEmerg.voltage</from>
        <to>voltage</to>
    </connection>
    <connection>
        <from>goPosEmerg.error</from>
        <to>error</to>
    </connection>
</connections>
</composite>
</implementation>
</cagentclass>

```

E.10 The Accelerate agent

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>Accelerate</name>
    <include><![CDATA[<cmath>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>activationinput</name>
            </input>
            <input>
                <type>real</type>
                <name>Z</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage</name>
            </output>
            <output>
                <type>real</type>
                <name>error</name>
            </output>
        </ports>
    </interface>
</cagentclass>

```

```

        </output>
    </ports>
    <parameterdefs>
        <parameterdef>
            <type>real</type>
            <name>endspeed</name>
            <defaultvalue>1.0</defaultvalue>
        </parameterdef>
        <parameterdef>
            <type>real</type>
            <name>endposZ</name>
            <defaultvalue>0.0</defaultvalue>
        </parameterdef>
        <parameterdef>
            <type>real</type>
            <name>sampletime</name>
            <defaultvalue>0.001</defaultvalue>
        </parameterdef>
        <parameterdef>
            <type>real</type>
            <name>K</name>
            <defaultvalue>5.0</defaultvalue>
        </parameterdef>
        <parameterdef>
            <type>real</type>
            <name>BW</name>
            <defaultvalue>5.0</defaultvalue>
        </parameterdef>
    </parameterdefs>
</interface>
<implementation>
    <elementary>
        <states>
            <state>
                <type>boolean</type>
                <name>isActive</name>
            </state>
            <state>
                <type>real</type>
                <name>previousZ</name>
            </state>
            <state>
                <type>real</type>
                <name>currspeed</name>
            </state>
            <state>
                <type>real</type>
                <name>speed</name>
            </state>
            <state>
                <type>real</type>
                <name>refspeed</name>
            </state>
            <state>
                <type>real</type>
                <name>speedstep</name>
            </state>
            <state>
                <type>real</type>
                <name>sly</name>
            </state>
            <state>
                <type>real</type>
                <name>s2y</name>
            </state>
            <state>
                <type>real</type>
                <name>BWrad</name>
            </state>
            <state>
                <type>real</type>
                <name>y</name>
            </state>
        </states>
        <start><![CDATA[
{
    BWrad = BW*2*3.1415926536;

```



```

        isAktive = false;
    }
        ]]></start>
        <initialize><![CDATA[
    {
        currspeed = (Z - previousZ) / sampletime;

        speedstep = currspeed - endspeed;

        if (endposZ<Z)
            speedstep = -speedstep;

        y = 0;

        voltage = 0.0;
    }
        ]]></initialize>
        <finalize><![CDATA[
    {
        speed = 0.0;
    }
        ]]></finalize>
        <activation><![CDATA[
    {
        if (isAktive)
            return 1.0;
        else
            return 0.0;
    }
        ]]></activation>
        <calculate><![CDATA[
    {
        s2y = BWrads*BWrads*(speedstep - y) - 1.4142*BWrads*sly;
        sly = sly + s2y*sampletime;
        y = y + sly*sampletime;

        refspeed = currspeed - y;

        speed = (Z - previousZ) / sampletime;

        error = refspeed - speed;
        voltage = error * K;

        if (fabs(Z-endposZ)< 0.02)
            isAktive = false;
    }
        ]]></calculate>
        <update><![CDATA[
    {
        if (activationinput == 1.0)
            isAktive = true;

        speed = (Z - previousZ) / sampletime;

        previousZ = Z;
    }
        ]]></update>
    </elementary>
</implementation>
</cagentclass>

```

E.11 The GoPosEmerg agent

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>GoPosEmerg</name>
    <include><![CDATA[<cmath>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>emergencybutton</name>

```

```

        </input>
        <input>
            <type>real</type>
            <name>Z</name>
        </input>
    </output>
        <type>real</type>
        <name>voltage</name>
    </output>
    <output>
        <type>real</type>
        <name>error</name>
    </output>
</ports>
<parameterdefs>
    <parameterdef>
        <type>real</type>
        <name>K</name>
        <defaultvalue>20</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>Td</name>
        <defaultvalue>0.02</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>N</name>
        <defaultvalue>10</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>Ti</name>
        <defaultvalue>1000</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>minimum</name>
        <defaultvalue>-5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>maximum</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>MV_scale</name>
        <defaultvalue>1.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>output_scale</name>
        <defaultvalue>1.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>sampletime</name>
        <defaultvalue>0.001</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>BW</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>EndposZ</name>
        <defaultvalue>0.0</defaultvalue>
    </parameterdef>
</parameterdefs>
</interface>
<implementation>
    <elementary>
        <states>
            <state>
                <type>real</type>

```

```

        <name>scaled_MV</name>
    </state>
    <state>
        <type>real</type>
        <name>factor</name>
    </state>
    <state>
        <type>real</type>
        <name>uD</name>
    </state>
    <state>
        <type>real</type>
        <name>uI</name>
    </state>
    <state>
        <type>real</type>
        <name>ideal_output</name>
    </state>
    <state>
        <type>real</type>
        <name>prevError</name>
    </state>
    <state>
        <type>boolean</type>
        <name>isAktive</name>
    </state>
    <state>
        <type>real</type>
        <name>refpos</name>
    </state>
    <state>
        <type>real</type>
        <name>currpos</name>
    </state>
    <state>
        <type>real</type>
        <name>posstep</name>
    </state>
    <state>
        <type>real</type>
        <name>s1y</name>
    </state>
    <state>
        <type>real</type>
        <name>s2y</name>
    </state>
    <state>
        <type>real</type>
        <name>BWrad</name>
    </state>
    <state>
        <type>real</type>
        <name>y</name>
    </state>
</states>
<start><![CDATA[
{
    BWrad = BW*2*3.1415926536;
    isAktive = false;
}

]]></start>
<initialize><![CDATA[
{
    currpos = Z;

    posstep = EndposZ - currpos;

    prevError=0.0;

    y = 0;

    voltage = 0.0;
}

]]></initialize>
<finalize><![CDATA[
{
}

```

```

        ]]></finalize>
        <activation><![CDATA[
{
    if (isAktive)
        return 1.0;
    else
        return 0.0;
}
        ]]></activation>
        <calculate><![CDATA[
{
    s2y = BWrاد*BWrad*(posstep - y) - 1.4142*BWrاد*sly;
    sly = sly + s2y*samplertime;
    y = y + sly*samplertime;

    refpos = currpos + y;

    scaled_MV = MV_scale * Z;
    error = refpos - scaled_MV;

    factor = 1 / ( samplertime + Td / N );

    uD = factor * (samplertime * K *error + Td * K * (error - prevError) + Td * uD / N );

    uI = uI + samplertime * uD / Ti ;

    ideal_output = uI + uD;

    voltage = output_scale * ideal_output;

    if (voltage<minimum)
        voltage=minimum;
    if (voltage>maximum)
        voltage=maximum;

    prevError=error;
}
        ]]></calculate>
        <update><![CDATA[
{
    if (emergencybutton == 1.0)
        isAktive = true;
}
        ]]></update>
    </elementary>
</implementation>
</cagentclass>

```

E.12 The GuardedStandard agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>GuardedStandard</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>startbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
        </ports>
    </interface>

```

```

    <input>
      <type>real</type>
      <name>Z3</name>
    </input>
    <output>
      <type>real</type>
      <name>voltage1</name>
    </output>
    <output>
      <type>real</type>
      <name>voltage2</name>
    </output>
    <output>
      <type>real</type>
      <name>voltage3</name>
    </output>
  </ports>
</interface>
<implementation>
  <composite>
    <agency>
      <cagent>
        <type>Standard</type>
        <name>standard1</name>
      </cagent>
      <cagent>
        <type>ErrorGuard</type>
        <name>eg</name>
      </cagent>
    </agency>
    <coordination>
      <class>MasterSlaveCoordinator</class>
      <name>c1</name>
    </coordination>
    <connections>
      <connection>
        <from>startbutton</from>
        <to>standard1.startbutton</to>
      </connection>
      <connection>
        <from>stopbutton</from>
        <to>standard1.stopbutton</to>
      </connection>
      <connection>
        <from>Z1</from>
        <to>standard1.Z1</to>
      </connection>
      <connection>
        <from>Z2</from>
        <to>standard1.Z2</to>
      </connection>
      <connection>
        <from>Z3</from>
        <to>standard1.Z3</to>
      </connection>
      <connection>
        <from>standard1.voltage1</from>
        <to>eg.voltageIn1</to>
      </connection>
      <connection>
        <from>standard1.voltage2</from>
        <to>eg.voltageIn2</to>
      </connection>
      <connection>
        <from>standard1.voltage3</from>
        <to>eg.voltageIn3</to>
      </connection>
      <connection>
        <from>standard1.error1</from>
        <to>eg.error1</to>
      </connection>
      <connection>
        <from>standard1.error2</from>
        <to>eg.error2</to>
      </connection>
      <connection>
        <from>standard1.error3</from>

```

```

        <to>eg.error3</to>
    </connection>
    <connection>
        <from>eg.voltageOut1</from>
        <to>voltage1</to>
    </connection>
    <connection>
        <from>eg.voltageOut2</from>
        <to>voltage2</to>
    </connection>
    <connection>
        <from>eg.voltageOut3</from>
        <to>voltage3</to>
    </connection>
</connections>
</composite>
</implementation>
</cagentclass>

```

E.13 The Standard agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>Standard</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>startbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage1</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage2</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage3</name>
            </output>
            <output>
                <type>real</type>
                <name>error1</name>
            </output>
            <output>
                <type>real</type>
                <name>error2</name>
            </output>
            <output>
                <type>real</type>
                <name>error3</name>
            </output>
        </ports>
    </interface>
</cagentclass>

```

```

</interface>
<implementation>
  <composite>
    <cagency>
      <cagent>
        <type>ModeSwitchController</type>
        <name>msc</name>
      </cagent>
      <cagent>
        <type>GravityCompensator</type>
        <name>gc</name>
      </cagent>
    </cagency>
    <coordination>
      <class>FuzzyAdditionCoordinator</class>
      <name>c1</name>
    </coordination>
    <connections>
      <connection>
        <from>startbutton</from>
        <to>msc.startbutton</to>
      </connection>
      <connection>
        <from>stopbutton</from>
        <to>msc.stopbutton</to>
      </connection>
      <connection>
        <from>Z1</from>
        <to>msc.Z1</to>
      </connection>
      <connection>
        <from>Z2</from>
        <to>msc.Z2</to>
      </connection>
      <connection>
        <from>Z3</from>
        <to>msc.Z3</to>
      </connection>
      <connection>
        <from>msc.voltage1</from>
        <to>voltage1</to>
      </connection>
      <connection>
        <from>msc.voltage2</from>
        <to>voltage2</to>
      </connection>
      <connection>
        <from>msc.voltage3</from>
        <to>voltage3</to>
      </connection>
      <connection>
        <from>msc.error1</from>
        <to>error1</to>
      </connection>
      <connection>
        <from>msc.error2</from>
        <to>error2</to>
      </connection>
      <connection>
        <from>msc.error3</from>
        <to>error3</to>
      </connection>
      <connection>
        <from>Z1</from>
        <to>gc.Z1</to>
      </connection>
      <connection>
        <from>Z2</from>
        <to>gc.Z2</to>
      </connection>
      <connection>
        <from>Z3</from>
        <to>gc.Z3</to>
      </connection>
      <connection>
        <from>gc.voltage1</from>
        <to>voltage1</to>
      </connection>
    </connections>
  </composite>
</implementation>

```

```

        </connection>
        <connection>
            <from>gc.voltage2</from>
            <to>voltage2</to>
        </connection>
        <connection>
            <from>gc.voltage3</from>
            <to>voltage3</to>
        </connection>
    </connections>
</composite>
</implementation>
</cagentclass>

```

E.14 The GravityCompensator agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>GravityCompensator</name>
    <include><![CDATA[<cmath>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage1</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage2</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage3</name>
            </output>
        </ports>
        <parameterdefs>
            <parameterdef>
                <type>real</type>
                <name>Vg</name>
                <defaultvalue>0.25</defaultvalue>
            </parameterdef>
        </parameterdefs>
    </interface>
    <implementation>
        <elementary>
            <activation><![CDATA[
{
    //here we could return a factor that depends on Z1, Z2 and Z3
    //for the time being, just take 1.0
    return 1.0;
}

]]></activation>
            <calculate><![CDATA[ {
voltage1 = Vg;
voltage2 = Vg;
voltage3 = Vg;
}

]]></calculate>
        </elementary>
    </implementation>
</cagentclass>

```



```

    </implementation>
</cagentclass>

```

E.15 The ModeSwitchController agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
  <name>ModeSwitchController</name>
  <interface>
    <ports>
      <input>
        <type>real</type>
        <name>stopbutton</name>
      </input>
      <input>
        <type>real</type>
        <name>startbutton</name>
      </input>
      <input>
        <type>real</type>
        <name>Z1</name>
      </input>
      <input>
        <type>real</type>
        <name>Z2</name>
      </input>
      <input>
        <type>real</type>
        <name>Z3</name>
      </input>
      <output>
        <type>real</type>
        <name>voltage1</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage2</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage3</name>
      </output>
      <output>
        <type>real</type>
        <name>error1</name>
      </output>
      <output>
        <type>real</type>
        <name>error2</name>
      </output>
      <output>
        <type>real</type>
        <name>error3</name>
      </output>
    </ports>
  </interface>
  <implementation>
    <composite>
      <cagency>
        <cagent>
          <type>Stop</type>
          <name>stop</name>
        </cagent>
        <cagent>
          <type>Operate</type>
          <name>operate</name>
        </cagent>
      </cagency>
      <!-- Insert Shutdown agent here -->
      <cagent>
        <type>HoldZeroPID</type>
        <name>holdZeroPID</name>
        <parameters>

```

```

        <parameter>
          <name>K</name>
          <value>80</value>
        </parameter>
        <parameter>
          <name>Td</name>
          <value>0.02</value>
        </parameter>
        <parameter>
          <name>N</name>
          <value>10.0</value>
        </parameter>
        <parameter>
          <name>Ti</name>
          <value>500</value>
        </parameter>
        <parameter>
          <name>minimum</name>
          <value>-5</value>
        </parameter>
        <parameter>
          <name>maximum</name>
          <value>5</value>
        </parameter>
        <parameter>
          <name>MV_scale</name>
          <value>1.0</value>
        </parameter>
        <parameter>
          <name>output_scale</name>
          <value>1.0</value>
        </parameter>
        <parameter>
          <name>sampletime</name>
          <value>0.001</value>
        </parameter>
      </parameters>
    </cagent>
  </cagency>
  <coordination>
    <class>FixedPriorityCoordinator</class>
    <name>fixedPriorityM02</name>
  </coordination>
  <connections>
    <connection>
      <from>stopbutton</from>
      <to>stop.stopbutton</to>
    </connection>
    <connection>
      <from>Z1</from>
      <to>stop.Z1</to>
    </connection>
    <connection>
      <from>Z2</from>
      <to>stop.Z2</to>
    </connection>
    <connection>
      <from>Z3</from>
      <to>stop.Z3</to>
    </connection>
    <connection>
      <from>startbutton</from>
      <to>operate.startbutton</to>
    </connection>
    <connection>
      <from>stopbutton</from>
      <to>operate.stopbutton</to>
    </connection>
    <connection>
      <from>Z1</from>
      <to>operate.Z1</to>
    </connection>
    <connection>
      <from>Z2</from>
      <to>operate.Z2</to>
    </connection>
  </connections>

```

```
<from>Z3</from>
<to>operate.Z3</to>
</connection>
<connection>
  <from>Z1</from>
  <to>holdZeroPID.Z1</to>
</connection>
<connection>
  <from>Z2</from>
  <to>holdZeroPID.Z2</to>
</connection>
<connection>
  <from>Z3</from>
  <to>holdZeroPID.Z3</to>
</connection>
<connection>
  <from>stop.voltage1</from>
  <to>voltage1</to>
</connection>
<connection>
  <from>stop.voltage2</from>
  <to>voltage2</to>
</connection>
<connection>
  <from>stop.voltage3</from>
  <to>voltage3</to>
</connection>
<connection>
  <from>operate.voltage1</from>
  <to>voltage1</to>
</connection>
<connection>
  <from>operate.voltage2</from>
  <to>voltage2</to>
</connection>
<connection>
  <from>operate.voltage3</from>
  <to>voltage3</to>
</connection>
<connection>
  <from>holdZeroPID.output1</from>
  <to>voltage1</to>
</connection>
<connection>
  <from>holdZeroPID.output2</from>
  <to>voltage2</to>
</connection>
<connection>
  <from>holdZeroPID.output3</from>
  <to>voltage3</to>
</connection>
<connection>
  <from>stop.error1</from>
  <to>error1</to>
</connection>
<connection>
  <from>stop.error2</from>
  <to>error2</to>
</connection>
<connection>
  <from>stop.error3</from>
  <to>error3</to>
</connection>
<connection>
  <from>operate.error1</from>
  <to>error1</to>
</connection>
<connection>
  <from>operate.error2</from>
  <to>error2</to>
</connection>
<connection>
  <from>operate.error3</from>
  <to>error3</to>
</connection>
<connection>
  <from>holdZeroPID.error1</from>
```

```

        <to>error1</to>
    </connection>
    <connection>
        <from>holdZeroPID.error2</from>
        <to>error2</to>
    </connection>
    <connection>
        <from>holdZeroPID.error3</from>
        <to>error3</to>
    </connection>
</connections>
</composite>
</implementation>
</cagentclass>

```

E.16 The Stop agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>Stop</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage1</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage2</name>
            </output>
            <output>
                <type>real</type>
                <name>voltage3</name>
            </output>
            <output>
                <type>real</type>
                <name>error1</name>
            </output>
            <output>
                <type>real</type>
                <name>error2</name>
            </output>
            <output>
                <type>real</type>
                <name>error3</name>
            </output>
        </ports>
    </interface>
    <implementation>
        <composite>
            <cagency>
                <cagent>
                    <type>Brake</type>
                    <name>brake</name>
                    <parameters>
                        <parameter>

```

```

                <name>sampletime</name>
                <value>0.001</value>
            </parameter>
            <parameter>
                <name>K</name>
                <value>5.0</value>
            </parameter>
            <parameter>
                <name>BW</name>
                <value>5.0</value>
            </parameter>
        </parameters>
    </cagent>
    <cagent>
        <type>GoSteadyAll</type>
        <name>goSteadyAll</name>
    </cagent>
</cagency>
<coordination>
    <class>FixedPriorityCoordinator</class>
    <name>fixedPriorityM03</name>
</coordination>
<connections>
    <connection>
        <from>stopbutton</from>
        <to>brake.activationinput</to>
    </connection>
    <connection>
        <from>Z1</from>
        <to>brake.Z1</to>
    </connection>
    <connection>
        <from>Z2</from>
        <to>brake.Z2</to>
    </connection>
    <connection>
        <from>Z3</from>
        <to>brake.Z3</to>
    </connection>
    <connection>
        <from>brake.voltage1</from>
        <to>voltage1</to>
    </connection>
    <connection>
        <from>brake.voltage2</from>
        <to>voltage2</to>
    </connection>
    <connection>
        <from>brake.voltage3</from>
        <to>voltage3</to>
    </connection>
    <connection>
        <from>brake.error1</from>
        <to>error1</to>
    </connection>
    <connection>
        <from>brake.error2</from>
        <to>error2</to>
    </connection>
    <connection>
        <from>brake.error3</from>
        <to>error3</to>
    </connection>
    <connection>
        <from>stopbutton</from>
        <to>goSteadyAll.stopbutton</to>
    </connection>
    <connection>
        <from>Z1</from>
        <to>goSteadyAll.Z1</to>
    </connection>
    <connection>
        <from>Z2</from>
        <to>goSteadyAll.Z2</to>
    </connection>
    <connection>
        <from>Z3</from>

```

```

        <to>goSteadyAll.Z3</to>
    </connection>
    <connection>
        <from>goSteadyAll.voltage1</from>
        <to>voltage1</to>
    </connection>
    <connection>
        <from>goSteadyAll.voltage2</from>
        <to>voltage2</to>
    </connection>
    <connection>
        <from>goSteadyAll.voltage3</from>
        <to>voltage3</to>
    </connection>
    <connection>
        <from>goSteadyAll.error1</from>
        <to>error1</to>
    </connection>
    <connection>
        <from>goSteadyAll.error2</from>
        <to>error2</to>
    </connection>
    <connection>
        <from>goSteadyAll.error3</from>
        <to>error3</to>
    </connection>
</connections>
</composite>
</implementation>
</cagentclass>

```

E.17 The GoSteadyAll agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
  <name>GoSteadyAll</name>
  <interface>
    <ports>
      <input>
        <type>real</type>
        <name>stopbutton</name>
      </input>
      <input>
        <type>real</type>
        <name>Z1</name>
      </input>
      <input>
        <type>real</type>
        <name>Z2</name>
      </input>
      <input>
        <type>real</type>
        <name>Z3</name>
      </input>
      <output>
        <type>real</type>
        <name>voltage1</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage2</name>
      </output>
      <output>
        <type>real</type>
        <name>voltage3</name>
      </output>
      <output>
        <type>real</type>
        <name>error1</name>
      </output>
      <output>
        <type>real</type>

```

```

        <name>error2</name>
    </output>
    <output>
        <type>real</type>
        <name>error3</name>
    </output>
</ports>
</interface>
<implementation>
    <composite>
        <cagency>
            <cagent>
                <type>GoSteady</type>
                <name>goSteady1</name>
            </cagent>
            <cagent>
                <type>GoSteady</type>
                <name>goSteady2</name>
            </cagent>
            <cagent>
                <type>GoSteady</type>
                <name>goSteady3</name>
            </cagent>
        </cagency>
    </composite>
    <connections>
        <connection>
            <from>stopbutton</from>
            <to>goSteady1.stopbutton</to>
        </connection>
        <connection>
            <from>Z1</from>
            <to>goSteady1.Z</to>
        </connection>
        <connection>
            <from>Z2</from>
            <to>goSteady1.Z2</to>
        </connection>
        <connection>
            <from>Z3</from>
            <to>goSteady1.Z3</to>
        </connection>
        <connection>
            <from>stopbutton</from>
            <to>goSteady2.stopbutton</to>
        </connection>
        <connection>
            <from>Z2</from>
            <to>goSteady2.Z</to>
        </connection>
        <connection>
            <from>Z1</from>
            <to>goSteady2.Z2</to>
        </connection>
        <connection>
            <from>Z3</from>
            <to>goSteady2.Z3</to>
        </connection>
        <connection>
            <from>stopbutton</from>
            <to>goSteady3.stopbutton</to>
        </connection>
        <connection>
            <from>Z3</from>
            <to>goSteady3.Z</to>
        </connection>
        <connection>
            <from>Z1</from>
            <to>goSteady3.Z3</to>
        </connection>
        <connection>
            <from>Z2</from>
            <to>goSteady3.Z2</to>
        </connection>
        <connection>
            <from>goSteady1.voltage</from>
            <to>voltage1</to>
        </connection>
    </connections>
</implementation>
</composite>
</implementation>
</interface>

```

```

        <connection>
            <from>goSteady2.voltage</from>
            <to>voltage2</to>
        </connection>
        <connection>
            <from>goSteady3.voltage</from>
            <to>voltage3</to>
        </connection>
        <connection>
            <from>goSteady1.error</from>
            <to>error1</to>
        </connection>
        <connection>
            <from>goSteady2.error</from>
            <to>error2</to>
        </connection>
        <connection>
            <from>goSteady3.error</from>
            <to>error3</to>
        </connection>
    </connections>
</composite>
</implementation>
</cagentclass>

```

E.18 The GoSteady agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>GoSteady</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <output>
                <type>real</type>
                <name>voltage</name>
            </output>
            <output>
                <type>real</type>
                <name>error</name>
            </output>
        </ports>
    </interface>
    <implementation>
        <composite>
            <cagency>
                <cagent>
                    <type>Accelerate</type>
                    <name>accelerate</name>
                    <parameters>
                        <parameter>
                            <name>endspeed</name>
                            <value>0.7</value>
                        </parameter>
                        <parameter>
                            <name>endposZ</name>
                            <value>0.0</value>
                        </parameter>
                    </parameters>
                </cagent>
            </cagency>
        </composite>
    </implementation>
</cagentclass>

```



```

        </parameter>
        <parameter>
            <name>sampletime</name>
            <value>0.001</value>
        </parameter>
        <parameter>
            <name>K</name>
            <value>5.0</value>
        </parameter>
        <parameter>
            <name>BW</name>
            <value>3.0</value>
        </parameter>
    </parameters>
</cagent>
<cagent>
    <type>GoPos</type>
    <name>goPos</name>
    <parameters>
        <parameter>
            <name>K</name>
            <value>80</value>
        </parameter>
        <parameter>
            <name>Td</name>
            <value>0.02</value>
        </parameter>
        <parameter>
            <name>N</name>
            <value>10.0</value>
        </parameter>
        <parameter>
            <name>Ti</name>
            <value>1000</value>
        </parameter>
        <parameter>
            <name>minimum</name>
            <value>-5</value>
        </parameter>
        <parameter>
            <name>maximum</name>
            <value>5</value>
        </parameter>
        <parameter>
            <name>MV_scale</name>
            <value>1.0</value>
        </parameter>
        <parameter>
            <name>output_scale</name>
            <value>1.0</value>
        </parameter>
        <parameter>
            <name>sampletime</name>
            <value>0.001</value>
        </parameter>
        <parameter>
            <name>BW</name>
            <value>3.0</value>
        </parameter>
        <parameter>
            <name>EndposZ</name>
            <value>0.0</value>
        </parameter>
        <parameter>
            <name>EndposZ2</name>
            <value>0.0</value>
        </parameter>
        <parameter>
            <name>EndposZ3</name>
            <value>0.0</value>
        </parameter>
    </parameters>
</cagent>
</cagency>
<coordination>
    <class>FixedPriorityCoordinator</class>
    <name>fixedPriorityM04</name>

```

```

    </coordination>
    <connections>
      <connection>
        <from>stopbutton</from>
        <to>accelerate.activationinput</to>
      </connection>
      <connection>
        <from>Z</from>
        <to>accelerate.Z</to>
      </connection>
      <connection>
        <from>accelerate.voltage</from>
        <to>voltage</to>
      </connection>
      <connection>
        <from>accelerate.error</from>
        <to>error</to>
      </connection>
      <connection>
        <from>stopbutton</from>
        <to>goPos.stopbutton</to>
      </connection>
      <connection>
        <from>Z</from>
        <to>goPos.Z</to>
      </connection>
      <connection>
        <from>Z2</from>
        <to>goPos.Z2</to>
      </connection>
      <connection>
        <from>Z3</from>
        <to>goPos.Z3</to>
      </connection>
      <connection>
        <from>goPos.voltage</from>
        <to>voltage</to>
      </connection>
      <connection>
        <from>goPos.error</from>
        <to>error</to>
      </connection>
    </connections>
  </composite>
</implementation>
</cagentclass>

```

E.19 The GoPos agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
  <name>GoPos</name>
  <include><![CDATA[<cmath>]]></include>
  <interface>
    <ports>
      <input>
        <type>real</type>
        <name>stopbutton</name>
      </input>
      <input>
        <type>real</type>
        <name>Z</name>
      </input>
      <input>
        <type>real</type>
        <name>Z2</name>
      </input>
      <input>
        <type>real</type>
        <name>Z3</name>
      </input>
    </ports>
  </interface>
</cagentclass>

```

```

        <type>real</type>
        <name>voltage</name>
    </output>
</output>
    <type>real</type>
    <name>error</name>
</output>
</ports>
<parameterdefs>
    <parameterdef>
        <type>real</type>
        <name>K</name>
        <defaultvalue>20</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>Td</name>
        <defaultvalue>0.02</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>N</name>
        <defaultvalue>10</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>Ti</name>
        <defaultvalue>1000</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>minimum</name>
        <defaultvalue>-5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>maximum</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>MV_scale</name>
        <defaultvalue>1.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>output_scale</name>
        <defaultvalue>1.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>sampletime</name>
        <defaultvalue>0.001</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>BW</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>EndposZ</name>
        <defaultvalue>0.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>EndposZ2</name>
        <defaultvalue>0.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>EndposZ3</name>
        <defaultvalue>0.0</defaultvalue>
    </parameterdef>
</parameterdefs>
</interface>
<implementation>

```

```

    <elementary>
      <states>
        <state>
          <type>real</type>
          <name>scaled_MV</name>
        </state>
        <state>
          <type>real</type>
          <name>factor</name>
        </state>
        <state>
          <type>real</type>
          <name>uD</name>
        </state>
        <state>
          <type>real</type>
          <name>uI</name>
        </state>
        <state>
          <type>real</type>
          <name>ideal_output</name>
        </state>
        <state>
          <type>real</type>
          <name>prevError</name>
        </state>
        <state>
          <type>boolean</type>
          <name>isActive</name>
        </state>
        <state>
          <type>real</type>
          <name>refpos</name>
        </state>
        <state>
          <type>real</type>
          <name>currpos</name>
        </state>
        <state>
          <type>real</type>
          <name>posstep</name>
        </state>
        <state>
          <type>real</type>
          <name>s1y</name>
        </state>
        <state>
          <type>real</type>
          <name>s2y</name>
        </state>
        <state>
          <type>real</type>
          <name>BWrad</name>
        </state>
        <state>
          <type>real</type>
          <name>y</name>
        </state>
      </states>
      <start><![CDATA[
{
  BWrad = BW*2*3.1415926536;
  isActive = false;
}

]]></start>
<initialize><![CDATA[
{
  currpos = Z;

  posstep = EndposZ - currpos;

  prevError=0.0;

  y = 0;

  voltage = 0.0;
}

```

```

        ]]></initialize>
        <finalize><![CDATA[
{
}
        ]]></finalize>
        <activation><![CDATA[
{
    if (isAktive)
        return 1.0;
    else
        return 0.0;
}
        ]]></activation>
        <calculate><![CDATA[
{
    s2y = BWrada*BWrada*(posstep - y) - 1.4142*BWrada*sly;
    sly = sly + s2y*samplertime;
    y = y + sly*samplertime;

    refpos = currpos + y;

    scaled_MV = MV_scale * Z;
    error = refpos - scaled_MV;

    factor = 1 / ( samplertime + Td / N );

    uD = factor * (samplertime * K *error + Td * K * (error - prevError) + Td * uD / N );
    uI = uI + samplertime * uD / Ti ;

    ideal_output = uI + uD;

    voltage = output_scale * ideal_output;

    if (voltage<minimum)
        voltage=minimum;
    if (voltage>maximum)
        voltage=maximum;

    prevError=error;
    if ((fabs(Z - EndposZ)< 0.001) && (fabs(Z2 - EndposZ2)<0.001) && (fabs(Z3 -
EndposZ3)<0.001))
        isAktive = false;
}
        ]]></calculate>
        <update><![CDATA[
{
    if (stopbutton == 1.0)
        isAktive = true;
}
        ]]></update>
    </elementary>
</implementation>
</cagentclass>

```

E.20 The Operate agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>Operate</name>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>startbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
        </ports>
    </interface>
</cagentclass>

```

```

        </input>
        <input>
            <type>real</type>
            <name>Z2</name>
        </input>
        <input>
            <type>real</type>
            <name>Z3</name>
        </input>
        <output>
            <type>real</type>
            <name>voltage1</name>
        </output>
        <output>
            <type>real</type>
            <name>voltage2</name>
        </output>
        <output>
            <type>real</type>
            <name>voltage3</name>
        </output>
        <output>
            <type>real</type>
            <name>error1</name>
        </output>
        <output>
            <type>real</type>
            <name>error2</name>
        </output>
        <output>
            <type>real</type>
            <name>error3</name>
        </output>
    </ports>
</interface>
<implementation>
    <composite>
        <cagency>
            <cagent>
                <type>PID</type>
                <name>pidController</name>
                <parameters>
                    <parameter>
                        <name>K</name>
                        <value>50000</value>
                    </parameter>
                    <parameter>
                        <name>Td</name>
                        <value>0.008</value>
                    </parameter>
                    <parameter>
                        <name>N</name>
                        <value>10.0</value>
                    </parameter>
                    <parameter>
                        <name>Ti</name>
                        <value>0.25</value>
                    </parameter>
                    <parameter>
                        <name>minimum</name>
                        <value>-5</value>
                    </parameter>
                    <parameter>
                        <name>maximum</name>
                        <value>5</value>
                    </parameter>
                    <parameter>
                        <name>MV_scale</name>
                        <value>1.0</value>
                    </parameter>
                    <parameter>
                        <name>output_scale</name>
                        <value>0.012820</value>
                    </parameter>
                    <parameter>
                        <name>sampletime</name>
                        <value>0.001</value>
                    </parameter>
                </parameters>
            </cagent>
        </cagency>
    </composite>
</implementation>
</interface>

```

```
        </parameter>
    </parameters>
</cagent>
<cagent>
    <type>PathFromFile</type>
    <name>pathFromFile</name>
</cagent>
</cagency>
<connections>
    <connection>
        <from>startbutton</from>
        <to>pathFromFile.startbutton</to>
    </connection>
    <connection>
        <from>stopbutton</from>
        <to>pathFromFile.stopbutton</to>
    </connection>
    <connection>
        <from>startbutton</from>
        <to>pidController.startbutton</to>
    </connection>
    <connection>
        <from>stopbutton</from>
        <to>pidController.stopbutton</to>
    </connection>
    <connection>
        <from>Z1</from>
        <to>pidController.MV1</to>
    </connection>
    <connection>
        <from>Z2</from>
        <to>pidController.MV2</to>
    </connection>
    <connection>
        <from>Z3</from>
        <to>pidController.MV3</to>
    </connection>
    <connection>
        <from>Z1</from>
        <to>pathFromFile.Z1</to>
    </connection>
    <connection>
        <from>Z2</from>
        <to>pathFromFile.Z2</to>
    </connection>
    <connection>
        <from>Z3</from>
        <to>pathFromFile.Z3</to>
    </connection>
    <connection>
        <from>pathFromFile.Z1ref</from>
        <to>pidController.SP1</to>
    </connection>
    <connection>
        <from>pathFromFile.Z2ref</from>
        <to>pidController.SP2</to>
    </connection>
    <connection>
        <from>pathFromFile.Z3ref</from>
        <to>pidController.SP3</to>
    </connection>
    <connection>
        <from>pidController.output1</from>
        <to>voltage1</to>
    </connection>
    <connection>
        <from>pidController.output2</from>
        <to>voltage2</to>
    </connection>
    <connection>
        <from>pidController.output3</from>
        <to>voltage3</to>
    </connection>
    <connection>
        <from>pidController.error1</from>
        <to>error1</to>
    </connection>
</connections>
```

```

        <connection>
            <from>pidController.error2</from>
            <to>error2</to>
        </connection>
        <connection>
            <from>pidController.error3</from>
            <to>error3</to>
        </connection>
    </connections>
</composite>
</implementation>
</cagentclass>

```

E.21 The PathFromFile agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>PathFromFile</name>
    <include><![CDATA["PATHHEAD.H"]]></include>
    <include><![CDATA[<math>]]></include>
    <include><![CDATA[<stdio.h>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>startbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>stopbutton</name>
            </input>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <output>
                <type>real</type>
                <name>Z1ref</name>
            </output>
            <output>
                <type>real</type>
                <name>Z2ref</name>
            </output>
            <output>
                <type>real</type>
                <name>Z3ref</name>
            </output>
        </ports>
    </interface>
    <implementation>
        <elementary>
            <states>
                <state>
                    <type>boolean</type>
                    <name>canAktive</name>
                </state>
                <state>
                    <type>boolean</type>
                    <name>AtStartPos</name>
                </state>
                <state>
                    <type>real</type>
                    <name>Z1step</name>
                </state>
            </states>
        </elementary>
    </implementation>
</cagentclass>

```



```

        </state>
        <state>
            <type>real</type>
            <name>Z2step</name>
        </state>
        <state>
            <type>real</type>
            <name>Z3step</name>
        </state>
        <state>
            <type>real</type>
            <name>BWrad</name>
        </state>
        <state>
            <type>real</type>
            <name>s1y1</name>
        </state>
        <state>
            <type>real</type>
            <name>s2y1</name>
        </state>
        <state>
            <type>real</type>
            <name>y1</name>
        </state>
        <state>
            <type>real</type>
            <name>s1y2</name>
        </state>
        <state>
            <type>real</type>
            <name>s2y2</name>
        </state>
        <state>
            <type>real</type>
            <name>y2</name>
        </state>
        <state>
            <type>real</type>
            <name>s1y3</name>
        </state>
        <state>
            <type>real</type>
            <name>s2y3</name>
        </state>
        <state>
            <type>real</type>
            <name>y3</name>
        </state>
        <state>
            <type>real</type>
            <name>sampletime</name>
        </state>
        <state>
            <type>real</type>
            <name>BW</name>
        </state>
        <state>
            <type>real</type>
            <name>Z1curr</name>
        </state>
        <state>
            <type>real</type>
            <name>Z2curr</name>
        </state>
        <state>
            <type>real</type>
            <name>Z3curr</name>
        </state>
    </states>
    <start><![CDATA[
{
    sampletime = 0.001;
    BW = 1;

    BWrad = BW*2*3.1415926536;

```

```

canAktive = false;
AtStartPos = false;

FILE *fp;
char Dummy[200];
float f1, f2, f3;
int FilPos;

if((fp = fopen("path.egl", "r"))==NULL)
{
    printf("Cannot open file...");
    exit(1);
}

// get the number of samples and make the arrays

fgets(Dummy, 200, fp);
fgets(Dummy, 200, fp);
fgets(Dummy, 200, fp);
FilPos = ftell(fp);
fseek(fp, FilPos + 14, SEEK_SET);
fscanf(fp, "%u", &NrSamples);
fgets(Dummy, 200, fp);

RefArrayZ1 = new double[NrSamples];
RefArrayZ2 = new double[NrSamples];
RefArrayZ3 = new double[NrSamples];

for(index=0;index<NrSamples;index++)
{
    fscanf(fp, "%f %f %f", &f1, &f2, &f3);
    RefArrayZ1[index] = f1;
    RefArrayZ2[index] = f2;
    RefArrayZ3[index] = f3;
}

fclose(fp);
index = 0;
}

]]</start>
<initialize><![CDATA[
{
Z1step = RefArrayZ1[0] - Z1;
Z2step = RefArrayZ2[0] - Z2;
Z3step = RefArrayZ3[0] - Z3;

Z1curr = Z1;
Z2curr = Z2;
Z3curr = Z3;

y1 = 0.0;
y2 = 0.0;
y3 = 0.0;

if((fabs(Z1step) <0.002)&&(fabs(Z2step) <0.002)&&(fabs(Z3step) <0.002))
    AtStartPos = true;
else
    AtStartPos = false;

Z1ref = Z1;
Z2ref = Z2;
Z3ref = Z3;
index = 0;
}

]]</initialize>

<finalize><![CDATA[
{
canAktive = false;
}

]]</finalize>

```

```

        <activation><![CDATA[
{
    return canAktive;
}
]]></activation>
<calculate><![CDATA[
{
    if (AtStartPos)
    {
        if (index<NrSamples)
        {
            Z1ref = RefArrayZ1[index];
            Z2ref = RefArrayZ2[index];
            Z3ref = RefArrayZ3[index];
            index++;
        }
        else
        {
            Z1ref = RefArrayZ1[index-1];
            Z2ref = RefArrayZ2[index-1];
            Z3ref = RefArrayZ3[index-1];
        }
    }
    else
    {
        s2y1 = BWrads*BWrads*(Z1step - y1) - 1.4142*BWrads*sly1;
        sly1 = sly1 + s2y1*sampletime;
        y1 = y1 + sly1*sampletime;

        s2y2 = BWrads*BWrads*(Z2step - y2) - 1.4142*BWrads*sly2;
        sly2 = sly2 + s2y2*sampletime;
        y2 = y2 + sly2*sampletime;

        s2y3 = BWrads*BWrads*(Z3step - y3) - 1.4142*BWrads*sly3;
        sly3 = sly3 + s2y3*sampletime;
        y3 = y3 + sly3*sampletime;

        Z1ref = Z1curr + y1;
        Z2ref = Z2curr + y2;
        Z3ref = Z3curr + y3;

        if ((fabs(Z1 - RefArrayZ1[0]) <0.002)&&(fabs(Z2 - RefArrayZ2[0]) <0.002)&&(fabs(Z3
- RefArrayZ3[0]) <0.002))
            AtStartPos = true;
    }
}
]]></calculate>
<update><![CDATA[
{
    if (startbutton == 1.0)
        canAktive = true;

    if (stopbutton == 1.0)
        canAktive = false;
}
]]></update>
</elementary>
</implementation>
</cagentclass>

```

E.22 The PID agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>PID</name>
    <include><![CDATA[<cmath></include>
    <interface>
        <ports>
            <input>
                <type>real</type>

```

```

        <name>startbutton</name>
</input>
<input>
    <type>real</type>
    <name>stopbutton</name>
</input>
<input>
    <type>real</type>
    <name>SP1</name>
</input>
<input>
    <type>real</type>
    <name>SP2</name>
</input>
<input>
    <type>real</type>
    <name>SP3</name>
</input>
<input>
    <type>real</type>
    <name>MV1</name>
</input>
<input>
    <type>real</type>
    <name>MV2</name>
</input>
<input>
    <type>real</type>
    <name>MV3</name>
</input>
<output>
    <type>real</type>
    <name>output1</name>
</output>
<output>
    <type>real</type>
    <name>output2</name>
</output>
<output>
    <type>real</type>
    <name>output3</name>
</output>
<output>
    <type>real</type>
    <name>error1</name>
</output>
<output>
    <type>real</type>
    <name>error2</name>
</output>
<output>
    <type>real</type>
    <name>error3</name>
</output>
</ports>
<parameterdefs>
<parameterdef>
    <type>real</type>
    <name>K</name>
    <defaultvalue>10</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>Td</name>
    <defaultvalue>0.02</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>N</name>
    <defaultvalue>10</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>Ti</name>
    <defaultvalue>1000</defaultvalue>
</parameterdef>
</parameterdef>

```

```

        <type>real</type>
        <name>minimum</name>
        <defaultvalue>-5.0</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>maximum</name>
    <defaultvalue>5.0</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>MV_scale</name>
    <defaultvalue>1.0</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>output_scale</name>
    <defaultvalue>1.0</defaultvalue>
</parameterdef>
<parameterdef>
    <type>real</type>
    <name>sampletime</name>
    <defaultvalue>0.001</defaultvalue>
</parameterdef>
</parameterdefs>
</interface>
<implementation>
    <elementary>
        <states>
            <state>
                <type>boolean</type>
                <name>canAktive</name>
            </state>
            <state>
                <type>real</type>
                <name>scaled_MV1</name>
            </state>
            <state>
                <type>real</type>
                <name>scaled_MV2</name>
            </state>
            <state>
                <type>real</type>
                <name>scaled_MV3</name>
            </state>
            <state>
                <type>real</type>
                <name>factor</name>
            </state>
            <state>
                <type>real</type>
                <name>uD1</name>
            </state>
            <state>
                <type>real</type>
                <name>uD2</name>
            </state>
            <state>
                <type>real</type>
                <name>uD3</name>
            </state>
            <state>
                <type>real</type>
                <name>uI1</name>
            </state>
            <state>
                <type>real</type>
                <name>uI2</name>
            </state>
            <state>
                <type>real</type>
                <name>uI3</name>
            </state>
            <state>
                <type>real</type>
                <name>ideal_output1</name>
            </state>
        </states>
    </elementary>
</implementation>

```

```

        </state>
        <state>
            <type>real</type>
            <name>ideal_output2</name>
        </state>
        <state>
            <type>real</type>
            <name>ideal_output3</name>
        </state>
        <state>
            <type>real</type>
            <name>prevError1</name>
        </state>
        <state>
            <type>real</type>
            <name>prevError2</name>
        </state>
        <state>
            <type>real</type>
            <name>prevError3</name>
        </state>
    </states>
    <start><![CDATA[
{
    canAktive = false;
}
    ]]></start>
        <initialize><![CDATA[
{
    prevError1=0.0;
    prevError2=0.0;
    prevError3=0.0;
}
    ]]></initialize>
        <finalize><![CDATA[
{
}
    ]]></finalize>
        <activation><![CDATA[
{
    return canAktive;
}
    ]]></activation>
        <calculate><![CDATA[ {
scaled_MV1 = MV_scale * MV1;
scaled_MV2 = MV_scale * MV2;
scaled_MV3 = MV_scale * MV3;
error1 = SP1 - scaled_MV1;
error2 = SP2 - scaled_MV2;
error3 = SP3 - scaled_MV3;

factor = 1 / ( sampletime + Td / N );

uD1 = factor * (sampletime * K *error1 + Td * K * (error1 - prevError1) + Td * uD1 / N
);
uD2 = factor * (sampletime * K *error2 + Td * K * (error2 - prevError2) + Td * uD2 / N
);
uD3 = factor * (sampletime * K *error3 + Td * K * (error3 - prevError3) + Td * uD3 / N
);

uI1 = uI1 + sampletime * uD1 / Ti ;
uI2 = uI2 + sampletime * uD2 / Ti ;
uI3 = uI3 + sampletime * uD3 / Ti ;

ideal_output1 = uI1 + uD1;
ideal_output2 = uI2 + uD2;
ideal_output3 = uI3 + uD3;

output1 = output_scale * ideal_output1;
output2 = output_scale * ideal_output2;
output3 = output_scale * ideal_output3;

if (output1<minimum)
    output1=minimum;
if (output1>maximum)
    output1=maximum;

```

```

        if (output2<minimum)
            output2=minimum;
        if (output2>maximum)
            output2=maximum;

        if (output3<minimum)
            output3=minimum;
        if (output3>maximum)
            output3=maximum;

        prevError1=error1;
        prevError2=error2;
        prevError3=error3;
    }
        ]]></calculate>
        <update><![CDATA[
    {
        if (startbutton == 1.0)
            canAktive = true;
        if (stopbutton == 1.0)
            canAktive = false;

        scaled_MV1 = MV_scale * MV1;
        scaled_MV2 = MV_scale * MV2;
        scaled_MV3 = MV_scale * MV3;
        error1 = SP1 - scaled_MV1;
        error2 = SP2 - scaled_MV2;
        error3 = SP3 - scaled_MV3;
    }
        ]]></update>
    </elementary>
</implementation>
</cagentclass>

```

E.23 The HoldZero agent XML code

```

<?xml version="1.0"?>
<cagentclass xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CagentSchema.xsd">
    <name>HoldZeroPID</name>
    <include><![CDATA[<cmath>]]></include>
    <interface>
        <ports>
            <input>
                <type>real</type>
                <name>Z1</name>
            </input>
            <input>
                <type>real</type>
                <name>Z2</name>
            </input>
            <input>
                <type>real</type>
                <name>Z3</name>
            </input>
            <output>
                <type>real</type>
                <name>output1</name>
            </output>
            <output>
                <type>real</type>
                <name>output2</name>
            </output>
            <output>
                <type>real</type>
                <name>output3</name>
            </output>
            <output>
                <type>real</type>
                <name>error1</name>
            </output>
            <output>
                <type>real</type>

```

```

        <name>error2</name>
    </output>
    <output>
        <type>real</type>
        <name>error3</name>
    </output>
</ports>
<parameterdefs>
    <parameterdef>
        <type>real</type>
        <name>K</name>
        <defaultvalue>20</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>Td</name>
        <defaultvalue>0.02</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>N</name>
        <defaultvalue>10</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>Ti</name>
        <defaultvalue>1000</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>minimum</name>
        <defaultvalue>-5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>maximum</name>
        <defaultvalue>5.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>MV_scale</name>
        <defaultvalue>1.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>output_scale</name>
        <defaultvalue>1.0</defaultvalue>
    </parameterdef>
    <parameterdef>
        <type>real</type>
        <name>sampletime</name>
        <defaultvalue>0.001</defaultvalue>
    </parameterdef>
</parameterdefs>
</interface>
<implementation>
    <elementary>
        <states>
            <state>
                <type>real</type>
                <name>scaled_MV1</name>
            </state>
            <state>
                <type>real</type>
                <name>scaled_MV2</name>
            </state>
            <state>
                <type>real</type>
                <name>scaled_MV3</name>
            </state>
            <state>
                <type>real</type>
                <name>factor</name>
            </state>
            <state>
                <type>real</type>
                <name>uD1</name>
            </state>
        </states>
    </elementary>
</implementation>

```



```

        </state>

        <state>
            <type>real</type>
            <name>uD2</name>
        </state>
        <state>
            <type>real</type>
            <name>uD3</name>
        </state>
        <state>
            <type>real</type>
            <name>uI1</name>
        </state>
        <state>
            <type>real</type>
            <name>uI2</name>
        </state>
        <state>
            <type>real</type>
            <name>uI3</name>
        </state>
        <state>
            <type>real</type>
            <name>ideal_output1</name>
        </state>
        <state>
            <type>real</type>
            <name>ideal_output2</name>
        </state>
        <state>
            <type>real</type>
            <name>ideal_output3</name>
        </state>
        <state>
            <type>real</type>
            <name>prevError1</name>
        </state>
        <state>
            <type>real</type>
            <name>prevError2</name>
        </state>
        <state>
            <type>real</type>
            <name>prevError3</name>
        </state>
    </states>
    <start><![CDATA[
{
}

    ]]></start>
    <initialize><![CDATA[
{
    prevError1=0.0;
    prevError2=0.0;
    prevError3=0.0;
}

    ]]></initialize>
    <finalize><![CDATA[
{
}

    ]]></finalize>
    <activation><![CDATA[
{

    return 1.0;
}

    ]]></activation>
    <calculate><![CDATA[ {
    scaled_MV1 = MV_scale * Z1;
    scaled_MV2 = MV_scale * Z2;
    scaled_MV3 = MV_scale * Z3;
    error1 = - scaled_MV1;
    error2 = - scaled_MV2;
    error3 = - scaled_MV3;

    factor = 1 / ( sampletime + Td / N );

```

```

    uD1 = factor * (samplettime * K *error1 + Td * K * (error1 - prevError1) + Td * uD1 / N
);
    uD2 = factor * (samplettime * K *error2 + Td * K * (error2 - prevError2) + Td * uD2 / N
);
    uD3 = factor * (samplettime * K *error3 + Td * K * (error3 - prevError3) + Td * uD3 / N
);

    uI1 = uI1 + samplettime * uD1 / Ti ;
    uI2 = uI2 + samplettime * uD2 / Ti ;
    uI3 = uI3 + samplettime * uD3 / Ti ;

    ideal_output1 = uI1 + uD1;
    ideal_output2 = uI2 + uD2;
    ideal_output3 = uI3 + uD3;

    output1 = output_scale * ideal_output1;
    output2 = output_scale * ideal_output2;
    output3 = output_scale * ideal_output3;

    if (output1<minimum)
        output1=minimum;
    if (output1>maximum)
        output1=maximum;

    if (output2<minimum)
        output2=minimum;
    if (output2>maximum)
        output2=maximum;

    if (output3<minimum)
        output3=minimum;
    if (output3>maximum)
        output3=maximum;

    prevError1=error1;
    prevError2=error2;
    prevError3=error3;
}
        ]]></calculate>
    </elementary>
</implementation>
</cagentclass>

```

Bibliography

- Angeles, J. (2003). *Fundamentals of Robotic Mechanical Systems Theory, Methods, and Algorithms*, Second Edition, Springer-Verlag, ISBN: 0-387-95368-X.
- Bajracharya, G. (2003). *Integrated Design and Implementation Tool for Multi-Agent Controllers [IDITmac]*. MSc thesis, University of Twente, Enschede, The Netherlands.
- Coelingh, H. (2000). *Design support for motion control systems*, PhD thesis, University of Twente, Enschede, The Netherlands.
- De Kruif, B.J., De Vries, T.J.A. (2003). *On-line Nonparametric Regression to learn State-Dependent Disturbances*. Control Engineering, University of Twente, Enschede, The Netherlands.
- De Kruif, B.J. *Approximation of the plateau position, 20-Sim model*. Personal communication.
- Deitel, H. M. (2001). *XML: how to program*, Prentice Hall, ISBN: 0-13-028417-3
- Franklin, S., Graesser, A. (1997). "Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents", Intelligent Agents III: Agent Theories, Architectures, and Languages, Proceedings of ECAI'96 Workshop(ATAL), Hungary, Aug. 1996 (From Lecture Notes in Artificial Intelligence 1193, pp. 21-35, 1997).
- Küpers, B.E. (1998). *Implementation of a control algorithm and safety measures for a manipulator*. MSc thesis, University of Twente, Enschede, The Netherlands.
- Schildt, H. (1998). *C++: The Complete Reference*, Third Edition, McGraw-Hill, ISBN: 0-07-882476-1
- Starrenburg, J.G., De Vries, T.J.A. (1995). *Learning vehicle control for the Mobile Autonomous Robot*. RB Electronica, ISSN: 0928-5008, pp. 12-15.
- Stramigioli, S. (1998). *From Differentiable Manifolds to Interactive Robot Control*, PhD thesis, Delft University of Technology, Delft, The Netherlands.
- Van Breemen, A.J.N. (2000). *An Agent-Based Multi Controller Systems, A design framework for complex control problems*. PhD thesis, University of Twente, Enschede, The Netherlands.
- Van Breemen, A.J.N., De Vries, T.J.A. (2001). Design and implementation of a room thermostat using an agent-based approach. *Control Engineering Practice* **9**, 233 – 248.
- Van De Mast, F. (1992). *Safety measures for the OSCAR-6 robot*. MSc thesis, University of Twente, Enschede, The Netherlands.
- Velthuis, W.J.R. (2000). *Learning feed-forward control - theory, design and applications*. PhD thesis, University of Twente, Enschede, The Netherlands.
- De Vries, T.J.A. *20-Sim model*. Personal communication.