MASTER

Preliminary investigation of a planar motor with a moving magnet configuration and with six degrees of freedom

Jansen, J.W.

Award date:
2003

TU/e Technische Universiteit Eindhoven
/Sectie Elektromechanica en Vermogenselektronica

Master of Science thesis

Preliminary Investigation of a Planar Motor with a Moving Magnet Configuration and with Six Degrees of Freedom

EPE 2003-04 J.W. Jansen

Supervisors: Prof.dr.ir. A.J.A. Vandenput
Dr. E.A. Lomonova

Eindhoven, June 2003

De Faculteit der Elektrotechniek van de Technische Universiteit Eindhoven aanvaardt geen verantwoordelijkheid voor de inhoud van stage- en afstudeerverslagen.

Abstract

New magnetically suspended planar motors, which can be controlled in six degrees of freedom and can operate in deep vacuum, are being developed as an alternative for multi-dimensional drives, which are constructed by stacking one-dimensional linear drives. Solutions, based on a stationary magnet array and moving coils, have been developed, but the cable slab to the propelled platform and the cooling of the moving coils are major disadvantages. For this reason, the goal of the IOP-EMVT planar motor project is the synthesis and design of a planar motor with the reversed structure: standstill coils and moving magnets. The subject of this thesis is the preliminary investigation of such a planar motor.

As an alternative and supplementary to finite element simulations fast and time saving numerical tools have been implemented in Matlab for the three dimensional static analysis of the planar motor. These tools can calculate currents, forces, torques and inductances in planar motor structures. The calculations are based on analytical solutions of the magnetic vector potential and the magnetic flux density. Two three-dimensional finite element packages, Maxwell 3D and FLUX3D, have been used to verify these tools. After a review of these packages, the current version of FLUX3D proved to be the most suitable package for further use in the project.

Two geometric variables of a planar motor have been optimized using the tools, implemented in Matlab. This optimization is carried out for two different magnet structures, a Halbach array and a regular N-S array. As the objective function, the minimization of the rms-force ripple is chosen. The optimization shows that a position independent force can be produced by the motor with a ripple of less than 2%. From these solutions the ohmic losses of the planar motor are estimated as function of the mass of the moving platform. The motor with the regular N-S array proved to have less losses due to the better force to pure magnet mass ratio.

To perform experiments on the planar motor in the future an experimental setup has to be built, which is based on a H-drive with three linear motors.

Preface

This Master's thesis concludes my study at the Electrical Engineering Department of the Eindhoven University of Technology. It describes the preliminary investigation of a moving magnet planar motor. During this project, I realized more and more which challenging problems have to be solved, before this motor can be built.

I would like to thank prof.dr.ir. André Vandenput for the opportunity to make my final project about planar drives in the Electromechanics and Power Electronics group, his supervision and pressure to finish the project in time.

I am grateful to dr. Elena Lomonova for her support, ideas and her precious comments on my thesis.

I am grateful to prof.dr.ir John Compter for arranging several visits to Philips CFT and introducing me to several people.

I wish to thank Ad Vermeer and the company Assembléon because they partly sponsor an H-drive for the planar motor project.

I also want to thank all other people, who helped me during this project.

Contents

List of symbols

symbol	dimension	name
α	kg/m	mass per meter
θ	rad	rotation angle about the y-axis
μ_0	Hm^{-1}	permeability of vacuum (per definition: $4\pi \cdot 10\cdot^{-7}$)
μ_r	-	relative permeability
σ	T	magnetic surface charge
τ	-	unity vector along a local tangent of a contour
τ_p	m	pole pitch
ϕ	rad	rotation angle about the z-axis
ϕ_m	Wb	linked flux of the permanent magnets
ψ	rad	rotation angle about the x-axis
$\mathbf{A}$	Wbm^{-1}	magnetic vector potential
$\mathbf{B}$, B	T	magnetix flux density
B_r	T	remanence
C_e	J	magnetic co-energy
$\mathbf{F}$,F	N	force
I, i	A	current
$\mathbf{J}$	Am^{-2}	current density
L	H	inductance
M	H	mutual inductance
N	-	number of turns
S	m^2	surface
$\mathbf{T}$	Nm	torque
V	m^3	volume
W_m	J	magnetic energy
a	ms^{-2}	acceleration
ag	m	air gap height
cf	m	fillet radius
ch	m	height of coils
cl	m	length of coils
cw	m	width of coils
ct	m	conductor thickness
f_{cost}	-	cost function
h	m	height
l	m	length
m	kg	mass
ms	m	length and width of square magnets

symbol	dimension	name
msh	m	height of square magnets
mr	m	length of short side of rectangular magnets
mrh	m	height of rectangular magnets
n	-	unity vector perpendicular to a surface
r	m	distance to a point
w	-	weight factor

subscript	refers to
a	magnet array
c	coil array
l	load
p	platform
m	mean value
r	rms-ripple
x	x-direction
y	y-direction
z	z-direction

Chapter 1

Introduction

1.1 Background, topic and goal

In many industrial machineries, like pick-and-place machines, wafer scanners and inspection systems, objects are positioned and moved in a horizontal plane. Most multi-dimensional positioning stages in these systems have long strokes in the x- and y-directions and smaller strokes in one or more other directions. These stages are often constructed by stacking several one-dimensional linear drives. The disadvantage of this concept is that the specification of the individual drives is determined rather by the weight of the propelled drives, than the weight of the object, propelled by the positioning stage.

An alternative for these drives are planar motors. Planar motors have only one propelled platform. The earliest planar drives have air or roller bearings and are limited to three degrees of freedom (DOF). Because of the bearings, they are not fitted for vacuum applications. Therefore, planar drives with six degrees of freedom are being developed. The concept of the earliest solutions is based on a standstill magnet array and a moving and magnetically levitated coil array. However, the dynamic disturbances of the cable slab to the propelled platform and the high weight of the platform, due to the necessary cooling system, are major disadvantages.

The object of this research is to overcome these disadvantages and to develop and design a planar motor with the reversed structure: a long stroke planar motor with standstill coils and moving magnets, which is suitable for vacuum applications. This research is sponsored by the Dutch government as part of the innovation driven research programs (IOP-EMVT). The research will be carried out by two Ph.D. students, who will work in close collaboration. One of the students will focus on the electromagnetic design, the other on the control algorithms.

The object of this thesis is the preliminary investigation of planar motors with moving magnets, using both analytical and numerical tools and the design of a setup for experiments with the planar motor.

1.2 Outline of the planar motor

Figure 1.1 shows the outline of the planar motor structure, which will be designed; a stationary part with the coil elements and a platform with permanent magnets, which floats above the coils and can be positioned in six degrees of freedom. The

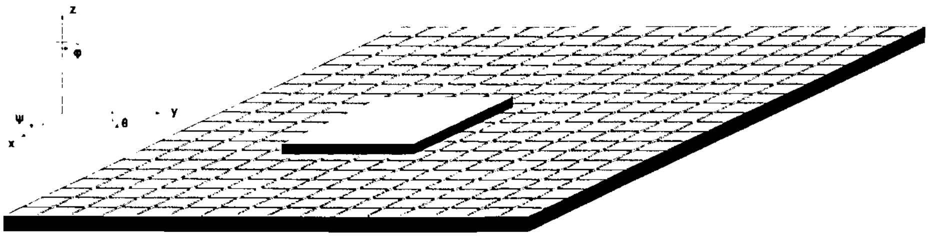


Figure 1.1. *Planar motor structure with standstill coils and moving magnets (white platform).*

coil array is only at one side of the moving platform. Therefore, the stroke of the machine is only limited by the size of the coil array.

1.3 Definitions

The next definitions are used in this report for the description and characterization of planar motors:

stroke is the maximum displacement of the moving platform in x and y-direction.

propulsion is related to the movements of the moving platform in the x- and y-directions.

suspension is related to the movements of the moving platform in the z-direction.

1.4 Structure of the thesis

This chapter and the second chapter of this thesis provide an introduction to planar motor technology. The literature review in chapter 2 covers all types of planar motors but concentrates on the motors with six degrees of freedom. The chapter ends with a planar motor structure, which is the starting point of the calculations and optimizations in this thesis.

The third chapter presents the tools, which are based on analytical equations, for the electromechanical analysis of planar motors with cuboidal magnets, coils and iron. These tools are faster and therefore more suitable for parametric optimization than the 3D-finite element software packages for electromechanical analysis. Two of these packages, FLUX3D and Maxwell 3D, are reviewed in chapter 4. To validate the tools based on the analytical equations, they are compared with finite element simulations in chapter 5.

An optimization of a planar motor geometry, by means of the analytical model is presented in chapter 6. The power loss of the optimized configurations is also estimated in this chapter. A test setup for measurements on a planar motor is proposed in chapter 7. This thesis ends with conclusions and recommendations for the planar motor project in chapter 8.

Chapter 2

Planar motor technology

Planar motors have been designed and applied in different applications for many years. Simple, relative inaccurate planar stepper motors, which can move in two or three degrees of freedom, have been applied in xy-printers and xy-positioning tables. The most advanced planar motors are designed for the integrated circuit manufacturing industry to position wafers with nanometer accuracy during lithographical processes. This chapter provides an overview of the different planar motor technologies. Different groups of planar motors can be distinguished on their working principle and structure:

- planar stepper motors (2.1),
- planar induction motors (2.2),
- planar permanent magnet motors constructed of linked actuators (2.3),
- planar permanent magnet motors constructed of integrated actuators (2.4).

Many planar permanent magnet (PM) motors, which provide movement in three or less degrees of freedom, can be found in literature. In the sections 2.3 and 2.4 only the planar PM motors, which provide movements in six degrees of freedom, are described. In the final section of this chapter, a planar motor structure is proposed which is the starting point for the calculations and optimizations in this thesis.

2.1 Planar stepper motors

The oldest type of planar motors is the planar stepper motor. One of the earliest planar stepper motors has been patented by B.A. Sawyer in 1968 [1]. Therefore, these types of planar motor are also named *Sawyer motor.* Planar stepper motors can only move in three degrees of freedom; in the x-direction, the y-direction and they can rotate a small angle about the z-axis. Figure 2.1a shows the structure of the Sawyer motor. The large stationary part is a plate of a soft-magnetic material with equally spaced slots in the x- and y-directions. Figure 2.1b shows a view of the moving part. The moving part consists of four sets of teeth and coils. Two sets (27 and 29) are used to produce force in the x-direction, the other two sets (28 and 20) are used to produce force in the y-direction. Two sets in every direction are used in order that the torque about the z-axis, produced by every set, is cancelled.

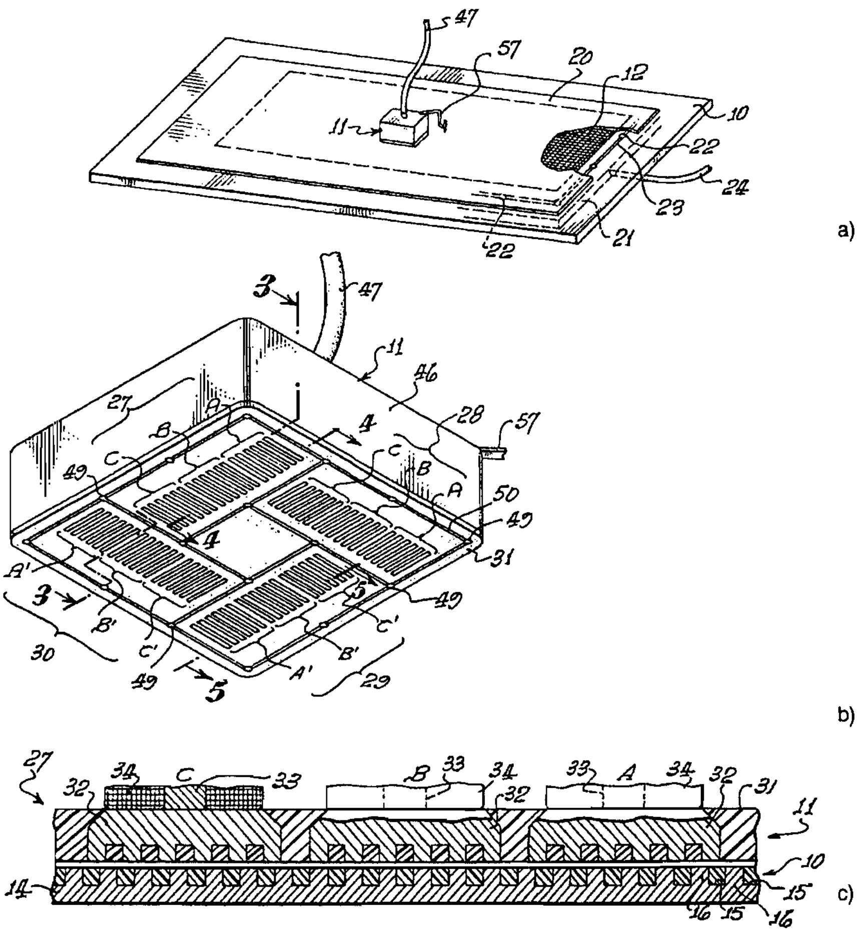


Figure 2.1. *Sawyer motor, a) overview, b) detail of the moving part, c) sectional view (after [1]).*

Figure 2.1c shows a cross section of the planar motor with the coils and teeth of set 27 and the teeth in the stationary part. In this case, coil B has been energized as the teeth below coil B are aligned with the teeth of the stator. The set moves to the right if the coils C, A and B are energized successively.

The position accuracy of planar stepper motors is limited but the advantage is that a feedback controller is not required.

2.2 Planar induction motors

A planar induction motor can move in three degrees of freedom; rotation about the z-axis, and translations along the x- and y-axes. Figure 2.2a and b show a planar induction motor, developed by Fujii et al. [2]. The motor has a moving toroidal core with eight sections. There is a set of three-phase windings in every section. The stationary part simply consists of a conducting plate and a back-iron plate.

For rotation about the z-axis, all coils are used to generated a rotating mag-

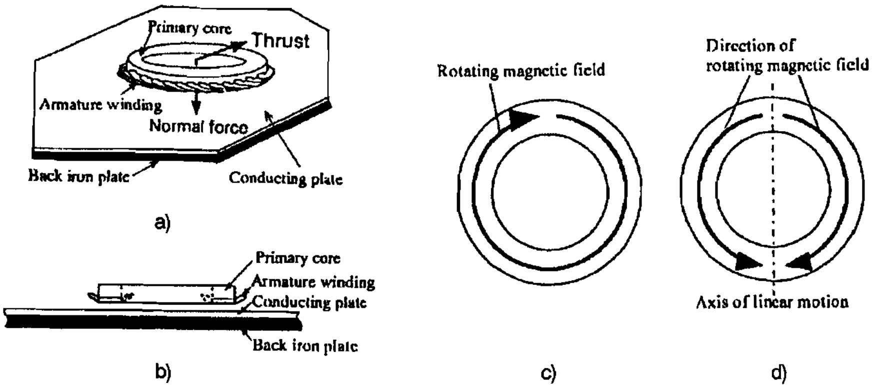


Figure 2.2. *Planar induction motor, a) overview, b) side view, c) direction of the magnetic field during rotation and d) translation of the core (after [2]).*

netic field, as shown in Figure 2.2c. For linear movement, the windings are separated into two groups, which produce two opposite rotating magnetic fields as shown in Figure 2.2d. The direction of linear movement can be changed arbitrarily by changing the current amplitude and phase in the eight coil groups.

The disadvantage of induction motors is that the dynamic behavior of the motor is difficult to model and, therefore, the motor is not suitable for precision applications.

2.3 Planar PM motors constructed of linked actuators with six degrees of freedom

Another group of planar motors is constructed of several linked linear permanent magnet actuators. Contrary to the gantry-like positioning systems with stacked drives, this group of drives has one moving part, which is the common moving member of several drives. The disadvantage of these motors is that the stroke of the planar motor is relatively small and is limited by the stroke of the separate actuators. Different types of planar motors which are magnetically suspended and have six degrees of freedom can be found in literature. These motors are described below.

From Earnshaw's theorem formulated in 1842, it can be shown that in a static magnetic field, the equilibrium of the forces acting on a body is not stable if the magnetic permeability of the materials in the field, $\mu_r > 1$ [3]. As in all these drives the translator is magnetically levitated, a feedback controller is necessary to maintain stable levitation. Only levitation of diamagnetic materials ($\mu_r < 1$) can be stable without feedback control.

2.3.1 Auer (1995)

The planar motor of F. Auer is constructed of three suspension and propulsion units [4]. A single unit is shown in Figure 2.3a. The stator consists of two E-shape cores. The translator is suspended by the reluctance forces produced by the coils

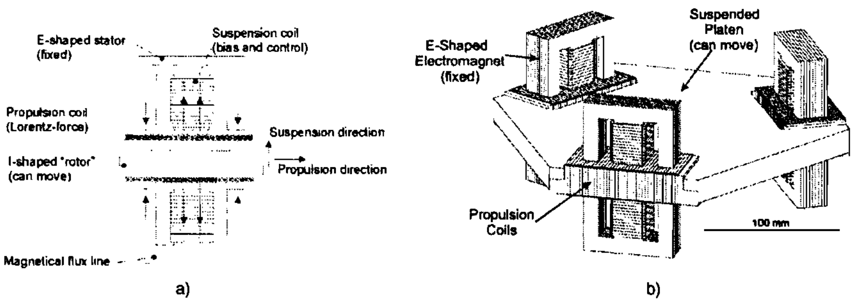

E-shaped stator (fixed)
Suspension coil (bias and control)
Propulsion coil (Lorentz-force)
I-shaped 'rotor' (can move)
Suspension direction
Propulsion direction
Magnetical flux line
a)
E-Shaped Electromagnet (fixed)
Suspended Platen (can move)
Propulsion Coils
100 mm
b)

Figure 2.3. *Planar motor designed by Auer, a) single suspension and propulsion unit, b) planar motor constructed of three units (after [4]).*

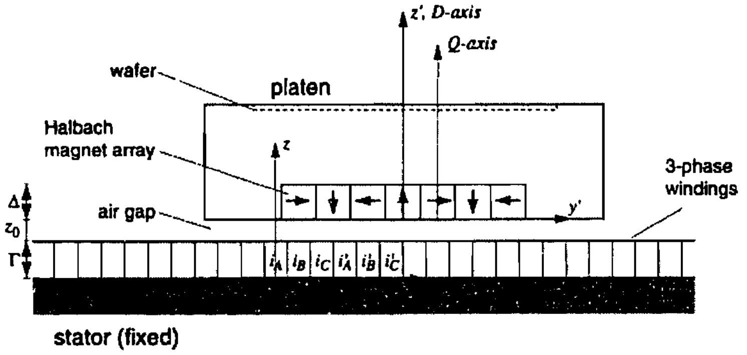


Figure 2.4. *Cross-section of a linear motor bearing (after [5]).*

around the middle teeth of the cores. The translator can move horizontally by the Lorentz forces on the windings around it. Figure 2.3b shows a planar motor, which is constructed by linking three of these units. The stroke of the motor is limited to 10x10 mm. The motor has an accuracy of 0.5 μm. The disadvantage of this drive is that the flux in the air gap is low and, therefore, the propulsion force is only 0.7 N. The low force density results also in a high power dissipation of 104 W for suspension only. The suspended mass is 3.0 kg.

2.3.2 Kim (1997)

The planar positioning stage of W.J. Kim is a precision positioning device with an accuracy of 50 nm and with a stroke of 50x50 mm [5]. The motor is designed for the precise positioning of wafers in a wafer stepper. The stage consists of four magnetically levitated linear motors with moving magnets. A section of one of these linear motors is shown in Figure 2.4. Each linear motor consists of stationary 3-phase windings and a Halbach magnet array, attached to the moving platen. The Halbach magnet array is used to concentrate the flux density at one side of the array and to increase the efficiency of the motor [6].

A DQ-decomposition can be applied to the motor. The stator current on the

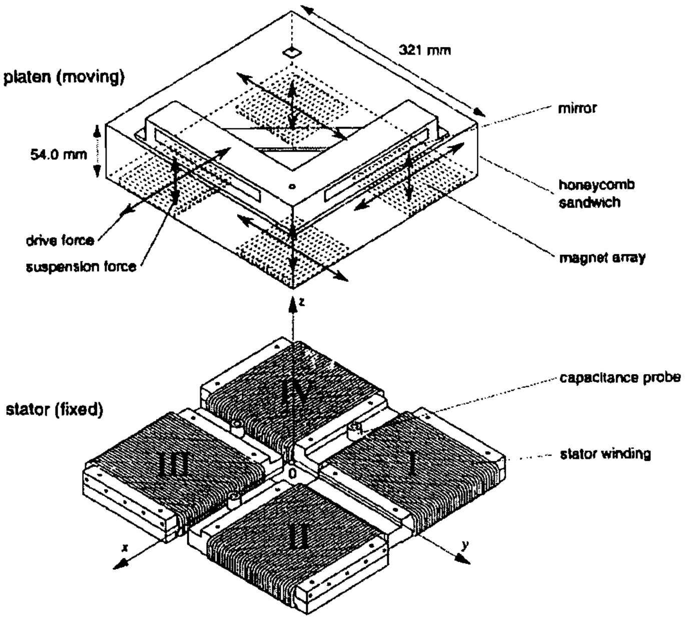


Figure 2.5. *Planar motor constructed of four magnetically levitated linear motors (after [5]).*

direct axis (D-axis) causes a Lorentz force in the z-direction and is responsible for the levitation of the platen. The stator current on the quadrature axis (Q-axis), causes a Lorentz force in the y-direction and is responsible for propulsion of the platen. Figure 2.5 shows the drive with the four linear motors. Two are placed in the x-direction and the other two in the y-direction. The disadvantage of this drive is that the stroke is limited because every magnet array has to stay above its own coil set.

2.3.3 Molenaar (2000)

The planar motor by L. Molenaar is a new type of Lorentz drive, named *moving iron* [7]. Both the permanent magnets and the propulsion coils are located on the stationary part of the motor. Figure 2.6 shows the planar motor. The flux from the permanent magnets prestresses the suspension of the T-shaped translator. The suspension is controlled by the three suspension coils. Lorentz forces, which propels the translator, are produced by the six propulsion coils and the flux of the permanent magnets. The heat dissipation of this motor is extremely low (0.8 mW) because of the prestressed magnetic bearings. The motor is therefore very suitable for vacuum application. For comparison, Kim's motor without prestressed magnetic bearings has a continuous dissipation of 21.6 W. The stroke of Molenaar's motor is only 130 mm and cannot be extended without redesign of the motor. The position accuracy of the motor is 75 nm.

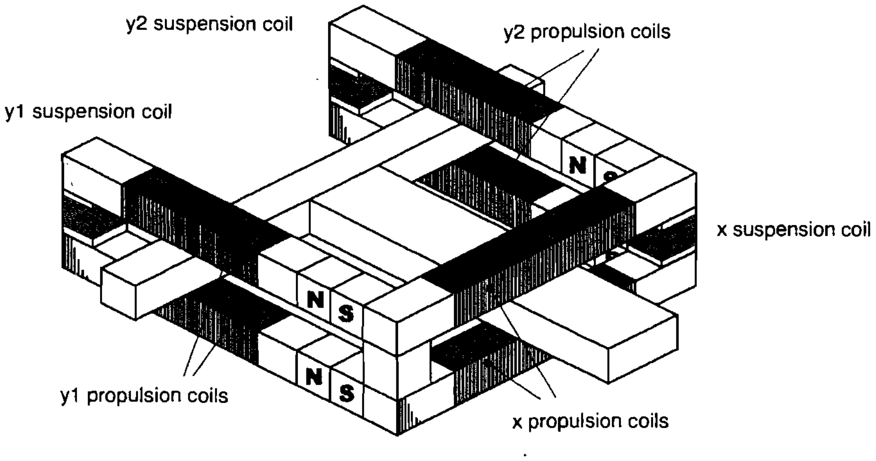


Figure 2.6. *Overview of Molenaar's planar motor (after [7]).*

2.4 Planar motors constructed of integrated actuators with six degrees of freedom

The limitation of the motors, presented in the previous section, is that the stroke of the motor is limited by the individual actuators in the construction. Extension of the stroke in x- and y-directions is not possible without a redesign of the complete actuator. The planar motors with six degrees of freedom presented in this section, do not have this limitation. The stroke of the actuators can be enlarged by extension of the stationary part of the motor. Therefore, they are more suitable as long stroke actuators.

2.4.1 Compter & Frissen (2001)

The 6 DOF planar motor, which is designed by J.C. Compter & P.C.M. Frissen at Philips CFT, is a moving coil motor [8], [9]. Similar to Kim's motor, the motor consists of a combination of four magnetically levitated linear motors or forcers of which two are placed in one direction and the other two are in a direction perpendicular to the first one. The difference with Kim's planar motor is that these forcers share one magnet array.

Figure 2.7a shows a top view of the motor. The four forcers are placed at an angle of 45 degrees with respect to the Halbach magnet array. Every forcer consists of two sets of three phase windings. Both sets are displaced over the half pole pitch. This way, a torque about the z-axis produced by every set during displacement of the moving part is counteracted. Figure 2.7b shows a detail of the magnet array with the magnetization directions of the magnets.

The demonstrator of this motor has a precision of 10 μm and can accelerate with 1 m/s^2. The disadvantage of a moving coil motor is that the cable slab to the propelled platform, influences the dynamic behaviour of the motor.

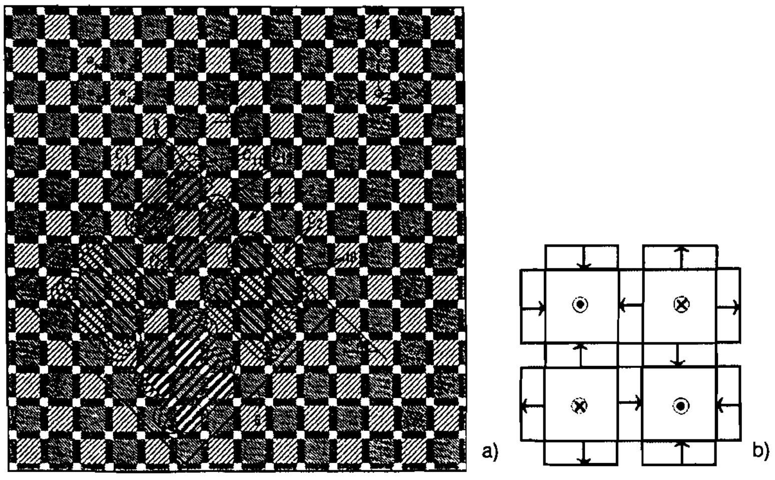


Figure 2.7. *Moving coil planar motor, a) top view , b) detail of the magnet array with the magnetization directions of the magnets (after [8]).*

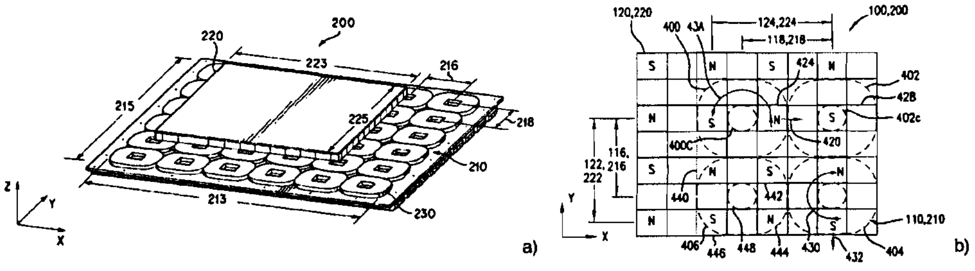


Figure 2.8. *Moving magnet planar motor of Hazelton et al., a) overview, b) detail of the magnet array and the coils (after [10]).*

2.4.2 Hazelton et al. (2001)

The planar motor invented by A.J. Hazelton et al. is a moving magnet planar motor [10], which provides movement in six degrees of freedom. The invention is owned by Nikon Coorporation. Figure 2.8a shows a perspective view of the planar motor. On the stationary part there are coils, of which 16 coils are covered by the moving magnet array. These 16 coils are divided into 4 groups. Figure 2.8b shows a top view of one of those four coil groups and its orientation with respect to the magnet array. The four coils, 400, 402, 404, 406 produce in this position Lorentz forces in, respectively, the y-direction, z-direction, x-direction and no force if a current is applied to the coils. The force can be kept constant during displacement by means of commutation of the current, "in fact," in two-dimensional two-phase windings.

In addition to this geometry, embodiments with double-sided coil arrays (on top and bottom side of the array) and with extra prestressed magnetic bearings are presented in the patent. In those embodiments the moving platen is enclosed by the stationary part and the stroke is limited by the extra magnetic bearings. Therefore, these embodiments should be classified into the planar motors with linked actuators.

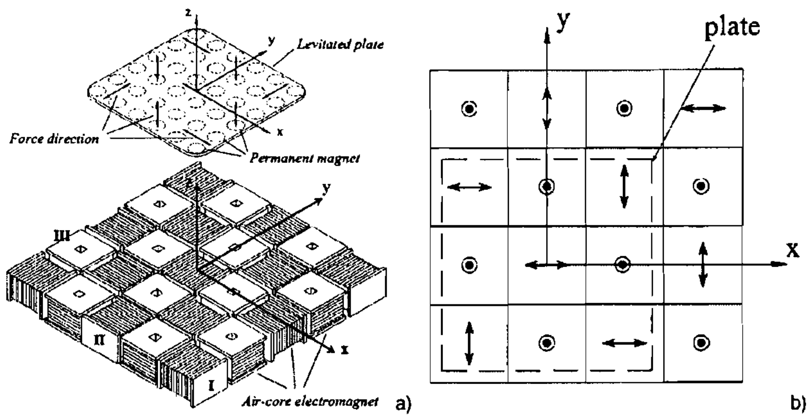


Figure 2.9. *Moving magnet motor of Jung & Baek , a) overview, b) direction of the force produced by the coils (after [11]).*

Because of this reason, the capabilities of the embodiment shown in Figure 2.8, to move with 6 DOFs, are questionable.

2.4.3 Jung & Baek (2002)

Jung & Beak developed a moving magnet planar motor with a small stroke of 32x32 mm [11]. The levitated magnet array consists of 36 magnets, which have all the same magnetization direction. The coil array consists of three types of air-core coils which are arranged parallel with the x-, y- and z-axes. Figure 2.9a shows a perspective view on the motor. Figure 2.9b shows the forces on the magnets, produced by the coils for one position of the plate.

The disadvantage of the motor is that the levitated weight is only 173 gram, and that the stroke is very limited. In Figure 2.9b the plate is shown in one posiition. Two coils are used for force production in the y-direction and three are for the force in the x-direction. If the plate moves one coil set to the right, only two coils for both the x- and y-directions are available. Moreover, the torque produced by the coils should be controlled. Therefore, the controller of this motor is complicated.

2.4.4 Binnard (2002)

M. Binnard is the inventor of another planar motor with moving magnets [12]. This invention is also owned by Nikon Cooperation. The planar motor is shown in Figure 2.10a. The motor has four layers of windings on top of each other. The top two layers are shifted 90 electrical degrees in phase and are oriented in the y-direction. These layers can produce forces in the x- and z-directions. The two bottom layers have the same shape and dimensions as the two top layers, but they are oriented in the x-direction to produce forces in the y- and z-directions.

The disadvantage of this drive is that the current density in the bottom layers should be much higher than in the top layers to produce the same force, as the magnetic flux density decreases fast with the distance to the magnet array. Figure 2.10b shows a depiction of the magnet array.

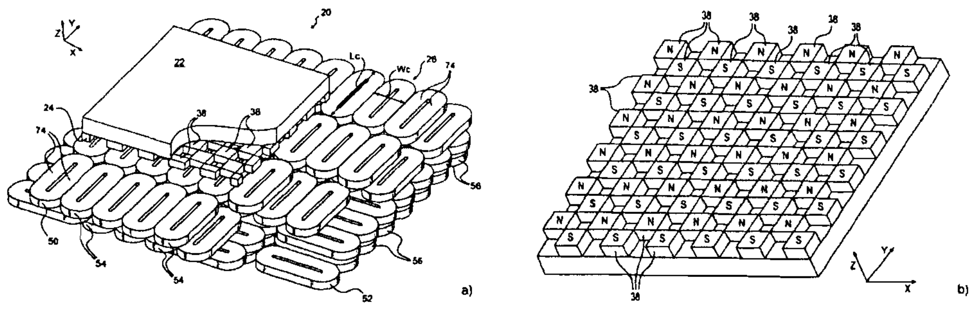


Figure 2.10. *Moving magnet motor of Binnard, a) overview, b) magnet array of the motor (after [12]).*

Table 2.1. *Comparison of the characteristics of 6-DOF permanent magnet planar motors.*

characteristic	Auer (1995)	Kim (1997)	Molenaar (2000)	Jung&Baek (2002)
principle	moving coils	moving magnets	moving iron	moving magnets
x, y-stroke	10x10 mm	50x50 mm	130x130 mm	32x32 mm
suspended mass	3.0 kg	5.47 kg	3.37 kg	173 g
position accuracy	0.5 μm	50 nm	75 nm	0.5 μm
propulsion force	0.7 N	53.6 N	6.5 N	unknown
acceleration	0.2 m/s^2	9.84 m/s^2	1.9 m/s^2	unknown
continuous dissipation	104 W	21.6 W	0.8 mW	unknown

2.5 Conclusions

In the previous sections, four different types of planar motors are described. The first two types, the planar stepper motor and the planar induction motor, cannot provide six degrees of freedom. The permanent magnet planar motors can serve movement in six degrees of freedom. The specifications of these motors are summarized in Table 2.1. As the patents of Nikon do not offer any specific information about the specifications or status of their moving magnet planar motors, they are not included in the table. Also the motor of Compter&Frissen is not included because only some specifications of the demonstrator are known.

From the literature review can be concluded that the IOP-EMVT moving magnet planar motor project is, indeed, strategic research. Only limited literature resources can be found on planar motors with moving magnets. However, these designs have some clear disadvantages or the status of these designs is unknown. One of the risks of the IOP-EMVT planar motor project is an infringement of the Nikon patents.

From Table 2.1 specifications for the new planar motor can be derived. The table shows that the small stroke planar motors of Kim and Molenaar have nanometer precision. But because of the costs of interferometers, micrometer precision is the more feasible target for the new planar motor. The fastest accelerating motor is Kim's motor. Therefore, this acceleration is a good target specification. But, as long as the size and mass of the moving platform are unknown, it is impossible to calculate the propulsion force and suspension force. The minimum specifications of the planar motor are:

- stroke only limited by the size of the coil array,
- position accuracy at μm level,
- acceleration of 1 g,
- low dissipation, design for vacuum application.

2.6 Starting point for the planar motor analysis

As a starting point for the preliminary analysis of planar motors with moving magnets in this thesis, the planar motor structure, which is shown in Figure 2.11, is selected. The meaning of the symbols is summarized in Table 2.2. This simplified planar motor consists of the combination of the Halbach magnet array, presented in section 2.4.1 and the two-phase windings in x- and y-directions, presented in section 2.4.2. In this embodiment only the force calculations are interesting because no stable levitation can be made with only four coils. At least three of these coil groups should support the magnet array. However, this structure is useful to gain insight in the working principle of the motor.

The pole pitch, τ_p is defined as the distance between the middle of two adjacent in z-direction magnetized magnets. Assumed that at a certain distance z under the array with pole pitch, τ_p, the magnetic flux density, $\mathbf{B}$, is sinusoidally distributed and equal to:

$$\mathbf{B} = B \begin{bmatrix} \sin\left(\frac{\pi x}{\tau_p}\right) \\ \sin\left(\frac{\pi y}{\tau_p}\right) \\ \cos\left(\frac{\pi x}{\tau_p}\right)\cos\left(\frac{\pi y}{\tau_p}\right) \end{bmatrix}, \tag{2.1}$$

and the coil currents, $I_{1,1}$, $I_{1,2}$, $I_{2,1}$, $I_{2,2}$ are equal to:

$$I_{1,1} = -I_x \cos\left(\frac{\pi x}{\tau_p}\right) \sin\left(\frac{\pi y}{\tau_p}\right) - I_y \sin\left(\frac{\pi x}{\tau_p}\right) \cos\left(\frac{\pi y}{\tau_p}\right) + I_z \sin\left(\frac{\pi x}{\tau_p}\right) \sin\left(\frac{\pi y}{\tau_p}\right) \tag{2.2}$$

$$I_{1,2} = I_x \cos\left(\frac{\pi x}{\tau_p}\right) \cos\left(\frac{\pi y}{\tau_p}\right) - I_y \sin\left(\frac{\pi x}{\tau_p}\right) \sin\left(\frac{\pi y}{\tau_p}\right) - I_z \sin\left(\frac{\pi x}{\tau_p}\right) \cos\left(\frac{\pi y}{\tau_p}\right) \tag{2.3}$$

$$I_{2,1} = -I_x \sin\left(\frac{\pi x}{\tau_p}\right) \sin\left(\frac{\pi y}{\tau_p}\right) + I_y \cos\left(\frac{\pi x}{\tau_p}\right) \cos\left(\frac{\pi y}{\tau_p}\right) - I_z \cos\left(\frac{\pi x}{\tau_p}\right) \sin\left(\frac{\pi y}{\tau_p}\right) \tag{2.4}$$

$$I_{2,2} = I_x \sin\left(\frac{\pi x}{\tau_p}\right) \cos\left(\frac{\pi y}{\tau_p}\right) + I_y \cos\left(\frac{\pi x}{\tau_p}\right) \sin\left(\frac{\pi y}{\tau_p}\right) + I_z \cos\left(\frac{\pi x}{\tau_p}\right) \cos\left(\frac{\pi y}{\tau_p}\right) \tag{2.5}$$

then a position independent force in the x, y and z-directions on the magnet array is generated, which is proportional to the currents I_x, I_y and I_z, respectively.

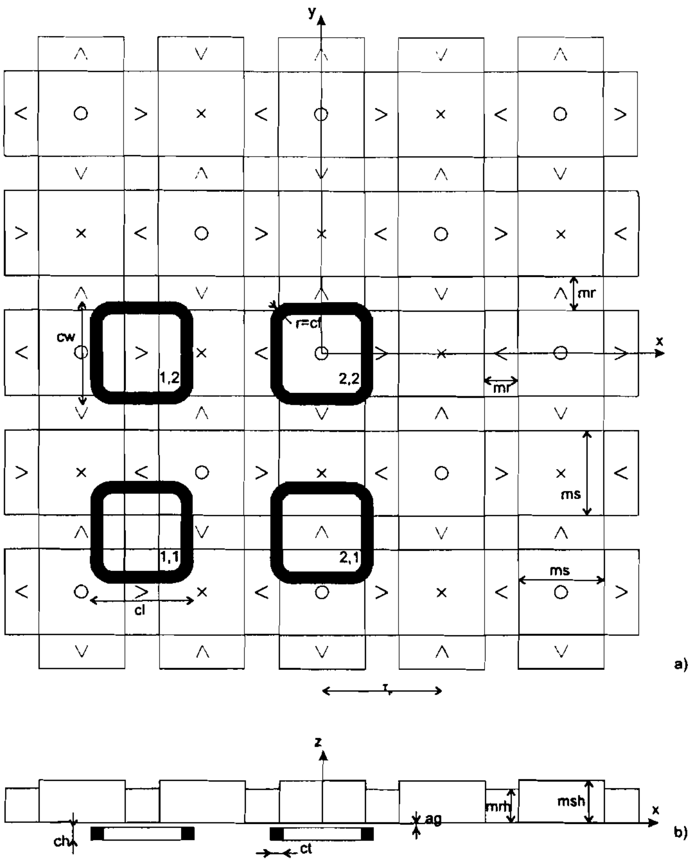


Figure 2.11. *Starting point for the analysis of planar motors. a) top view, b) side view.*

Table 2.2. *Variables of the motor geometry.*

symbol	name
ms	length and width of square magnets
mr	length of short side of rectangular magnets
msh	height of square magnets
mrh	height of rectangular magnets
cl	length of coils
cw	width of coils
ch	height of coils
ct	conductor thickness
cf	fillet radius
ag	air gap height

Chapter 3

Analytical analysis of a planar motor

The analysis of electromechanical converters is rather complicated due to the nonlinear properties of permanent magnets and iron. Therefore, finite element analysis is often used for the calculation of magnetic fields, fluxes, and forces. But for 3-dimensional structures, like planar motors, finite element analysis is an extremely time consuming process and is therefore not suitable for a fast evaluation and optimization of different actuators. However, methods which are based on analytical equations can be used instead of FEM packages. These methods are much faster but less accurate than the finite element methods because the assumption is made that all materials have linear properties.

In this chapter the techniques, based on analytical field solutions, are described for calculating:

- the magnetic field of permanent magnets,
- the magnetic field of current carrying conductors,
- the presence of iron,
- the self and mutual inductances of coils,
- the torques and forces on permanent magnets.

These calculation methods are implemented in Matlab for the analysis of planar motors.

3.1 Field calculation for cuboidal magnets

In literature two different methods can be found for the analytical analysis of 3-dimensional electromechanical structures. In many studies, for example [5], [13], [14], the method of space harmonics field analysis has been used. Both the current in the coils and the equivalent magnetizing current, which models the permanent magnets, are described as Fourier series. The magnetic vector potential is obtained by solving the Poisson equation. The advantage of this method is that it can be employed for permanent magnets with arbitrary shape [13]. But the major disadvantage of this method is that the end-effects of a geometry are not included, because the coil and permanent magnet arrays are assumed to be infinitely long.

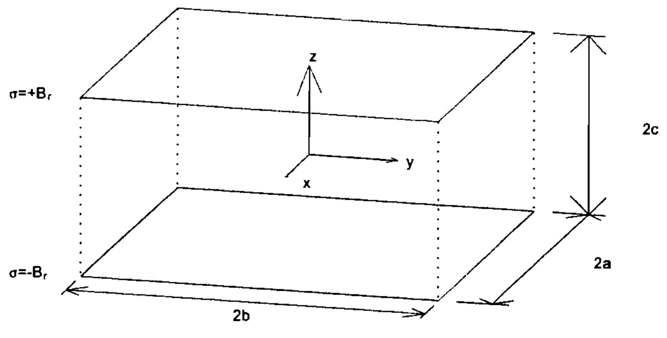


Figure 3.1. *In z-direction magnetized permanent magnet.*

Furthermore, this method is inflexible because new Fourier series has to be formulated for every significant change in the layout of the system.

In [15], another method is presented to calculate the magnetic flux density caused by a permanent magnet and to calculate the forces between permanent magnets. This method is valid for cuboidal magnets with a magnetization perpendicular to one of the sides. The permanent magnet is modelled by two surface charges with opposite polarity. These surface charges represent the sides of the magnet perpendicular to the magnetization direction. Figure 3.1 shows the charged surfaces of a permanent magnet with the lengths $2a$, $2b$ and $2c$ in x, y and z-directions, respectively. The magnetic surface charge (σ) is equal to the remanence of the magnets (B_r). The magnetization direction is in the positive z-direction. The center of the magnet coincides with the center of the coordinate system.

The magnetic flux density due to one of the charged surfaces in a point with coordinates (x, y, z) can be calculated using (3.1). The magnetic flux density caused by the magnet in a point can be calculated by adding the contribution of the fields of the two opposite charged surfaces. The magnetic flux density of an array of magnets in a point can be calculated by summing the flux density contributions of all individual magnets in the array in that point.

$$\begin{aligned} B_x &= \frac{B_r}{4\pi} \sum_{i=0,1} \sum_{j=0,1} (-1)^{i+j} \ln(R_{i,j} - T_j), \\ B_y &= \frac{B_r}{4\pi} \sum_{i=0,1} \sum_{j=0,1} (-1)^{i+j} \ln(R_{i,j} - S_i), \\ B_z &= \frac{B_r}{4\pi} \sum_{i=0,1} \sum_{j=0,1} (-1)^{i+j} \mathrm{atan2}^1 \left(S_i T_j, R_{i,j}(z-c)\right), \end{aligned} \tag{3.1}$$

where

$$\begin{aligned} R_{i,j} &= \sqrt{S_i^2 + R_j^2 + z^2}, \\ S_i &= x - (-1)^i a, \\ T_j &= y - (-1)^j b. \end{aligned} \tag{3.2}$$

[1] A four quadrant arctan-function should be used for the calculation of B_z.

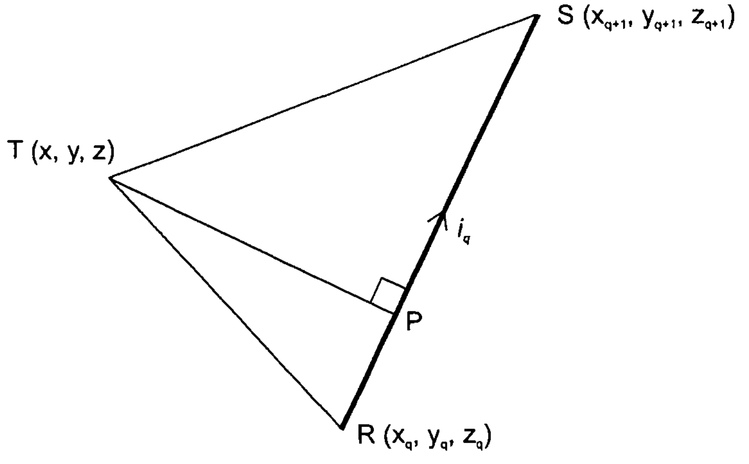


Figure 3.2. *Magnetic flux density and magnetic vector potential calculation in point T due to the current in the filamentary conductor RS.*

The great advantage of using this method is that the end-effects of the array are included in the calculations and that the array of permanent magnets can be described simply. The main disadvantage of this method is that the magnets should have a cuboidal shape.

3.2 Magnetic vector potential and field calculation for current carrying conductors

The calculation of the magnetic vector potential and the magnetic field of current carrying coils is necessary for the inductance calculation, and the force and torque calculation using the Maxwell stress tensor and the virtual work method. As the analytical calculation of the magnetic vector potential and the flux density for coils with real dimensions is unfeasible, these quantities can be calculated by modelling the coil with line currents. In [19], the calculation of the vector potential and the magnetic field density of a filamentary conductor, using the Biot-Savart's law is described.

Figure 3.2 shows conductor RS carrying a current i_q. The vector potential and the magnetic flux density in point T (x, y, z) can be calculated with the next equations. Let:

$$\begin{aligned} a_x &= x_q - x_{q+1}, b_x = x - x_q, c_x = x_{q+1} - x, \\ a_y &= y_q - y_{q+1}, b_y = y - y_q, c_y = y_{q+1} - y, \\ a_z &= z_q - z_{q+1}, b_z = z - z_q, c_z = z_{q+1} - z, \end{aligned} \tag{3.3}$$

then directly:

$$\begin{aligned} \mathrm{RS} &= \sqrt{a_x^2 + a_y^2 + a_z^2}, \\ \mathrm{TR} &= \sqrt{b_x^2 + b_y^2 + b_z^2}, \\ \mathrm{TS} &= \sqrt{c_x^2 + c_y^2 + c_z^2}, \end{aligned} \tag{3.4}$$

and let:

$$\begin{aligned} l_1 &= a_x/\mathrm{RS}, \\ m_1 &= a_y/\mathrm{RS}, \\ n_1 &= a_z/\mathrm{RS}, \end{aligned} \tag{3.5}$$

then:

$$\begin{aligned} \mathrm{RP} &= l_1 b_x + m_1 b_y + n_1 b_z, \\ \mathrm{PS} &= l_1 c_x + m_1 c_y + n_1 c_z, \\ \mathrm{TP} &= \sqrt{\mathrm{TR}^2 - \mathrm{RP}^2}. \end{aligned} \tag{3.6}$$

The magnetic vector potential **A** is equal to:

$$\mathbf{A} = \frac{i_q \mu_0}{4\pi} \left(\sinh^{-1} \frac{\mathrm{RP}}{\mathrm{TP}} + \sinh^{-1} \frac{\mathrm{PS}}{\mathrm{TP}} \right) \begin{bmatrix} l_1 \\ m_1 \\ n_1 \end{bmatrix}. \tag{3.7}$$

Let:

$$\begin{aligned} l' &= (a_z c_y - a_y c_z)/G, \\ m' &= (a_x c_z - a_z c_x)/G, \\ n' &= (a_y c_x - a_x c_y)/G, \end{aligned} \tag{3.8}$$

where:

$$G = \sqrt{(a_z c_y - a_y c_z)^2 + (a_x c_z - a_z c_x)^2 + (a_y c_x - a_x c_y)^2}. \tag{3.9}$$

The magnetic flux density **B** is equal to:

$$\mathbf{B} = \frac{i_q \mu_0}{4\pi \cdot \mathrm{TP}} \left(\frac{\mathrm{RP}}{\mathrm{TR}} + \frac{\mathrm{PS}}{\mathrm{TS}} \right) \begin{bmatrix} l' \\ m' \\ n' \end{bmatrix}. \tag{3.10}$$

3.3 Presence of iron

The presence of iron or other soft-magnetic material can be modelled by the method of images [19]. The influence of a boundary on the applied charges and currents can be calculated by distributing images of the applied currents and charges behind the boundary line. The field is given by the sum of the applied and the image field. The field of a magnetic surface charge, $\sigma = B_r$, in air, representing a permanent magnet with remanence B_r, near a infinite plane of a relative permeability μ_r can be calculated by introducing an image of the surface charge, σ', with the surface charge:

$$\sigma' = -\frac{\mu_r - 1}{\mu_r + 1} B_r. \tag{3.11}$$

The field of a current carrying conductor with current density J in air, near a infinite plane of a relative permeability μ_r can be calculated by introducing an image of the conductor with current density J':

$$J' = \frac{\mu_r - 1}{\mu_r + 1} J. \tag{3.12}$$

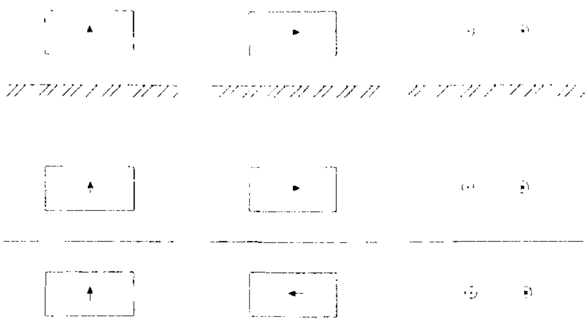


Figure 3.3. *Equivalence of two different magnet-iron systems and a coil-iron system.*

For large values of μ_r, as in iron, the error involved in assuming the relative permeability infinite, is negligible. The system of magnets, coils and iron can then be replaced by the magnets and coils and their exact images. This is shown in Figure 3.3. This assumption is implemented in the tools written in Matlab.

The method of images shows that back-iron behind the permanent magnets doubles the height of the permanent magnets in fact, if the magnetization is perpendicular to the back-iron. If the magnetization is parallel to the back-iron the magnetization of the image is opposite to that of the original magnet. Therefore, the use of back iron behind a Halbach magnet array will have a negligible influence on the magnetic field density distribution.

3.4 Self and mutual inductances

The self and mutual inductances of a coil can be calculated using the magnetic vector potential. However, the vector potential can only be calculated for a line current using 3.7. Therefore, a coil should be built of several line currents, distributed over the physical dimensions of the coil. The numerical technique for distributing the line currents is described in section 3.6.2.

To calculate the flux linkage of a coil, the vector potential along the contour of the coil should be integrated. Finding the right contour is not trivial, because a part of the flux, which is inside the conductor itself, is linked by the conductor. The self-inductance of a coil, L, can be calculated using [16]:

$$L = N \iiint_{coil} \frac{\mathbf{A} \cdot \mathbf{J}}{I^2} \mathrm{d}V \tag{3.13}$$

where N the number of turns of the coil. The mutual inductance, M_{12}, is equal to:

$$M_{12} = N_2 \iiint_{coil2} \frac{\mathbf{A}_1 \cdot \mathbf{J}_2}{|I_1||I_2|} \mathrm{d}V \tag{3.14}$$

where the index 1 is related to the excited coil and the index 2 to the linked coil. The technique to numerically integrate a function, in this case the magnetic vector potential, is described in section 3.6.2.

3.5 Force and torque calculation

A force on the magnet array is generated if a current in applied to the nearby coils. There are different methods to calculate the forces and torques in a system of permanent magnets, current carrying conductors and iron [16]:

- virtual work,
- Maxwell stress,
- Lorentz force.

3.5.1 Virtual work

The virtual work method is the macroscopic approach for force and torque calculations, based on the magnetic energy and coenergy [16]. The force, (**F**), on the magnet array can be obtained by:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial W'_m}{\partial x} \\ \frac{\partial W'_m}{\partial y} \\ \frac{\partial W'_m}{\partial z} \end{bmatrix} \tag{3.15}$$

The torque, (**T**), on the magnet array can be calculated by:

$$\mathbf{T} = \begin{bmatrix} \frac{\partial W'_m}{\partial \psi} \\ \frac{\partial W'_m}{\partial \theta} \\ \frac{\partial W'_m}{\partial \phi} \end{bmatrix} \tag{3.16}$$

where C_e is the coenergy of the permanent magnets.

From this equation the next equation can be derived in which the force is split into three terms. An example, the equation for the force in the x-direction, F_x is given. The equations for the other forces and torques are similar.

$$F_x = I\frac{\mathrm{d}\phi_m}{\mathrm{d}x} + \frac{1}{2}I^2\frac{\mathrm{d}L}{\mathrm{d}x} - \frac{\mathrm{d}W_m}{\mathrm{d}x} \tag{3.17}$$

where ϕ_m is flux from the permanent magnets linked by the coils, L the inductance of the coils and W_m the energy of the permanent magnets.

The first term is the mutual force acting on the coils and magnets, the second and third terms are the reluctance forces, acting on the iron and the coils and acting on the iron and the magnets, respectively. The mutual force is the main force source in the planar motor. Reluctance forces are mainly caused when back-iron is used. If back iron is applied above the magnets, the inductance of the coils depends on the air gap height. If a current is applied to the coils, a force in the z-direction will always be generated. Therefore, the use of back-iron will cause cross-talk between the force and torque components. If back iron is applied below the coils, the energy of the permanent magnets is a function of the air gap height. The permanent magnets are attracted to the iron. This force has to be counteracted to levitate the platform.

The disadvantage of calculating the force using the virtual work method is that the field has to be solved in two slightly displaced positions per degree of freedom. The method is therefore very time consuming if three force and three torque components have to be calculated.

3.5.2 Maxwell stress

Forces and torques can also be calculated using the Maxwell stress tensor. This tensor has to be integrated over a closed surface in air around the magnet array to calculate the total force and torque on the permanent magnet array. The force on the magnets (**F**) is obtained by:

$$\mathbf{F} = \oiint_{magnets} \frac{1}{\mu_0} \begin{bmatrix} \frac{1}{2}(B_x^2 - B_y^2 - B_z^2) & B_xB_y & B_xB_z \\ B_xB_y & \frac{1}{2}(B_y^2 - B_x^2 - B_z^2) & B_yB_z \\ B_xB_z & B_yB_z & \frac{1}{2}(B_z^2 - B_x^2 - B_y^2) \end{bmatrix} \cdot n dS \tag{3.18}$$

The torque (T) on the magnet array is:

$$\mathrm{T} = \oiint_{magnets} \frac{1}{\mu_0} r_s \times \begin{bmatrix} \frac{1}{2}(B_x^2 - B_y^2 - B_z^2) & B_xB_y & B_xB_z \\ B_xB_y & \frac{1}{2}(B_y^2 - B_x^2 - B_z^2) & B_yB_z \\ B_xB_z & B_yB_z & \frac{1}{2}(B_z^2 - B_x^2 - B_y^2) \end{bmatrix} \cdot \mathrm{n} dS \tag{3.19}$$

where n is the unity vector perpendicular to the surface element dS and where r_s is the vector from the mass center point of the suspended platform to the surface element, dS.

The disadvantage of this method is that the surface integral has to be solved over a closed surface surrounding the permanent magnet array. As the surface integral of the Maxwell stress tensor cannot be solved analytically, it should be solved numerically. A dense mesh should be allied because the fields of both permanent magnets and coils vary strongly as a function of the position. In all mesh points both the field of the coils and the field of the permanent magnets should be calculated. Therefore, this calculation is also very time consuming.

3.5.3 Lorentz force

The Lorenz force calculation is the most simple force calculation in a electromechanical system. It can only be used to calculate the mutual force on the coils if they are completely surrounded by air and do not have a soft magnetic core. Reluctance forces cannot be calculated. The force on the magnet array, **F**, is opposite to the force on the coils:

$$\mathbf{F} = -\iiint_{V_{coils}} \mathbf{J} \times \mathbf{B} \mathrm{d}V \tag{3.20}$$

Also the torque on magnet array, **T**, can be calculated. As the point of application of the force on the coil is the same as the force on the magnet, the torque on the magnet is opposite to the torque on the coil:

$$\mathbf{T} = -\iiint_{V_{coils}} r_v \times \mathbf{J} \times \mathbf{B} \mathrm{d}V \tag{3.21}$$

where r_v is the vector from the mass center point of the suspended platform to the volume element, dV.

This method is the fastest method for the force calculation in the planar motor, because only the field of the permanent magnets has to be calculated inside the coil volumes. But the calculation time is directly related to the dimensions and number of coils in the geometry. The disadvantage of this method is that the results are only valid if the reluctance forces can be neglected.

3.6 Matlab implementation

The calculation of magnetic flux density, from magnets and line currents have been implemented in Matlab in order to calculate the forces and torques on the moving platform of the planar motor and the self and mutual inductances of the coils. The program makes use of the Lorentz force calculation because it is the simplest and fastest force calculation. This section describes the structure of the program and the numerical integration algorithm for the Lorentz force and induction calculation. The programming code can be found in the Appendices A, B, C, D and E.

3.6.1 Program structure

The Matlab program is programmed in a flexible way and can be easily adapted for different planar motor configurations. The program can calculate forces and torques on the moving platform with the permanent magnets at different positions. It can also calculate the new position and speed of the moving platform after a certain time-step using Matlab's Aerospace block set. Of course, this is only possible if the mass and inertia of the platform are known. This function has been implemented for future transient simulations.

Two scripts are used to generate the geometry. One scripts generates an array with datasets, called 'structs.' Every struct contains the position, sizes and properties of a magnet. This script can automatically generate Halbach magnet arrays and regular N-S magnet arrays of arbitrary sizes with and without back-iron. The magnets' position is defined in a coordinate system, which is linked to the mass center point of the moving platform. Another script generates an array of structs, which contains the information about the coils in the geometry. Every struct contains the position, size and current information of a single coil. The coils are defined relative to their own coordinate system.

The link between the two arrays of structs, `MagnetArray` and `CoilArray`, containing the magnet and coil information respectively, is a another struct, called `Carpet`. `Carpet` contains the vector defining the position and orientation of the coordinate system of the moving platform relative to the coordinate system of the coils. The forces and torques are calculated in the coordinate system of the moving platform. During the calculation the coordinates of the coils are transformed to the coordinate system of the magnets using the roll-pitch-yaw transformation (see section 3.6.3). This way calculations can be made for every position and orientation of the moving platform. The calculated forces and torques can be used directly as an input for Matlab's Aerospace blockset, to calculate the speed and position of the moving platform in time.

The inductance between the coils is calculated with the script `inductance`. The input of the script are two structs from `CoilArray`. The first input struct is the excited coil, the second input is the linked coil.

3.6.2 Numerical integration method

The magnetic flux density has to be integrated over the coil volumes to calculate Lorentz forces and torques. To calculate the inductance, a coil has to be modelled with line currents, and the mean vector potential has to be integrated along the contour of the coil. Numerical techniques should be applied because the integrals cannot be solved analytically.

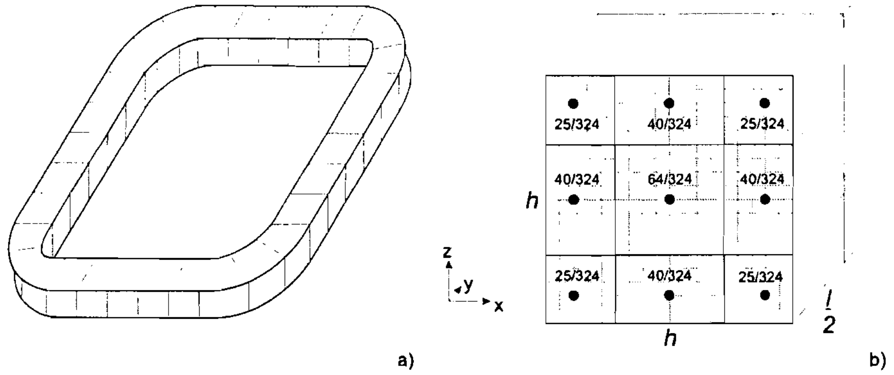


Figure 3.4. *Numerical integration over a coil, a) coil divided in volume elements, b) cross section of one element.*

To calculate the integrals for the forces, torques and inductances, the coil is divided into small elements, which Figure 3.4a shows. Figure 3.4b shows a cross section of one of these volume elements. The current direction and the local contour of the coil is perpendicular to the shown surface. Therefore, this surface is suitable for the calculation of the mean vector potential in that section for the inductance calculation. The surface integral over this section is calculated, using the nine-point approximation of [18]:

$$\iint_S f(x,z)\mathrm{d}x\mathrm{d}z = h^2 \sum_{i=1}^{9} w_i f(x_i, z_i) + \mathcal{O}(h^6) \tag{3.22}$$

where h is the size of the square section, f the function to be integrated and w_i is a matrix with the weight factors for each of the nine points. The value of the function in each point is assumed to be representative for the surrounding area. The points, its weight factors and its areas are also shown in Figure 3.4b. The coordinates of the nine points are $(0,0)$, $(0, \pm\sqrt{\frac{3}{5}}h)$, $(\pm\sqrt{\frac{3}{5}}h, 0)$ and $(\pm\sqrt{\frac{3}{5}}h, \pm\sqrt{\frac{3}{5}}h)$.

The depth of the volume element (in y-direction) is l. So the volume integral over the element is approximated by:

$$\iiint_V f(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z = lh^2 \sum_{i=1}^{9} w_i f(x_i, y, z_i) \tag{3.23}$$

The points and their weight factors are also used to model a real coil with line currents. The coordinates of the points in two adjacent elements can be used to define the beginning and the end of the line currents. The current in each line is proportional to the weight factors.

The nine-points approximation is only valid for square surfaces. In case of non-square conductor sections or very large conductors sections, the accuracy of the nine-point approximation decreases. Figures 3.5a, b, and c shows respectively, three different solutions for a larger non-square conductor section:

- nine-points approximation in the rectangular conductor,

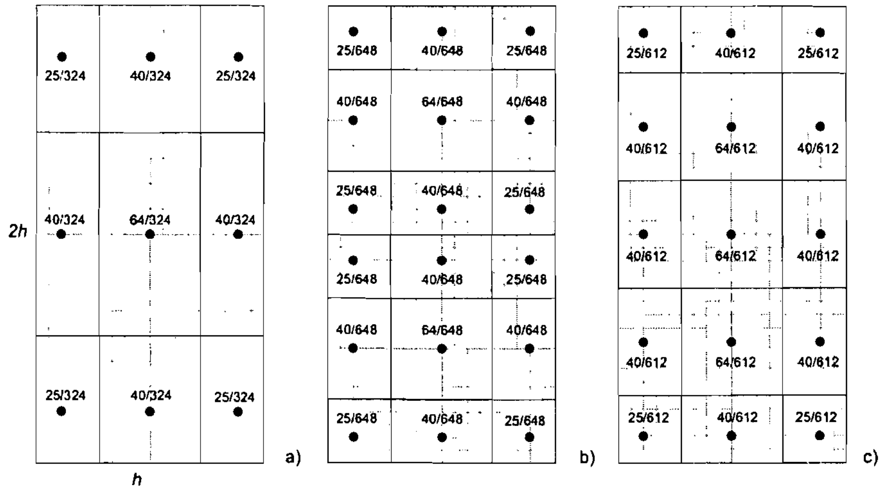


Figure 3.5. *Three approaches of the surface integral over a non-square coil section, a) 9-points approximation, b) division into square sections, c) insertion of extra points.*

- division of the rectangular section into two square sections, in which the nine-point approximation is used,

- insertion of extra points with a certain density, distributed according to the nine-point approximation method.

The last method has been implemented in Matlab, because it is the best way to control the density of the points and the mean size of the volume elements, independent of the shape of the conductor section.

3.6.3 Transformation from global to local space

The coils of the planar motor are defined in the global space and the magnets of the motor are placed in a local space. To calculate the forces on the magnet array in its local space, the coordinates of the coils and the current density directions should be transferred to the local space of the magnets. A vector in Carpet contains the position and orientation of the magnet array, defined in the space of the coil array. The necessary rotations and translations are defined with respect to the global reference; the coordinate system of the coils. Therefore, the roll-pitch-yaw transformation ([17]) is used to transfer from one system to the other. Besides, it is the same transformation, which is implemented in Matlab's Aerospace blockset. Therefore, this blockset can be easily integrated in the tools.

The transformation matrix from the coordinates in magnet space to the coordinates in coil space, ${}^cT_m$ is:

$$ {}^cT_m = \text{Trans}(z, p_3)\text{Trans}(y, p_2)\text{Trans}(x, p_1)\text{Rot}(z, \phi)\text{Rot}(y, \theta)\text{Rot}(x, \psi), \quad (3.24)$$

where the position of the magnet array is specified by (p_1, p_2, p_3). The translation matrix is:

$$\mathrm{Trans}(z, p_3)\mathrm{Trans}(y, p_2)\mathrm{Trans}(x, p_1) = \begin{bmatrix} 1 & 0 & 0 & p_1 \\ 0 & 1 & 0 & p_2 \\ 0 & 0 & 1 & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.25}$$

The rotation matrices are:

$$\mathrm{Rot}(x, \psi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.26}$$

$$\mathrm{Rot}(y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.27}$$

$$\mathrm{Rot}(z, \phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.28}$$

During the force calculation, the coordinates of the coils are calculated in the magnet space. This transformation, ${}^{m}T_{c}$ is equal to:

$${}^{m}T_{c} = ({}^{c}T_{m})^{-1}. \tag{3.29}$$

Due to these transformations, the planar motor can be simulated for all positions and orientations of the magnet array with respect to the coil array.

Chapter 4

Three-dimensional finite element methods

Numerical methods are of the outmost importance for the design of electromechanical actuators. These methods can accurately predict the properties of electromechanical actuators, such as the magnetic and electric field distribution, losses, temperature rise, efficiency and dynamic behavior. The accuracy of the numerical method is much better than that of analytical methods, when the actuator has a complex geometry or consists of materials with nonlinear characteristics.

The finite element method (FEM) is the most commonly used techniques for solving electromechanical problems, because FEM can be applied to a wide variety of problems in bounded regions: stationary and transient, linear and nonlinear, isotropic and anisotropic. Nowadays, there are different software packages for three-dimensional finite element calculations, available on the market. Maxwell 3D (Ansoft) and FLUX3D (Cedrat) are among the most sophisticated packages for the Windows platform. These packages are reviewed for the usefulness in the planar motor project.

4.1 FLUX3D

FLUX3D is a three-dimensional finite element program developed by the Cedrat group. The reviewed version is 3.30. The software can solve three-dimensional problems of three different types:

- magnetic problems, i.e. magneto static, magneto dynamic and transient problems. The transient problems can be coupled with circuit equations, rotations, and in the near future translations.

- electric problems, i.e. electrostatic, dielectric and electric conduction problems.

- thermal problems, i.e. steady-state and transient thermal problems.

Mainly, the magneto static solver has been reviewed because the package, which solves transient problems with translations was not available. Two models can be used for magneto static problem: a scalar model that uses formulations in magnetic scalar potentials and a vector model, that uses magnetic vector potentials with nodal approximations. However, the last formulation is not recommended by the manufacturer because of the limited accuracy and the required amount of memory.

Two different scalar models can be used. The most used model (called MS3RED / MS3SCA) describes ferromagnetic regions and magnets in total scalar potential (MS3SCA). The air regions and the source fields of coils in the air are described in reduced scalar potential (MS3RED). The coils in the air are non-meshed coils regions. The source field of these coils is calculated by Biot-Savart's law. This seems to be an interesting option for the simulation of the planar motor because no re-meshing is required if the position or dimension of the non-meshed coils are changed. However, the force calculations, which are based on virtual work, are not accurate because the magnetic field due to the coils is relatively small. Therefore, the support department of Cedrat recommends in this case the calculation of the lorentz force in another model.

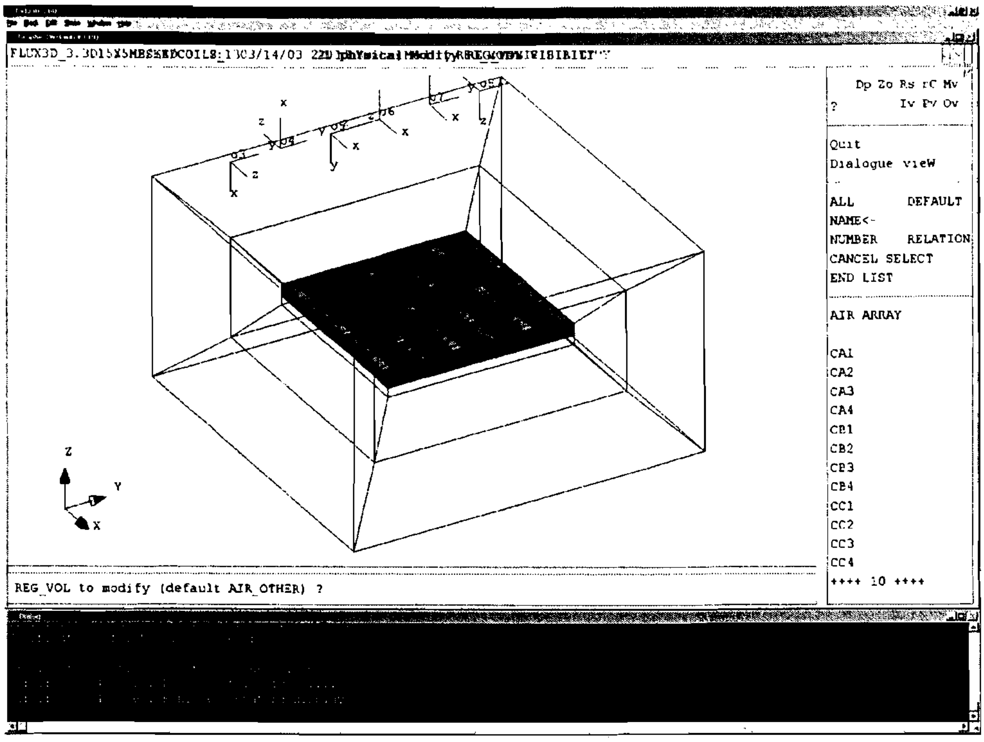
In the other model (MS3SCRTOW / MS3SCA) the MS3SCRTOW formulation, a reduced potential with respect to the electric vector potential, is used in stead of MS3RED. In this formulations, non-meshed coils are forbidden and re-meshing is necessary after every geometric change. The force calculation based on virtual work is also inaccurate in this formulation but the Lorentz force calculation is accurate. Moreover, the calculation of the Lorentz force is faster than the calculation with the virtual work method. A disadvantage of this formulation is that closed ferro-magnetic circuits, like in transformers, are forbidden, but they do not exist in motors.

FLUX3D is fully parameterized, which finds expression in the structure of the program. Mathematical functions can be used to link the parameters together. A device is built in a domain, which takes into account the symmetries, periodicities and the external layer of air or vacuum surrounding the device, the so-called infinite box. In this domain local and global coordinate systems can be defined. In these coordinate systems geometries can be defined, which can be displaced by moving the origin or orientation of that particulary coordinate system. Volumes cannot be created directly. First points, lines and surfaces have to be defined successively, using extrusions, propagations, automatic and manual tools. After creation of the device, volumes can be grouped in regional volumes, which define the material, the sources, the coordinate system, and the visibility of the region. Most of these properties can be parameterized, for example to link the current density in a coil to the position of the coil. Figure 4.1 shows the layout of FLUX3D.

After creation of the geometry and the assignment of the regions, the structure can be meshed manually only. Both tetrahedral and hexahedral second order elements can be used. The mesh can be assigned easily. Mesh generators can be created to assign a mesh to points, lines, surfaces or volumes. In practice, assigning mesh generators to most lines is sufficient to create a proper tetrahedral mesh. The mesh size is limited by the amount of memory in the computer. Problems with approximate 1,000,000 elements can be solved, although the program becomes very slow. Problems with approximate 500,000 elements are more practicable and need 30 to 60 minutes to be solved.

Before starting the solving process, sensors can be created for the automatic calculation of quantities, such as forces, torques and fluxes after the solving process. Sensors can contain both predefined and user-defined equations. Furthermore, the variation of parameters can be defined to create a parameterized solution.

The post-processor of FLUX3D is highly satisfactory and provides all necessary visualization and calculation functions. Data from arbitrary surfaces and lines



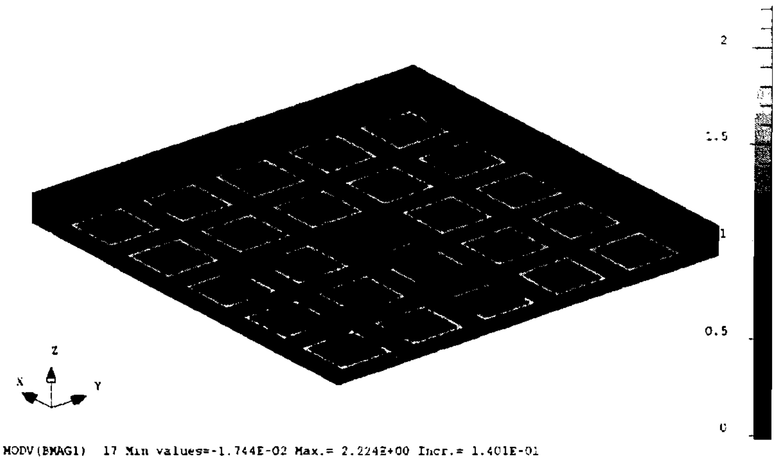
```
REG_VOL to modify (default AIR_OTHER) ?
```

Figure 4.1. *Screen from FLUX3D: the magnet array with differently coloured volume regions, visualizing the magnets with different magnetization directions, the infinite box of the domain and the coordinates systems, used to define the magnetization directions.*

can be easily exported in every coordinate system to for example Matlab. The disadvantages of the post-processor are that the plot functions are sometimes slow and that the orientation of the geometry on the screen cannot be altered with the mouse. Figure 4.2 shows a screen from the post-processor.

The 2D version of FLUX, FLUX2D, is contrary to FLUX3D not integrated in one program. Therefore, the program is less flexible and the structure is sometimes unclear, especially during the definition and assignment of materials. However, the post- processor is very extensive and gives very detailed information of the solutions, parameters and materials.

Both FLUX2D and FLUX3D are only to a certain extent useful for solving the thermal problems of the planar motor. As the platform moves in vacuum only radiation from the coil surface and eddy currents in the magnets can heat the platform. However, only the emissivity of the radiation from a surface can be defined and not the absorption and reflection. Therefore, the heating by radiation of the moving platform cannot be calculated but only the heat radiation into the vacuum.



```
MODV(BMAG1)  17 Min values=-1.744E-02 Max.= 2.224E+00 Incr.= 1.401E-01
```

Figure 4.2. *Screen of the post-processor of FLUX3D: Magnitude of the magnetic flux density on the magnets of a Halbach magnet array and on the four coils.*

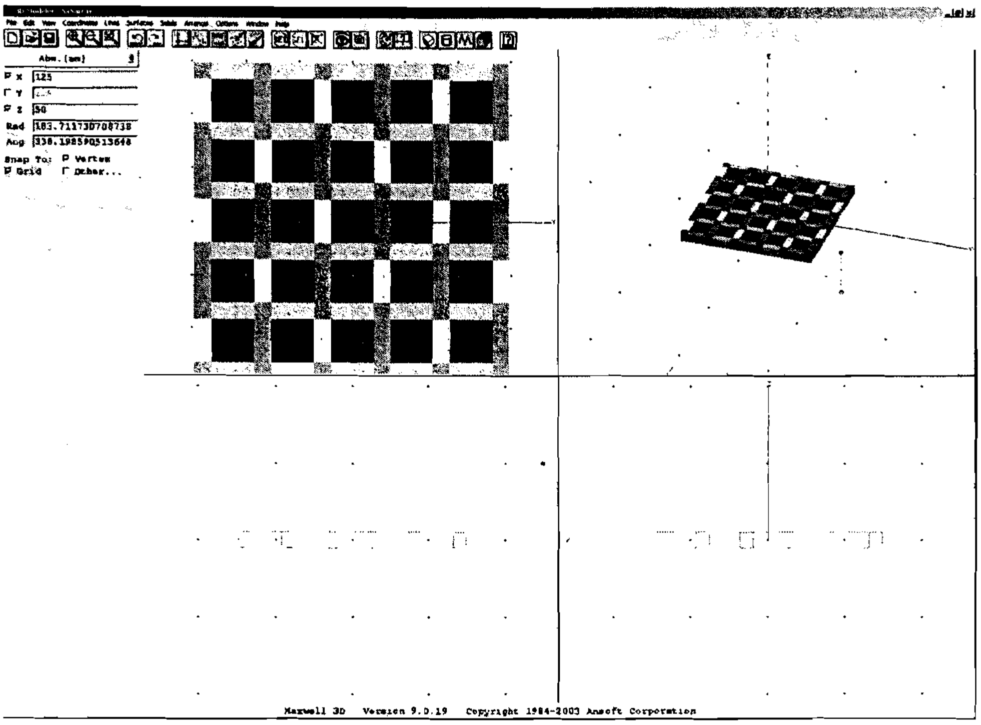
4.2 Maxwell 3D

Maxwell 3D is a three-dimensional finite element program developed by Ansoft Corporation. It can solve static, dynamic and transient magnetic problems, electrostatic fields and steady-state thermal problems. The transient solver can only solve problems with arbitrary current and voltage waveforms, but not with moving objects like rotors.

The structure of Maxwell 3D is in many respects different from FLUX3D. For example, volumes can be created directly. The volumes can be defined in different coordinate systems, but they are not linked to that coordinate system. The dimensions of the volumes cannot be parameterized. Parameterizing of a device is only possible via the macro language or by buying the optional program called Optimetrics. The available license did not include Optimetrics and the availability was too short for a proper review of the macro language. Figure 4.3 shows a screen of the drawing program. The screen is divided into four windows containing the three different normal views and a perspective view on the device.

After the drawing process, materials can be assigned to the volumes. These can be selected from the material database or created by the user. Thereafter, the boundary conditions and the sources are assigned. Although the assignment of the sources is easier than in FLUX3D, no mathematical functions can be used for the calculation of the current. Non-meshed coils are not included in Maxwell 3D.

Before the solving process the volumes have to be selected and grouped for the force, torque and flux calculations. Both force calculation by virtual work and Lorentz forces are supported. Maxwell 3D uses an adaptive mesh generator, which creates tetrahedral elements. The problem is solved with an initial mesh first. Then the meshed is refined in the volumes with the largest energy error and is solved again. This process repeats until the energy error is below a value, entered by the user or until the maximum number of elements is reached. The maximum number of elements is approximately 250,000 elements. The automatic remeshing procedure



☑ X 125
☐ Y
☑ Z 50
Rad 183.711730708738
Ang 338.198590513648
Snap To: ☑ Vertex
☑ Grid ☐ Other...

Maxwell 3D Version 9.0.19 Copyright 1984-2003 Ansoft Corporation

Figure 4.3. *Screen from Maxwell 3D: A device is drawn, using three normal views and a perspective view*

has problems with cylindrical volumes with small radii, such as the corners of coils. In those volumes the element size decreases much faster than in the surrounding volumes of the coil. The initial mesh can also be created by the user himself. However, the user has little control on the position and shape of the elements.

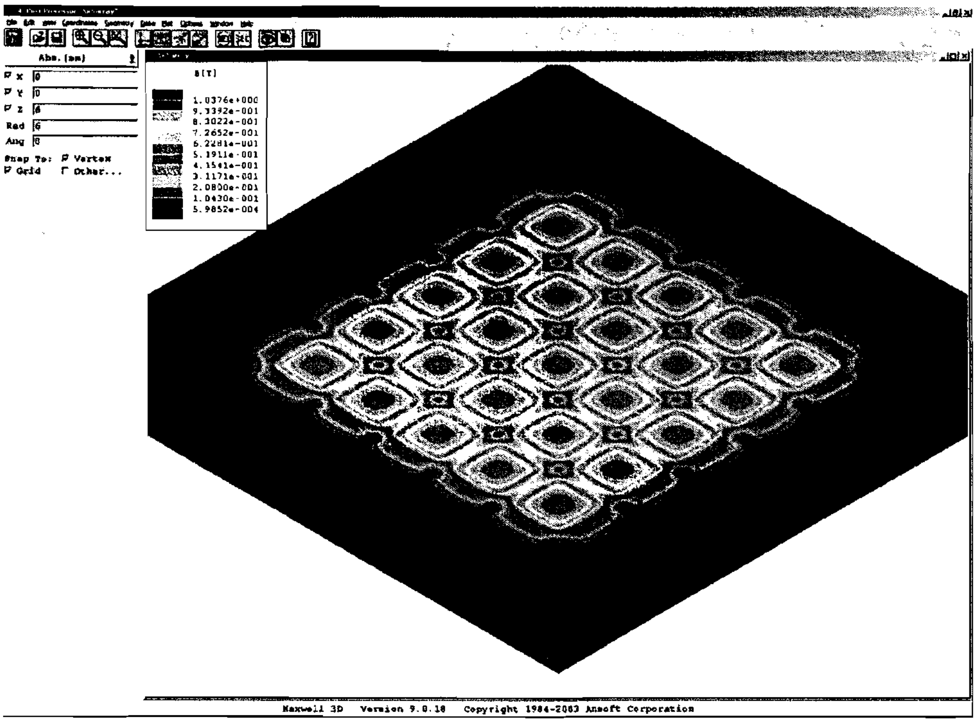
The post-processor has the same functionality as the FLUX3D's post-processor. However, Maxwell 3D is faster but the user interface and the quality of the graphics are worse. Data from the post-processor can only be exported in the main coordinate system. A list of the data points should be provided to the program by the user himself and cannot be generated by the program automatically. An advantage of Maxwell 3D is that the device can be rotated with the mouse. Figure 4.4 shows a section of a Halbach magnet array in the post-processor.

Also Maxwell 3D is not suitable for a full thermal analysis of the planar motor, as no absorption or reflection of thermal radiation can be defined on a surface.

4.3 Conclusions

The major differences between FLUX3D and Maxwell 3D are:

- FLUX3D allows to solve larger devices than Maxwell 3D because of the larger maximum number of elements.
- Although manual meshing takes more time, FLUX3D allows better control of the mesh. However, the automatic mesh generator of Maxwell 3D automati-



☑ X 0
☑ Y 0
☑ Z 6
Rad 6
Ang 0
Snap To: ☑ Vertex
☑ Grid ☐ Other...

B[T]
1.0376e+000
9.3392e-001
8.3022e-001
7.2652e-001
6.2281e-001
5.1911e-001
4.1541e-001
3.1171e-001
2.0800e-001
1.0430e-001
5.9852e-004

Maxwell 3D Version 9.0.18 Copyright 1984-2003 Ansoft Corporation

Figure 4.4. *Screen from the post-processor of Maxwell 3D: Magnitude of the magnetic flux density above a Halbach magnet array.*

cally identifies volumes, which need a denser mesh and, therefore, reduces the flux density errors at the magnet edges.

- FLUX3D is completely parameterized and can be controlled by macros. Maxwell 3D can only be parameterized by using macros or by the - so far not reviewed - Optimetrics package.

- The possibility to solve transient problems with linear motion is only available in FLUX3D.

Considering the above mentioned items, FLUX3D is the most extensive package and more suitable for three-dimensional finite element analysis of a planar motor than Maxwell 3D.

Chapter 5

Comparison between analytical tools and finite element method

In chapter 3, the tools for the analysis of planar motor geometries are presented, which are based on analytical equations for the solution of the magnetic field. In this chapter, these tools, which are implemented in Matlab release 12, are compared with the two finite element packages, Maxwell 3D and FLUX3D, which are reviewed in chapter 4. Three different types of calculations are compared, using the geometry, presented in section 2.6:

- magnetic flux density calculation,
- force calculation,
- self and mutual inductance calculation.

The analytical tools assume ideal magnets with a relative permeability, μ_r, equal to 1. To estimate the practical usefulness of the analytical tools, the magnets in the finite element packages have a relative permeability of 1.033. The remanence magnetization of the magnets, B_r, is equal to 1.3 T for both the tools in Matlab and the finite element packages. These numbers are realistic for high quality NdFeB magnets.

The detailed pictures of the models and its meshes in FLUX3D and Maxwell 3D can be found in Appendix F. As mentioned before, the availability of the finite element packages was limited, especially for Maxwell 3D. Therefore, not all calculations could be carried out in Maxwell 3D.

5.1 Magnetic flux density calculation

The calculation of the magnetic flux density in a point, caused by the permanent magnets, is very important for the analysis of the planar motor. This quantity can be calculated without numerical integration using the Matlab tools. The magnetic flux density calculation by the Matlab tools and by the finite element programs have been compared on a 200 mm long line parallel to a Halbach magnet array at three distances from the array.

Figure 5.1 shows the Halbach array and the 200mm long line, b-b'. The start of the line is at $(-0.05\sqrt{3}, -0.082)$ and the end is at $(0.05\sqrt{3}, 0.018)$. This line is chosen to avoid any symmetries of the magnet array on the line. The values of the motor geometry variables are summarized in Table 5.1.

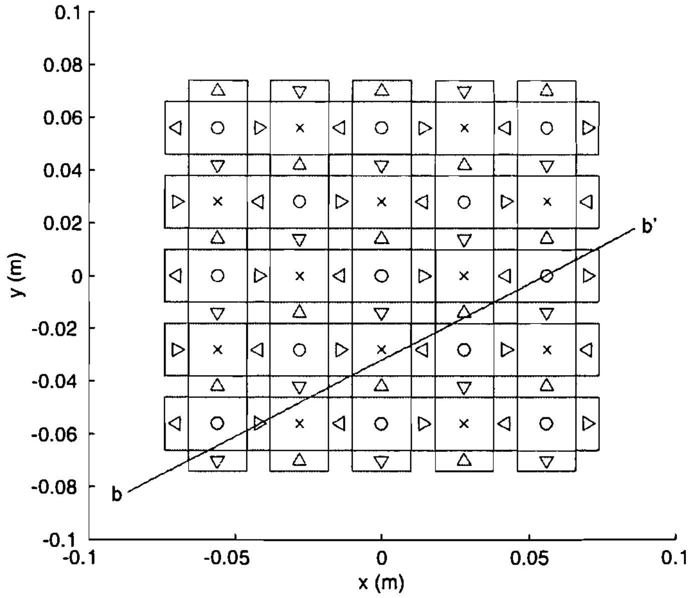


Figure 5.1. *Top view of the Halbach magnet array and the line, b-b', on which the magnetic flux density is determined. The symbols indicate the magnetization direction of the magnets.*

Table 5.1. *Variables of the motor geometry for magnetic flux density calculation (see Figure 2.11).*

symbol	*name*	*value*
ms	length and width of square magnets	20 mm
mr	length of short side of rectangular magnets	8 mm
msh	height of square magnets	10 mm
mrh	height of rectangular magnets	10 mm

Figures 5.2, 5.4, 5.5 show the magnitude of the magnetic flux density distribution along the line at 1 mm below, 5 mm below and 1 mm above the array with permanent magnets, respectively. The blocks at the bottom sides of the figures indicate the cut-off structure of the magnet array. The arrows indicate the magnetization direction of the square magnets, the letter 'a' indicates air. Figure 5.3 shows the relative error between the finite element calculations and the analytical calculation.

From the results can be concluded that the calculation of the magnetic flux density by the Matlab tools coincides with the finite element calculations. Figure 5.3 shows that the largest errors between the analytical and finite element calculation occur at the edges of the magnets. This error is related to the used mesh. This is clearly visible when the results of FLUX3D in Figure 5.2 and Figure 5.5 are compared. A worse mesh has been used deliberately for the latter calculation. This example shows clearly the advantage of the automatic mesh generator of Maxwell

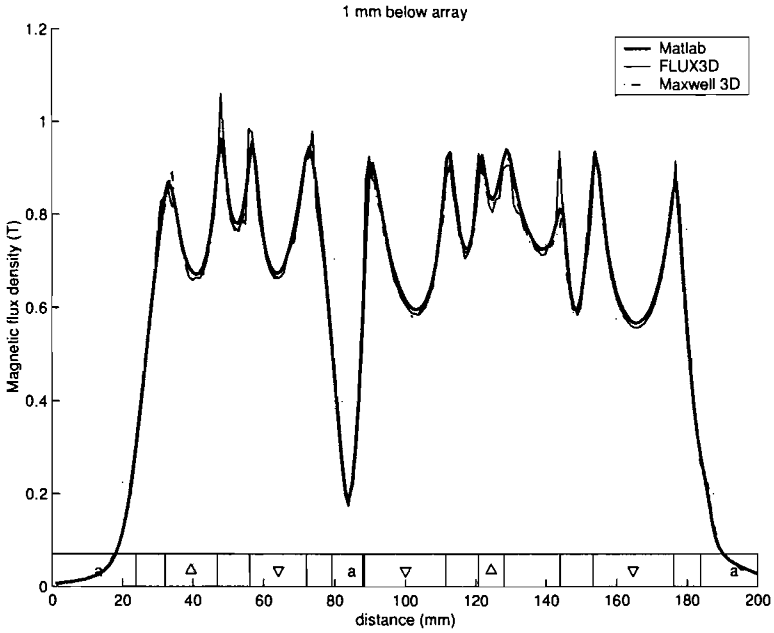


Figure 5.2. *Magnitude of the magnetic flux density at 1mm below the magnet array.*

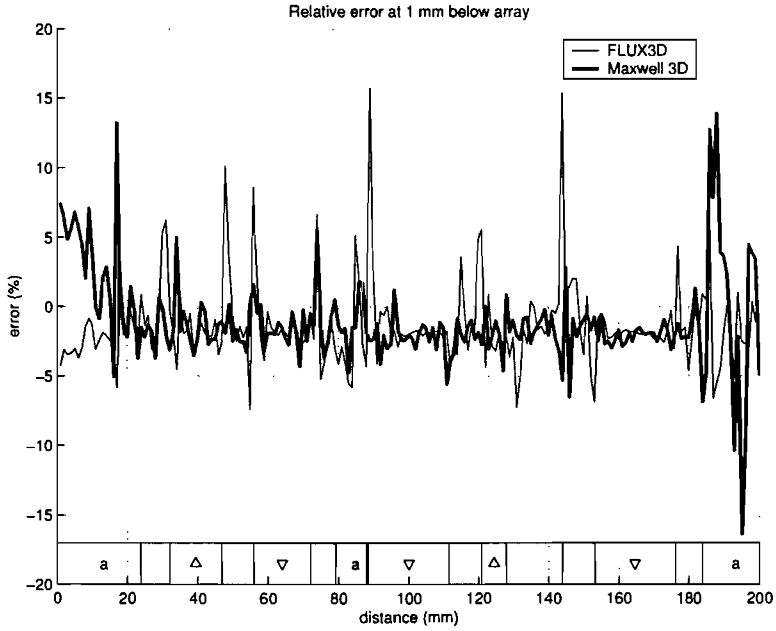


Figure 5.3. *Relative error between the magnetic flux density calculated by the analytical tools and the finite element packages at 1 mm below the magnet array.*

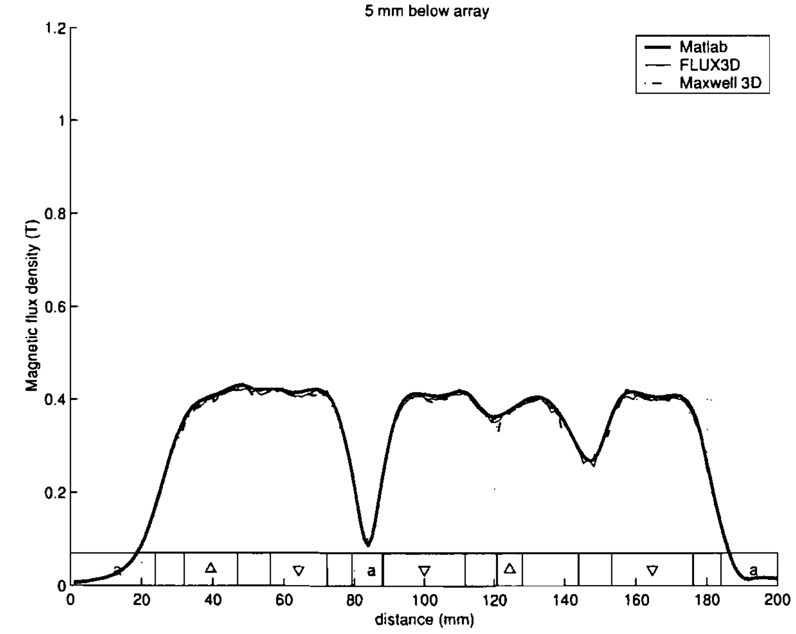


Figure 5.4. *Magnitude of the magnetic flux density at 5 mm below the magnet array.*

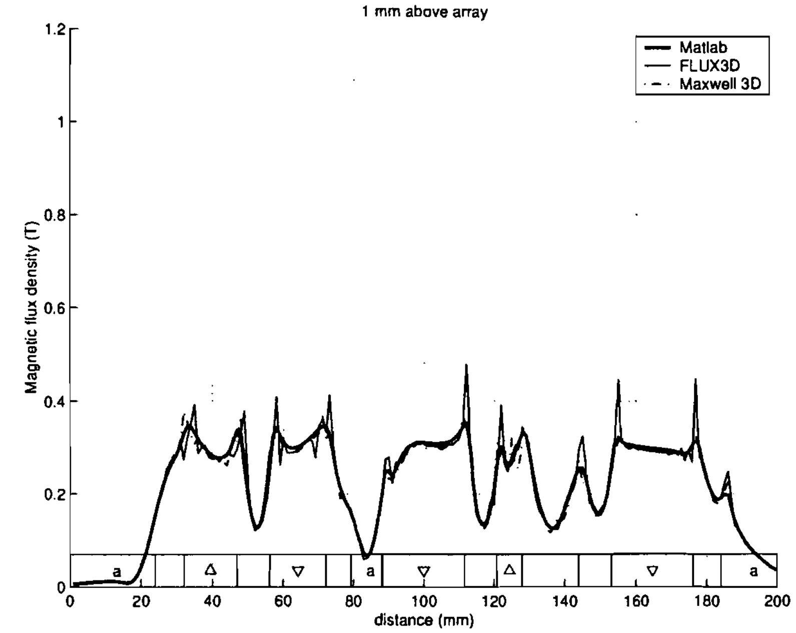


Figure 5.5. *Magnitude of the magnetic flux density at 1 mm above the magnet array.*

Table 5.2. *Variables of the motor geometry for the force calculation.*

symbol	*name*	*value*
ms	length and width of square magnets	20 mm
mr	length of short side of rectangular magnets	8 mm
msh	height of square magnets	10 mm
mrh	height of rectangular magnets	10 mm
τ_p	pole pitch	28 mm
cl	length coil	24 mm
cw	width coil	24 mm
ch	height coil	3 mm
ct	conductor thickness	3 mm
cf	fillet radius	4 mm
ag	air gap	1 mm

3D, which increases the mesh density in areas with a large energy error or a strong change of the flux density. Therefore, these errors disappear at larger distances from the array (see Figure 5.4). Figure 5.3 also shows that the results of the finite element calculations are generally smaller (about 2%) than the analytical results. The reason for this is the use of realistic magnets with a 3.3% larger relative magnetic permeability in the finite element programs.

5.2 Force calculation

For the analysis and optimization of the planar motor, the force calculation is very important for the dimensioning process. Therefore, the force calculation should be fast but also accurate. The force calculation by the Matlab tools has been compared with the force calculation by the two finite element packages. The comparison has been made by simulating a magnet array above 4 coils at 16 different positions of the magnet array. As Maxwell 3D was only available for a very short period and as it is not parameterized, only one magnet position has been calculated in Maxwell 3D. Figure 2.11 shows the top and side views of the used motor geometry. Table 5.2 contains the specifications of the geometry.

A position dependent current has been applied to the coils to generate a constant force in both x- and z-directions using (2.2), (2.3), (2.4) and (2.5), where $I_x = 25$ A, $I_y = 0$ A and $I_z = 50$ A. The magnets have been moved from the initial position (0, 0) to end position (-0.007,-0.007) in 16 steps of -0.00233 m in both the x and y-directions.

At position (-0.007,-0.007) the influence of the mean element volume on the numerical integration of Matlab tools has been determined. Figure 5.6 shows the relation between this element size and the force error with respect to the force calculated with a volume element size of 0.07 mm^3. Besides it shows the calculation time. The figure shows that the force calculation can be both fast (80 seconds per position for 4 coils with these dimensions and a set of 85 magnets) and accurate (less than 0.12% error) if a mean element size of 1 mm^3 is chosen.

Figure 5.7 shows a comparison between the force calculation results obtained in Matlab and in FLUX3D. The left diagrams show the results of the force calculation

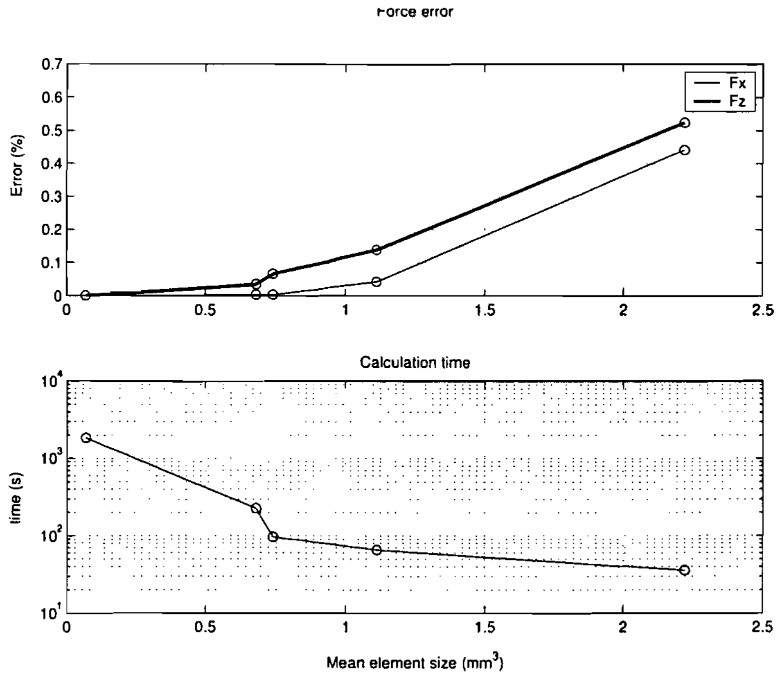


Figure 5.6. *Relation between the element volume and both the force error and the calculation time.*

for the three force components (F_x, F_y, F_z) in Matlab at the different positions of the magnet array. In the right diagrams the relative force error between FLUX3D and Matlab with respect to the Matlab calculation is shown. The force values calculated by FLUX3D are generally lower (1-2%) because realistic magnets have been used for the FLUX3D calculations. The y-component of the force, F_y, is equal to 0 N for $y = 0$. The analytical tools calculate this force correctly. However, FLUX3D cannot calculate such a small force but calculates a force of less than 0.02 N. Therefore, the absolute error is shown for F_y.

For the magnet position (0,0) also the result from Maxwell 3D is available. The forces, calculated by the different programs, are:
Maxwell 3D: $F_x = 0.4925$ N, $F_y = 0.01211$ N, $F_z = 1.8538$ N,
FLUX3D: $F_x = 0.5501$ N, $F_y = 0.01613$ N, $F_z = 1.9209$ N,
Matlab: $F_x = 0.5473$ N, $F_y = -1.719 \cdot 10^{-15}$ N, $F_z = 1.944$ N.
The lower force result from Maxwell 3D is probably caused by the relatively low number of elements in the coils.

The force calculation by the analytical tools is consistent with the force calculation by the finite element packages. The application of realistic magnets in the finite element calculations results in a lower force level, which coincides with the results of the magnetic flux density calculation. The calculation speed of Matlab (80 seconds) is much faster than the speed of FLUX3D (45 minutes) and Maxwell 3D (90 minutes). However, the calculation time increases much for larger geometries with more coils and magnets and for larger conductors. Therefore, a faster implementation on another platform could be useful in the future.

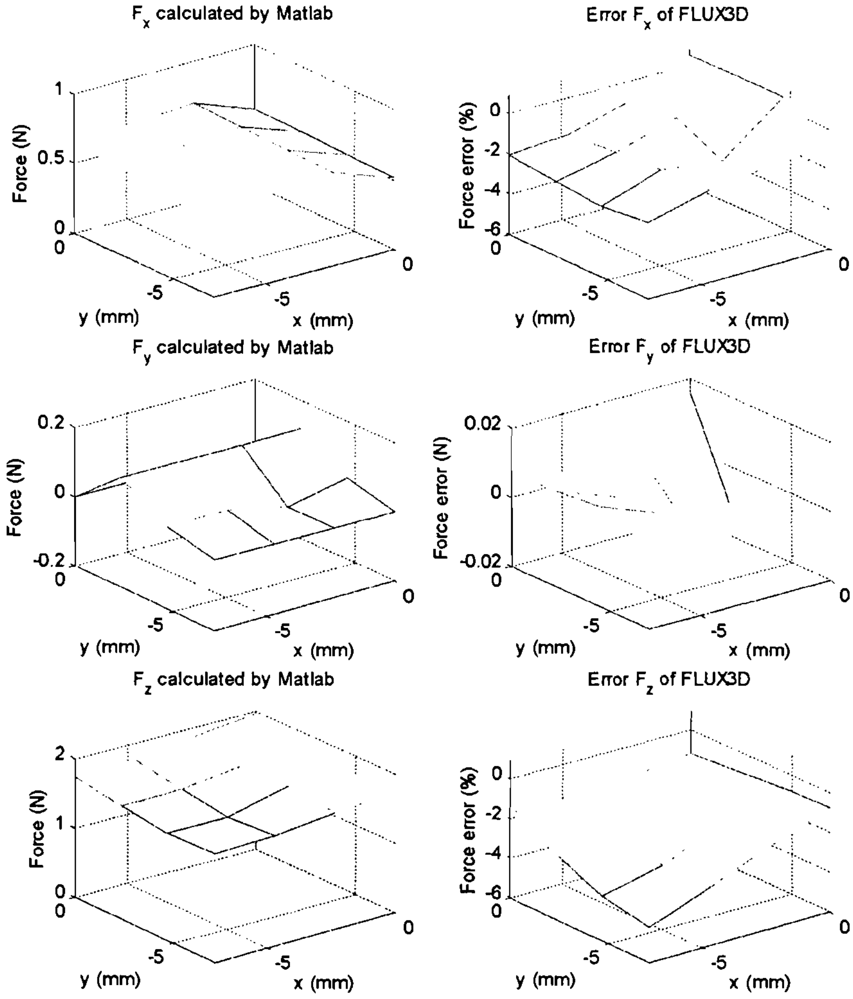


Figure 5.7. *Comparison of force calculation by Matlab and FLUX3D. The relative errors of FLUX3D are calculated with respect to the Matlab calculation.*

5.3 Inductance calculation

The calculation of the inductances of the coils is important to predict the dynamic behaviour of the planar motor. The flux linkage of each coil in the geometry in Figure 2.11 has been calculated, while a current of 1 A is applied to coil (2,2). This flux linkage can be used to calculate the self and mutual inductances of the coils, if the number of turns in each coil is known.

The same element volume can be used for the inductance calculation as for the force calculation. Also in this case the results change significantly if a element with a volume of less than 1 mm^3 is used. Also the number of line current, which

Table 5.3. *Flux linkage of the coils with coil (2,2).*

coil	*Matlab*	*FLUX3D*
(1,1)	$-9.81 \cdot 10^{-11}$ Wb	$-1.02 \cdot 10^{-10}$ Wb
(1,2)	$-3.14 \cdot 10^{-10}$ Wb	$-2.92 \cdot 10^{-10}$ Wb
(2,1)	$-3.14 \cdot 10^{-10}$ Wb	$-2.92 \cdot 10^{-10}$ Wb
(2,2)	$2.96 \cdot 10^{-8}$ Wb	$3.34 \cdot 10^{-8}$ Wb

models the real conductor, hardly influences the result. In this case 16 line currents are used for the simulation of a coil (1.8 line currents/mm^2 of the conductor cross section).

In Table 5.3 the flux linkages of the coils, calculated by Matlab and FLUX3D, are summarized. The error between the finite element calculations and the analytical calculation is 4-13%. This large error is not related to the permanent magnet array above the coils, because the value of the relative permeability, e.g. 1 or 1.033, does not influence the results of FLUX3D.

5.4 Conclusions

The tools, based on the analytical equations which are implemented in Matlab, are a good alternative for the finite element simulations. The tools are much faster and flexible than the finite element software packages. Generally, the results of the Matlab tools are 2-3% higher than the finite element results because ideal magnets ($\mu_r = 1$) are assumed. The best way to validate the Matlab tools and the results from the finite element simulations is to carry out experiments.

Chapter 6

Optimization of a planar motor structure

The geometry of the planar motor should be optimized to reach its specifications. At this moment, the exact target specification of the planar motor, such as the weight and the size of the moving platform, are unknown. However, the demand that the planar motor should have a position independent force and torque is obvious. The optimization of all variables of the planar motor is very complicated and not useful if not all specifications are defined. However, the influence of certain variables on the performance of the planar motor can be easily investigated, while other variables are fixed. Besides, the feasibility of a design, concerning the dissipation in the coils, can be predicted.

In this thesis, the planar structure has been optimized for the minimization of the force ripple. Then, the ideal ratio between the mass of the magnet array and the mass of the total platform has been determined, which maximizes the acceleration. The currents in the coils to suspend the platform are calculated for platforms with different masses. Based on this current, the minimal losses of the motor are estimated.

6.1 Optimization setup

To speed up the optimization, a smaller geometry than the geometry in section 2.6 is optimized. Only the force produced by two coils, while moved in only the x-direction has been calculated. The force generation in the y-direction has not been taken into account. By doing so, a small error is introduced, because the cross-talk with the y-axis is neglected. Figure 6.1 shows the geometry which has been optimized. Table 6.1 summarizes the (possible) values of the variables during the optimization process. The length of short side of rectangular magnets, mr, and the conductor thickness, th have been optimized. The air gap height, the size of the square magnets and the fillet radius are fixed during the optimization. The coil height is also fixed on ch = 5 mm, in order to speed up the calculation. The coil height can be extended further to increase the force density of the planar motor since the coil mass does not add to the moving mass in a moving magnet planar motor. Furthermore, the length and width of the coil are adapted to the conductor thickness and the size of the rectangular magnets. The distance between the centers of the conductors of a coil is exactly equal to 180 electrical degrees, as shown in Figure 6.1b. The distance between the centers of the conductors of two coils is

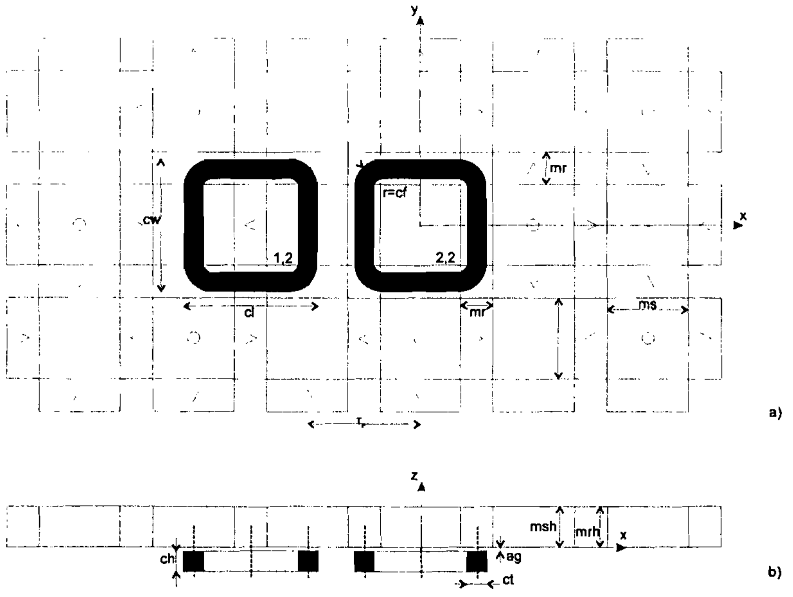


Figure 6.1. *Motor geometry to be optimized, a) top view, b) side view.*

Table 6.1. *Variables of the motor geometry.*

symbol	name	value (mm)
ms	length and width of square magnets	20
mr	length of short side of rectangular magnets	$5 < mr < 20$
msh	height of square magnets	5 and 10
mrh	height of rectangular magnets	$mrh = msh$
cl	length of coil	$cl = ms + mr + ct$
cw	width of coil	$cw = cl$
ch	height of coil	5
ct	conductor thickness	$5 < ct < \frac{1}{2}(ms + mr)$
cf	fillet radius	1
ag	air gap	1

exactly equal to 90 electrical degrees. This way the force is maximized and the ripple is minimized. For the same reason, all the magnets in the array have the same height.

Two optimizations have been carried out with the Halbach magnet array with a magnet height, mh of 5 and 10 mm. Two other optimizations have been carried with a regular N-S magnet array without the rectangular magnets. In that case, mr defines the distance between the magnets.

Every optimization step, the magnet array is moved 90 electrical degrees or half a pole pitch, τ_p, in negative x-direction in 10 steps. The current in the coils is

Table 6.2. *Mean force and the percentage force ripple in the optimized planar motors.*

Magnet array	F_{x_m}(N)	F_{x_r}(%)	F_{y_m}(N)	F_{y_r}(%)	F_{z_m}(N)	F_{z_r}(%)
Halbach, mh = 5mm	1.62	2.06	1.62	2.06	2.30	0.94
Halbach, mh = 10mm	2.52	1.47	2.52	1.47	3.57	0.50
Non-Halbach,mh = 5mm	0.814	1.53	0.814	1.53	1.154	1.51
Non-Halbach, mh = 10mm	1.20	1.09	1.20	1.09	1.71	1.06

varied according:

$$I_{1,2} = 25\cos\left(\frac{\pi x}{\tau_p}\right) - 50\sin\left(\frac{\pi x}{\tau_p}\right), \tag{6.1}$$

$$I_{2,2} = 25\sin\left(\frac{\pi x}{\tau_p}\right) + 50\cos\left(\frac{\pi x}{\tau_p}\right). \tag{6.2}$$

Thereafter, the mean force values, F_{x_m} and F_{z_m} and the rms force ripples, F_{x_r} and F_{z_r}, have been calculated. The cost function, to be minimized, is the ratio between the rms force ripple and the mean force:

$$f_{cost}(mr, ct) = \frac{F_{x_r}}{|F_{x_m}|} + \frac{F_{z_r}}{|F_{z_m}|}. \tag{6.3}$$

The cost function, $f_{cost}(mr, ct)$, is optimized using the Matlab function `fmincon`, which can optimize non-linear constrained problems. The function uses a sequential quadratic programming method: a quasi-Newton line search.

6.2 Optimization results

The cost function has been minimized for the four different planar motor configurations. The results are independent of the height of the magnet array of the planar motor, since the same solution is found for the two planar motors with the Halbach arrays and for the motors with the N-S arrays. The optimal sizes for the planar motor with the Halbach array are: mr = 17.6 mm and ct = 15.1 mm. For the planar motor with the N-S magnet array, the optimal results are: mr = 10.2 mm and ct = 13.1 mm. Figure 6.2 and 6.3 shows a top view of both optimized planar motor configurations. For these configurations, the ratio between the rms force ripple and mean force is calculated. For this calculation, four coils are used and the magnet array is moved both in negative x-direction and negative y-direction over a half pole pitch. The equations 2.2, 2.3, 2.4, 2.5 with $I_x = I_y = I_z$ = 100A are used for the current in the coils. The force in y-direction is now also calculated. The results are summarized in Table 6.2.

The table shows that the planar motors with a Halbach array generate the double force compared to the N-S array. The force ripple is less than 2%. The force ripple of the machines with the N-S array is almost constant for all force components. The force ripple of the motor with the Halbach array is significantly less in the z-direction. The double height of the magnet array does not double the force. However, the force ripple decreases.

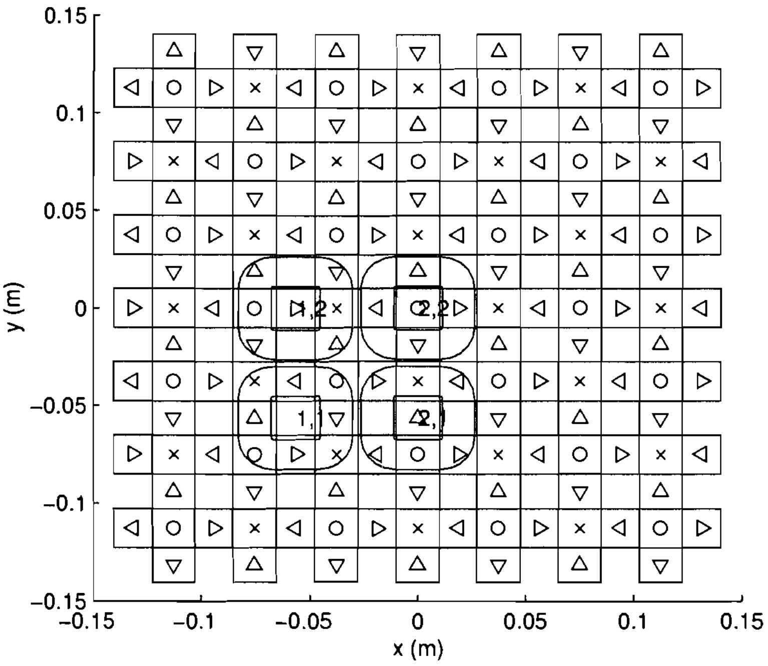


Figure 6.2. *Optimized planar motor with Halbach magnet array*

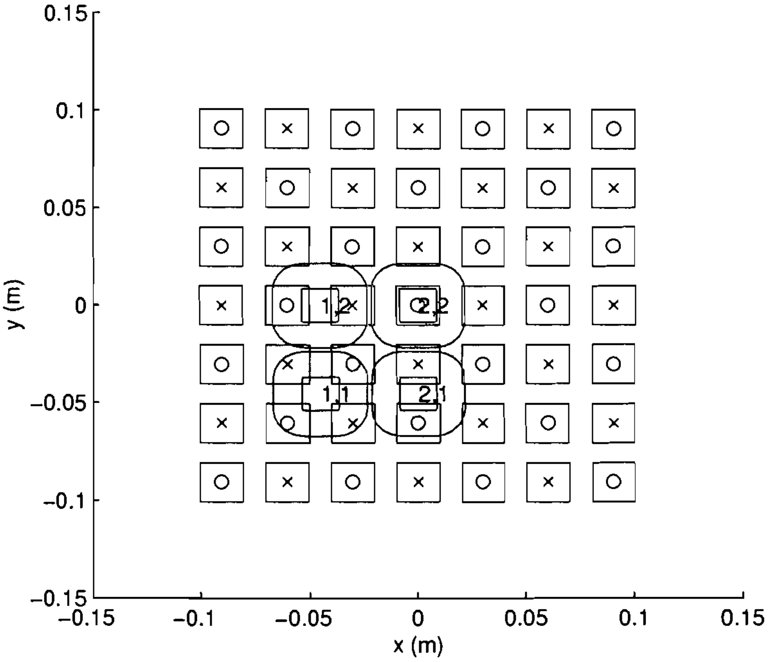


Figure 6.3. *Optimized planar motor with non-Halbach magnet array.*

6.3 Mass and acceleration

If the assumption is made that the found optimal values of mr and ct are the optimal values for all magnet heights, then the results of the previous section can be used to estimate the ideal mass for the moving platform for a given magnet height, in order that the acceleration in x- and y-directions is maximized. If the mass of the platform is known, an estimation can be made of the continuous dissipation of the motor, i.e. the copper losses during suspension of the platform. The following assumptions have been made for these calculations:

- the packing factor of the coils is 0.8,
- the magnet array is suspended 1 mm,
- one group of four coils lifts 25 % of the mass of the magnet array,
- the levitated magnet array consists of 8x8 in the z-direction magnetized magnets (size is essential to estimate the mass of the platform); the Halbach structure consists of 208 magnets, the N-S structure of 64 magnets,
- the mass density of the magnets is 7700 kg/m^3, the remanence is 1.3 T.

The force in x-, y-, and z-direction have been calculated using the Matlab tools for the two different magnet arrays as function of the magnet height with $I_x = I_y = I_z = 1A$ for one coil group. This way the force can be interpreted as a motor constant and the acceleration is the acceleration as acceleration per ampere.

When the magnet height is known, the mass of the different arrays, m_a can be estimated:

$$m_a(mh) = \alpha \cdot mh \tag{6.4}$$

where $\alpha = 197$ kg/m for the N-S array and $\alpha = 587$ kg/m for the Halbach array.

Using these numbers, the acceleration of the magnet array, a, can be determined in the different directions (x, y, z) as function of the mass of the platform. The mass of the platform consists of the magnet array mass, m_a, and the mass of the structure to which the magnets are attached, m_l. This way the magnet mass, which is related to the produced force, and the mass of the platform construction and its load are separated. The total mass of the moving platform, m_p is:

$$m_p = m_l + m_a. \tag{6.5}$$

The acceleration a can be calculated as function of mh and m_l is:

$$a_x = \frac{F_x(mh)}{m_p}, a_y = \frac{F_y(mh)}{m_p}, a_z = \frac{F_z(mh)}{m_p}. \tag{6.6}$$

Figure 6.4 and 6.5 show the acceleration per ampere for different values of m_l for the planar motor with the two magnet arrays. The figures show that for every value of m_l there is a value of mh, in order that the acceleration is maximized. These optimal values are used to calculate the copper losses in the planar motor, while the planar motor is suspended. The value of I_z for one coil group to levitate the mass platform is equal to, because the force is calculate for a current of 1 A:

$$I_z = \frac{m_p \cdot 9.81}{4F_z(mh)} \tag{6.7}$$

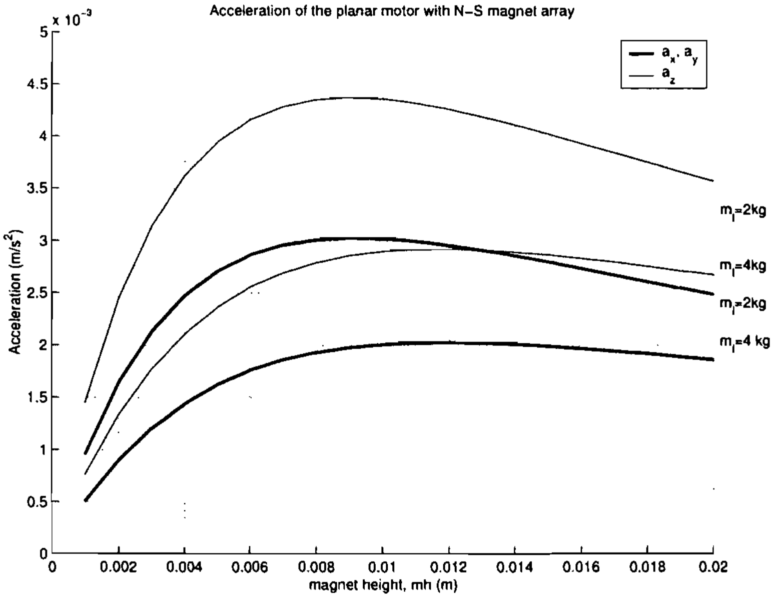


Figure 6.4. *Acceleration of the planar motor with the N-S magnet array with $I_x = I_y = I_z = 1A$ and different m_l.*

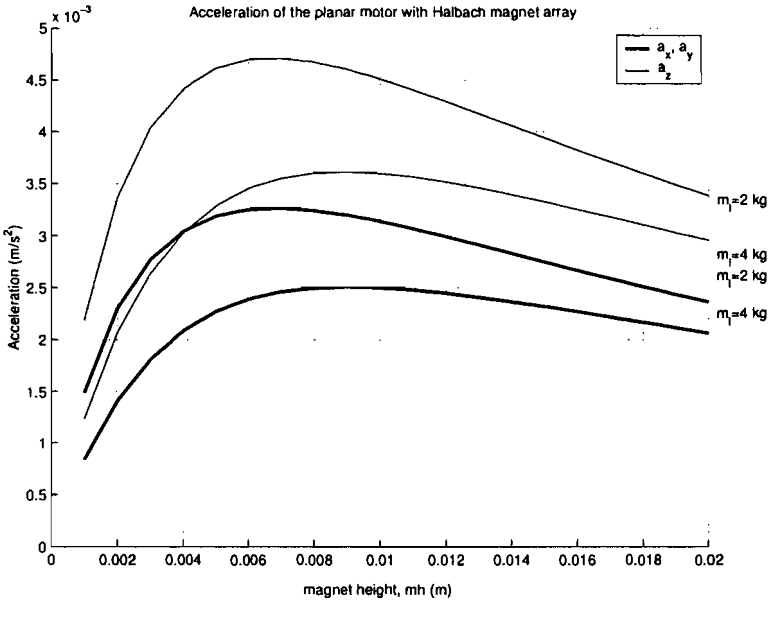


Figure 6.5. *Acceleration of the planar motor with the Halbach magnet array with $I_x = I_y = I_z = 1A$ and different m_l.*

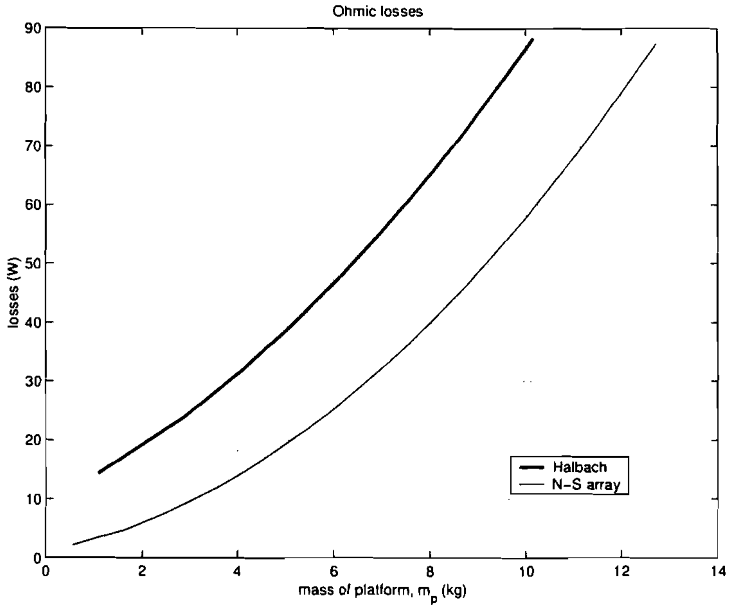


Figure 6.6. *Continuous power losses of two planar motors with a Halbach and a regular N-S magnet array as function of the total levitated mass.*

The ohmic loss in the coil group is: I_z^2R where R is the resistance of one coil, which is equal to 42.3 $\mu\Omega$ for the planar motor with the Halbach array and 39.1 $\mu\Omega$ for the planar motor with the N-S array. The total ohmic losses are, $4I_z^2R$, under the assumption that one coil group lifts a quarter of the mass of the platform.

The estimated continuous losses of the planar motor are shown in Figure 6.6 as function of the total weight of the levitated platform. The figure shows that the losses of the planar motor with the N-S magnet array are much smaller than the losses of the motor with the Halbach magnet array. This is caused by the large size of the rectangular magnets in the array, which increases the mass of the platform almost three times compared with the N-S structure, while the force only doubles. The total losses of the planar motor with the N-S magnet structure are comparable with the moving magnet motor of Kim [5].

6.4 Conclusions

The optimization of the planar motor shows that with a 2-phase winding a position independent force can be made with only a small ripple by controlling *mr* and *ct*. Due to the large size of the rectangular magnets in the optimized structure, force to the magnet mass ratio of the planar motor with the Halbach array is lower compared to that of the N-S structure. Therefore, the estimated copper losses are higher. The real power losses can only be estimated if the complete planar motor structure (with a least three coil groups to stabilize the platform) is known and the exact currents in each coil can be calculated. The optimization shows that the mass of the moving platform should be included in the cost function as it strongly influences the planar motor performance and losses.

Chapter 7

Measurement setup for the planar motor

A prototype of the planar motor will be built, after the design and optimization of the geometry. This prototype will be used to verify both the analytical and numerical models, which are used in the design phase. Thereafter, this prototype will be used to test and optimize the control algorithms. As the planar motor will only be set free at the end of all these experiments, a system should be developed, which supports the planar motor during all the experiments and which provides all electrical and mechanical quantities, such as back-EMF waveforms, inductances, position in 6-DOFs, speeds, forces and torques.

7.1 Measurement plan

The measurement and experiments with the planar motor can be distinguished into two parts. The first part is the verification of the used models and the design of the planar motor, the second part is the implementation and verification of the control algorithms.

7.1.1 Model verification

During the model verification, the properties of the prototype will be compared with the calculated ones. The prototype will be optimized for at least the following properties:

- mechanical properties; forces and torques with respect to the position of the platform with the magnets,
- electrical properties; back-EMF waveforms and the self and mutual inductances of the coils.

For the measurement of forces, torques and the influence of the moving platform on the inductances, the moving platform must be held in different positions, while currents are applied to the coils. For the measurement of the back-EMF waveforms the platform must be moved at a constant speed. The system which supports the planar motor during the experiments must both be able to move the magnet array to a specific location and keep it there during a measurement, and to move the magnet array with constant speed.

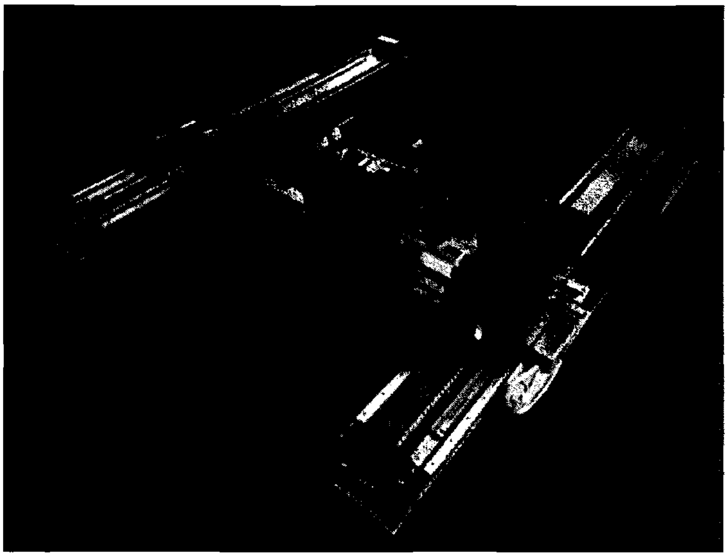


Figure 7.1. *H-bridge with three linear motors (picture: Assembléon)*

7.1.2 Implementation of the controllers

The controllers of the planar motor can be implemented after the verification of the design. To decrease the complexity and for reasons of safety, a controller for one degree of freedom will be implemented first. Later, the controllers for the other DOFs will be implemented sequentially. The support system must be able to move freely in the implemented degrees of freedom, while the other degrees of freedom are fixed. Furthermore, the support system must provide the controllers of the planar motor with sensor information, such as the position.

7.2 The support system

The demands for the support system, which are explained in the previous section, are different for both types of experiments. During the verification of the model, the support system is controlled actively. During the implementation of the planar motor controllers, only the degrees of freedom should be controlled by the support system, which are not controlled by the controllers of the planar motor. Furthermore, the support system should not load the planar motor unnaturally.

7.2.1 Gantry

The largest movements of the planar motor are in the x-and y-direction. Therefore, a gantry can be used as base of the support system. A gantry can move the magnet array over large distances in two degrees of freedom (x and y). Figure 7.1 shows a gantry. A gantry consists of two linear motors which move in y-direction. A third linear motor in the x-direction is mounted between the coil blocks of the two other motors. The platform with the magnet array is fixed to the coil block of the third motor. The coil array of the planar motor is placed on the baseplate of the gantry. The encoders of the linear motors can also provide the position information for the

x and y-axis during all measurements. Besides of movement in x and y-direction, the gantry can also provide a small rotation, ϕ, about the z-axis by displacing the linear motors on the y-axis with respect to each other.

The gantry loads the planar motor during the experiments with the controllers. The masses of the linear motors add to the mass of the platform, to which the magnet array is mounted. This mass is different for the different degrees of freedom. Also a damping will be introduced, which is absent for a planar motor moving in vacuum. The control algorithms of the different linear motors of the gantry should compensate both the mass and damping, caused by the gantry. This could be done by a force feedforward controller.

The design of a gantry, especially for a single project is very costly [22]. The company Assembleon, a manufacturer of pick-and-place machines, partly sponsors a gantry, with linear motors designed by Philips CFT, for the planar motor project.

7.2.2 Other degrees of freedom during model verification

Three degrees of freedom (x, y and a small angle ϕ) are controlled by the gantry. Implementation of the other three degrees of freedom and increasing the angle ϕ is costly because extra mass is added to the gantry and a high stiffness must be maintained. So the surplus value of every extra degree of freedom (z, ψ, θ, large ϕ) has to be considered:

z-axis The air-gap between coils and magnet array can be controlled by adding a z-axis. Adding this degree of freedom is recommended as the generated force strongly depends on the air gap height. There are two different implementations possible of the z-axis: a linear motor and a ball screw drive. Only a solution with the linear motor could be used during the implementation of the controllers, because the z-axis can be moved freely when the machine is not used. However, the thermal loading of this actuator is high because during the other experiments the linear motor must be energized continuously to stay on the same position. In case of power outage it could damage the planar motor when the magnet array is dropped onto the coil array. Therefore, a ball screw drive, is the better and cheaper option.

large angle ϕ The gantry can rotate the planar motor over a small angle ϕ. Addition of a motor to increase this angle is not useful, because from literature can be seen that a planar motor is designed for a specific angle ϕ between the coils and the magnets [8], [10].

angles ψ and θ The air gap between the coils and magnet array is small (of the order of millimeters). Therefore, the maximum angle ψ and θ, the rotations about the x-axis and y-axis, respectively, will be very small and the surplus value of adding (one of) these degrees of freedom is therefore questionable.

7.2.3 Other degrees of freedom during controller implementation

During the implementation of the controllers of the planar motor, the degrees of freedom, which will be controlled by the controllers of the planar motor must be variable. The gantry can support the magnet array in three degrees of freedom (x, y and ϕ) . The coupling between the gantry and the platform with the magnets

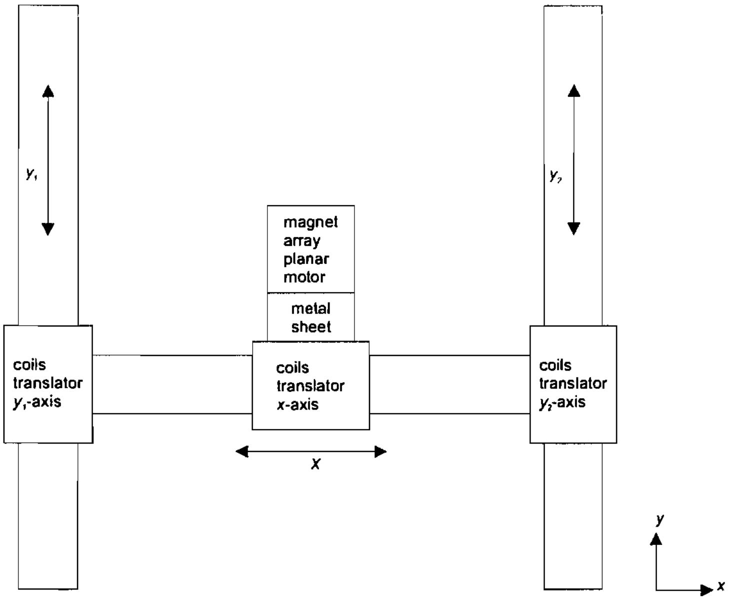

y_1
y_2
magnet array planar motor
metal sheet
coils translator y_1-axis
coils translator x-axis
coils translator y_2-axis
x
y
x

Figure 7.2. *Top view of the gantry with the magnet array and the sheet coupling*

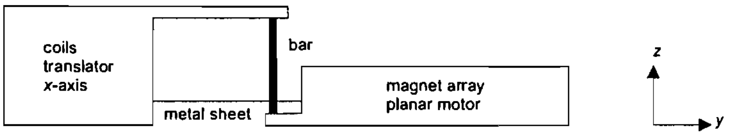


Figure 7.3. *Side view of the coupling with blocked z-axis movement*

has to provide the other degrees of freedom but every degree of freedom should be lockable.

During the development of a planar motor within Philips CFT [9], a mechanical coupling has been used [21]. This coupling is also usable for this planar motor project. The magnet array is coupled with the gantry by a metal sheet. This sheet coupling is stiff in both x and y-direction but not in the other directions. Figure 7.2 shows the top view of the gantry with the magnet array coupled by a metal sheet. The magnet array can move in x and y direction and can rotate slightly around the z-axis if there is a position difference between the y_1- and the y_2-leg of the gantry. The other degrees of freedom can be locked by placing metal bars between the magnet array and the gantry. Figure 7.3, figure 7.4 and figure 7.5 show how the z-axis, the rotation angles ψ and θ can be controlled, respectively. The figures show the principles. They are not drawn to scale.

7.3 Sensors

During the experiments sensors measure the position, forces and torques. The different types of sensors and their application are described in this section.

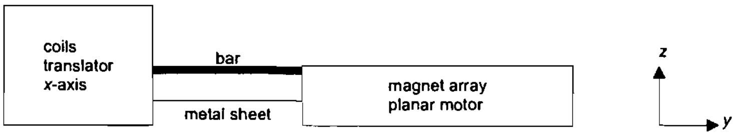


Figure 7.4. *Side view of the gantry with blocked rotation around the x-axis*

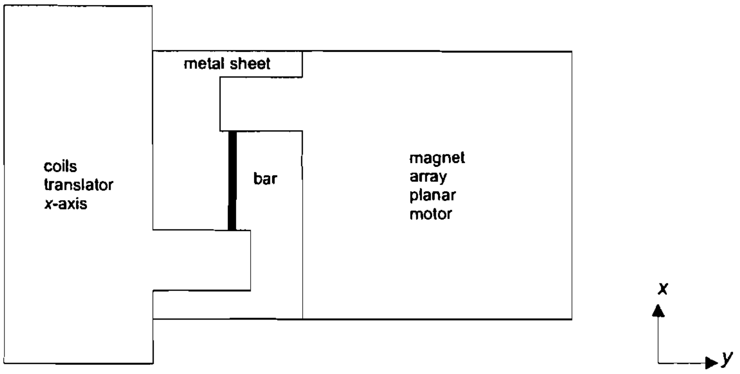


Figure 7.5. *Top view of the coupling with blocked rotation around the y-axis*

7.3.1 Force sensors

Forces and torques on the planar motor will be measured during the verification of the design. Forces and torques can be measured with force and torque transducers. On the market there are transducers available, which can measure three force components and one torque component (Kistler) and there are systems available which can measure both three force and three torque components (ATI and AMTI). The transducers differ in mass, stiffness and accuracy.

There are two different options to place the sensor. It can be between the baseplate of the gantry and the coil array or between the platform with the magnet array and the gantry. But if the sensor is placed under the coil array, a heavier and, therefore, less accurate sensor should be used than if the sensor is placed between the platform with magnets and the gantry.

The force and torque measurement can be carried out in two ways. One sensor could be used (ATI and AMTI) or three sensors, placed in a triangle, could be used to measure all forces and torques (Kistler). Another advantage of the use of more sensors is that less heavy sensors could be applied, which are more accurate.

7.3.2 Position sensors

The measurement of the position of the platform with magnets of the planar motor is partly done by the encoders of the linear motors of the gantry. They provide the x- and y-positions and the angle ϕ. The z-position and the other angles, ψ and θ, have to be measured by capacitive sensors mounted on the platform with the magnet array. By measuring the distance from the magnet array to the coil array in three points the orientation of the magnet array can be calculated.

7.4 Conclusion

The use of a gantry for the support of the planar motor during all experiments is applicable. However, the usefulness of the system depends on the implemented controllers for the gantry during the different experiments, the quality of the coupling between the planar motor and the gantry and the used sensors. A definitive specification of the planar motor is necessary before the sheet coupling can be designed and the sensors selected.

Chapter 8

Conclusions and recommendations

8.1 Conclusions

- The research into planar motors with six degrees of freedom has rapidly developed in the last years. However, only limited literature resources can be found on planar motors with moving magnets, and the performance of the structures is unclear. Therefore, the research into planar motors with moving magnets is indeed strategic.

- Fast and time saving numerical tools for the calculation of static currents, forces, torques and inductances in planar motors with ideal material properties have been implemented in Matlab. These tools are based on the analytical solution of magnetic fields and vector potentials. The results obtained by these tools are comparable to three-dimensional magneto static finite element simulations. Besides, these tools can be further used for optimizations and can be easily modified for transient simulations.

- The magneto static solvers of two three-dimensional finite element packages have been reviewed for the planar motor project: FLUX3D and Maxwell 3D. FLUX3D is the most suitable package for the project, because of the complete parametrization of the program, the larger maximum number of elements and the possibility for transient simulations, associated with motion. Both packages cannot be used for thermal simulations in vacuum.

- The analytical tools are used to optimize two geometric variables of the planar motor with a Halbach magnet array and a regular N-S magnet array using a single objective function: the minimization of the rms-force ripples. The optimization method used, is an quasi-Newton line search algorithm implemented in Matlab. Both structures produce a position independent force with a rms-force ripple of less than 2 %.

- The ohmic losses of the the optimized planar motor structures have been estimated as function of the mass of the moving platform during levitation. The motor with the optimized N-S magnet array proved to have less ohmic losses due to the better force to pure magnet mass ratio than the motor with the optimized Halbach array.

- A H-drive can be used both for supporting a planar motor during experiments, which aim to verify the results obtained by used models, and during implementation and testing of controllers for the planar motor.

8.2 Recommendations

- Verification of the developed tools by experiments.
- Modification of the developed tools for transient simulations and developing tools for thermal and mechanical simulations.
- Improvement of the calculation time of these tools in order that more variables can be optimized at the same time.
- Development and verification of multi-objective optimization algorithms for multi-variable optimization of the planar motor.
- Synthesis and design of a full planar motor structure, which can support itself and is controllable in six degrees of freedom, using the developed tools, and design and synthesis of the power amplifiers. These designs should not infringe any patents.
- Final design of the measurement setup and selection of the sensors.

Bibliography

[1] Sawyer, B.A., 'Magnetic positioning device,' Apr. 2 1968, US Patent 3,376,578.

[2] Fujii, N. & M. Fujitake, 'Two-dimensional drive characteristics by circular-shaped motor,' *IEEE Trans. Ind. Applicat.*, vol. 35, no 4, July/Aug 1999, pp. 803-809.

[3] Earnshaw, S., 'On the nature of the molecular forces which regulate the constitution of the luminiferous ether,' *Trans. Camb. Phil, Soc.*, VII part I, 1842, pp. 97-112.

[4] Auer, F., '*Combined electromagnetic suspension and propulsion for positioning with sub-micrometer accuracy,*' doctoral dissertation, Delft University of Technology, Delft, The Netherlands, 1995.

[5] Kim, W.J., '*High-precision planar magnetic levitation,*' doctoral dissertation, Massachusetts Institute of Technology, Cambridge, USA, 1997.

[6] Zhu, Z.Q. & D. Howe, 'Halbach permanent magnet machines and applications: a review,' *IEE Proceedings on Electric Power Applications*, Vol 148, Iss. 4, Jul 2001, pp. 299-308.

[7] Molenaar, L, '*A novel planar magnetic bearing and motor cnfiguration applied in a positioning stage,*' doctoral dissertation, Technische Universiteit Delft, Delft, The Netherlands, 2000.

[8] Compter, J.C. & P.C.M. Frissen, '*Displacement device*', Mar. 15, 2001, Patent WO 01/18944.

[9] Compter, J.C., 'Electro-dynamic planar motor,' not published.

[10] Hazelton, A.J. & M.B. Binnard, J.M. Gery, '*Electric motors and positioning devices having moving magnet arrays and six degrees of freedom,*' Mar. 27 2001, US Patent 6,208,045.

[11] Jung, K.S. & Y.S. Baek, 'Study on a novel contact-free planar system using direct drive DC coils and permanent magnet,' *IEEE/ASME Trans. Mechatron.*, vol. 7, no. 2, March 2002, pp. 35-43.

[12] Binnard, M., '*Planar motor with linear coil arrays,*' Sep. 3. 2002, US Patent 6,445,093.

[13] Cho, H.S. & C.H. Im, H.K. Jung, 'Magnetic field analysis of 2-D permanent magnet array for planar motor,' *IEEE Trans. Magn.*, vol. 37, no. 5, 2001, pp. 3762-3766.

[14] Kang, G.H. & J.P. Hong, G.T. Kim, 'A novel design of an air-core type permanent magnet linear brushless motor by space harmonics field analysis,' *IEEE Trans. Magn.*, vol. 37. no. 5, 2001, pp. 3732-3736.

[15] Yonnet, G.P. & G. Akoun, '3D analytical calculation of the forces exerted between two cuboidal magnets,' *IEEE Trans. Magn.*, vol 20, no 5, 1984, pp. 1962-1964.

[16] Reece, A.B.J. & T.W. Preston, '*Finite elemtent methods in electrical engineering*', Oxford University Press Inc., New York, USA, 2000.

[17] McKerrow, P.J., '*Introduction to robotics*,' Addison-Wesley Publishers Ltd., Sydney, Australia, 1993.

[18] Abramowitz, M. & I.A. Stegun, '*Handbook of mathematical functions with formulas, graphs, and mathematical tables*,' Dover publications, New York, USA, 1965, pp. 891-4.

[19] Binns, K.J. & P.J. Lawrenson, C.W. Towbridge, '*The analytical and numerical solutions of electrical and magnetic fields*,' John Wiley & Sons Ltd., Chichester, England, 1994.

[20] '*FLUX3D version 3.30 user's guide*', Cedrat, Meylan, Fance, Jan 2002.

[21] Soemers, HMJR, '*Coupling of a planar motor to a xy-positioning stage*, private conversation,' Philips CFT, Eindhoven, The Netherlands, 31-01-2003.

[22] Verstraaten, T, '*Selection of a gantry as part of a measurement system for a planar motor*,' private conversation, Anorad Europe B.V., Best, The Netherlands, 08-01-2003.

Appendix A

Matlab code for the calculations of forces on a magnet array and 4 coils

First the magnets and coils are defined. Then, the force is calculated for different positions of the magnets. The positions are defined by Carpet. The currents in the coils are a function of the magnet position.
Author: J.W. Jansen.

```
function sol=main_sol(x)
global optdata
optdata=x;

MagnetSpecification=struct('ArraySize', [6 6], 'specs', magnetspecs,
'Halbach', optdata(4),'ironlayer', -1);
MagnetArray=magnet_array(MagnetSpecification,1);
CoilSpecification=coilspecification;
CoilArray=coil_array(CoilSpecification,1);
tau=20e-3+optdata(1);

clear sol;
ny=0;
for y=0:tau/20:tau/2
    ny=ny+1;
    disp(ny);
    nx=0;
    for x=0:tau/20:tau/2
        nx=nx+1;
        MagnetPosition=[-x,-y,1e-3,0,0,0];
        Carpet=struct('MassCenter', MagnetPosition + MagnetMassCenter,
           'BFieldCalc', MagnetPosition +
           [0 0 MagnetSpecification.specs(3)/2 0 0 0], 'Mass', 1,
           'Inertia', [1 0 0; 0 1 0; 0 0 1], 'SpeedBody', [0 0 0],
           'SpeedObserver', [0 0 0 ], 'Rotation',[0 0 0],
           'vec_MC_Bfield', -[MagnetMassCenter(1) MagnetMassCenter(2)
           MagnetMassCenter(3)-MagnetSpecification.specs(3)/2]);
        Ix=100;
        Iy=100;
        Iz=100;
        CoilArray(2,2).Current=+Ix*cos(pi*x/tau)*sin(pi*y/tau)
          +Iy*sin(pi*x/tau)*cos(pi*y/tau)-Iz*sin(pi*x/tau)*sin(pi*y/tau);
        CoilArray(2,1).Current=+Ix*cos(pi*x/tau)*cos(pi*y/tau)
          -Iy*sin(pi*x/tau)*sin(pi*y/tau)-Iz*sin(pi*x/tau)*cos(pi*y/tau);
        CoilArray(1,2).Current=-Ix*sin(pi*x/tau)*sin(pi*y/tau)
          +Iy*cos(pi*x/tau)*cos(pi*y/tau)-Iz*cos(pi*x/tau)*sin(pi*y/tau);
        CoilArray(1,1).Current=-Ix*sin(pi*x/tau)*cos(pi*y/tau)
          -Iy*cos(pi*x/tau)*sin(pi*y/tau)-Iz*cos(pi*x/tau)*cos(pi*y/tau);
        [F, void]=force(CoilArray, MagnetArray, Carpet);
        sol(ny, nx).pos=[x,y];
        sol(ny, nx).Force=F;
    end
end
```

Appendix B

Matlab code for the definition of the magnet array

Matlab code to define a magnet array with the size ArraySize, and the magnet specification, magnetspecs, with a Halbach or N-S structure and with a possible ironlayer at a certain location.
Author: J.W. Jansen

```
MagnetSpecification=struct('ArraySize', [6 3], 'specs', magnetspecs,
   'Halbach', 1,'ironlayer', -1);
MagnetArray=magnet_array(MagnetSpecification, 0);}
```

B.1 Function magnetspecs

```
function spec=magnetspecs
global optdata
x=optdata;
%Three types of magnets:
%  -----
% |     |
% |     |
% |_____| square magnets
%
%  -----
% |_____| horizontal magnets
%
%  _
% | |
% | |
% |_|     vertical magnets
%

%square magnets
as=20e-3;   %size in x-direction
bs=20e-3;   %size in y-direction
cs=10e-3;   %size in z-direction
Brs=1.3;    %Remenace
```

```
%horizontal oriented magnets
bh=x(1);    %size in y-direction
ch=x(2);    %size in z-direction
Brh=1.3;


%Because of symmetry
ah=as;
av=bh;
bv=bs;
cv=ch;
Brv=Brh;

spec=[as, bs, cs, Brs, ah, bh, ch, Brh, av, bv, cv, Brv, 0];
```

B.2 Function magnet_array

```
function magnet=magnet_array(data, plt)

%Extra magnets in iron layer
if data.ironlayer==-1
    [magnet ydirl]=build_magnet_array(data);
else
    [magnet1 ydirl]=build_magnet_array(data);
    data.ironlayer=-1;
    [magnet2 ydirl]=build_magnet_array(data);
    magnet=append(magnet2, magnet1);
end


%plots magnet array
if plt==1
    figure;
    subplot('position',[0.25 0.35 0.5 0.6])
    hold on;
    s=length(magnet);
    for n=1:s
        if  magnet(n).Magnetisation(1)==6
            plot(magnet(n).Position(1), magnet(n).Position(2), 'x');
        end
        if  magnet(n).Magnetisation(1)==5
            plot(magnet(n).Position(1), magnet(n).Position(2), 'o');
        end
        if  magnet(n).Magnetisation(1)==1
            plot(magnet(n).Position(1), magnet(n).Position(2), '>');
        end
        if  magnet(n).Magnetisation(1)==2
            plot(magnet(n).Position(1), magnet(n).Position(2), '<');
        end
        if  magnet(n).Magnetisation(1)==3
            plot(magnet(n).Position(1), magnet(n).Position(2), '^');
        end
        if  magnet(n).Magnetisation(1)==4
            plot(magnet(n).Position(1), magnet(n).Position(2), 'v');
        end
        plot(magnet(n).Position(1)+magnet(n).Dimension(1)*[-1 1],
           (magnet(n).Position(2)+magnet(n).Dimension(2))*[1 1]);
        plot(magnet(n).Position(1)+magnet(n).Dimension(1)*[-1 1],
           (magnet(n).Position(2)-magnet(n).Dimension(2))*[1 1]);
        plot((magnet(n).Position(1)+magnet(n).Dimension(1))*[1 1],
           magnet(n).Position(2)+magnet(n).Dimension(2)*[-1 1]);
        plot((magnet(n).Position(1)-magnet(n).Dimension(1))*[1 1],
           magnet(n).Position(2)-magnet(n).Dimension(2)*[-1 1]);
    end
    %square window for a nice picture
    v=axis;
    %v=v-[x x y y];
    m=max(abs(v));
    %axis([-m+x m+x -m+y m+y]);
    axis([-m m -m m]);
%    axis([-0.15 0.15 -0.15 0.15])
    hold off;
    subplot('position',[0.25 0.065 0.5 0.12])
    hold on
%    axis([-0.15 0.15 -0.02 0.02]);
    for n=1:s
        if magnet(n).Magnetisation(1)==5|magnet(n).Magnetisation(1)==6
             |magnet(n).Magnetisation(1)==1|magnet(n).Magnetisation(1)==2
            plot(magnet(n).Position(1)+magnet(n).Dimension(1)*[-1 1],
               (magnet(n).Position(3)+magnet(n).Dimension(3))*[1 1]);
            plot(magnet(n).Position(1)+magnet(n).Dimension(1)*[-1 1],
               (magnet(n).Position(3)-magnet(n).Dimension(3))*[1 1]);
            plot((magnet(n).Position(1)+magnet(n).Dimension(1))*[1 1],
               magnet(n).Position(3)+magnet(n).Dimension(3)*[-1 1]);
            plot((magnet(n).Position(1)-magnet(n).Dimension(1))*[1 1],
               magnet(n).Position(3)-magnet(n).Dimension(3)*[-1 1]);
        end
        if magnet(n).Position(2)==min(ydirl)
            if  magnet(n).Magnetisation(1)==6
                plot(magnet(n).Position(1), magnet(n).Position(3), 'v');
            end
            if  magnet(n).Magnetisation(1)==5
                plot(magnet(n).Position(1), magnet(n).Position(3), '^');
            end
            if  magnet(n).Magnetisation(1)==1
                plot(magnet(n).Position(1), magnet(n).Position(3), '>');
            end
            if  magnet(n).Magnetisation(1)==2
                plot(magnet(n).Position(1), magnet(n).Position(3), '<');
                end
            end

        end
end
```

B.3 Function build_magnet_array

```
%function constructs a halbach array of a given size and offset.
%Function plots the array afterwards.
%
%function: array=halbach_array(s, x,y,z, ao, fm)
%
%s is matrix with [size in x-direction, size in y-direction]
%x,y,z is the center of the aray in global coordinates
%ao dictates the orientation of the array. field concentration on upper
%or lower side fm dictates the magnetization of outmost square magnet
%in the third quadrant
function [array, ydirl]=build_magnet_array(MagnetData)
clear array;

as=MagnetData.specs(1)/2;
bs=MagnetData.specs(2)/2;
cs=MagnetData.specs(3)/2;
Brs=MagnetData.specs(4);
ah=MagnetData.specs(5)/2;
bh=MagnetData.specs(6)/2;
ch=MagnetData.specs(7)/2;
Brh=MagnetData.specs(8);
av=MagnetData.specs(9)/2;
bv=MagnetData.specs(10)/2;
cv=MagnetData.specs(11)/2;
Brv=MagnetData.specs(12);


s=MagnetData.ArraySize;

if MagnetData.ironlayer==-1
    ao=1; %down
    z=0;
    height_hv=MagnetData.specs(13);
else
    ao=0; %up
    z=2*(MagnetData.ironlayer-cs);
    height_hv=2*cs-2*ch-MagnetData.specs(13);
end

fm=0;
```

```
%calculates x-coordinates of the single magnets for the three types
%(square, vertical and horizontal)
if mod(s(1), 2)==0
    xdirl=[-(s(1)/2-1)*(2*as+2*av)-(av+as):2*as+2*av:
       (s(1)/2-1)*(2*as+2*av)+(av+as)];
else
    xdirl=[-(s(1)-1)/2*(2*as+2*av):2*as+2*av:(s(1)-1)/2*(2*as+2*av)];
end
if mod(s(1), 2)==0
    xdirh=[-(s(1)/2-1)*(2*as+2*av)-(av+as):2*as+2*av:
       (s(1)/2-1)*(2*as+2*av)+(av+as)];
else
    xdirh=[-(s(1)-1)/2*(2*as+2*av):2*as+2*av:(s(1)-1)/2*(2*as+2*av)];
end
if mod(s(1), 2)==0
    xdirv=[-(s(1)/2)*(2*as+2*av):2*as+2*av:(s(1)/2)*(2*as+2*av)];
else
    xdirv=[-(s(1)-1)/2*(2*as+2*av)-(av+as):2*as+2*av:
       (s(1)-1)/2*(2*as+2*av)+(av+as)];
end

%calculates y-coordinates of the single magnets for the three types
%(square, vertical and horizontal)
if mod(s(2), 2)==0
    ydirl=[-(s(2)/2-1)*(2*bs+2*bh)-(bh+bs):2*bs+2*bh:
       (s(2)/2-1)*(2*bs+2*bh)+(bh+bs)];
else
    ydirl=[-(s(2)-1)/2*(2*bs+2*bh):2*bs+2*bh:(s(2)-1)/2*(2*bs+2*bh)];
end
if mod(s(2), 2)==0
    ydirh=[-(s(2)/2)*(2*bs+2*bh):2*bs+2*bh:(s(2)/2)*(2*bs+2*bh)];
else
    ydirh=[-(s(2)-1)/2*(2*bs+2*bh)-(bh+bs):2*bs+2*bh:
       (s(2)-1)/2*(2*bs+2*bh)+(bh+bs)];
end
if mod(s(2), 2)==0
    ydirv=[-(s(2)/2-1)*(2*bs+2*bh)-(bh+bs):2*bs+2*bh:
       (s(2)/2-1)*(2*bs+2*bh)+(bh+bs)];
else
    ydirv=[-(s(2)-1)/2*(2*bs+2*bh):2*bs+2*bh:(s(2)-1)/2*(2*bs+2*bh)];
end


%Combines x and y-coordinates for different magnet sizes.
%large magnets;
%x, y, z (origing magnet), as, bs, cs (size in x, y and z direction),
%magnitization (1-6 for x-axis positive, x negative, yp, yn, zp, zn), Br
row=1;
for n1=1:length(xdirl)
```

```
    fms=fm;
    for n2=1:length(ydirl)
        array(row).Position=[xdirl(n1) ydirl(n2) z];
        array(row).Dimension=[as bs cs];
        array(row).Magnetisation=[mod(fm,2)+5 Brs];
        row=row+1;
        fm=fm+1;
    end
    fm=fms+1;
end


if MagnetData.Halbach==1
%horizontal magnets
fm=ao;
for n1=1:length(xdirh)
    fms=fm;
    for n2=1:length(ydirh)
        array(row).Position=[xdirh(n1) ydirh(n2) z+ch-cs+height_hv];
        array(row).Dimension=[ah bh ch];
        array(row).Magnetisation=[mod(fm,2)+3 Brh];
        row=row+1;
        fm=fm+1;
    end
    fm=fms+1;
end

%vertical magnets
fm=ao;
for n1=1:length(xdirv)
    fms=fm;
    for n2=1:length(ydirv)
        array(row).Position=[xdirv(n1) ydirv(n2) z+cv-cs+height_hv];
        array(row).Dimension=[av bv cv];
        array(row).Magnetisation=[mod(fm,2)+1 Brv];
        row=row+1;
        fm=fm+1;
    end
    fm=fms+1;
end
end
```

B.4 Function append

```
%Appends two matrices with structs
function out=append(in1, in2)
s1=size(in1);
s2=size(in2);
if s1(1)~=s2(1)
```

```
    error('Matrices do not have the same number of columns');
end
for n=1:s1(2);
    out(:,n)=in1(:,n);
end
for n=1:s2(2)
    out(:,n+s1(2))=in2(:,n);
end
```

Appendix C

Matlab code for the definition of the coil array

Matlab code to define a coil array with specifications coilspecification.
Author: J.W. Jansen

```
CoilSpecification=coilspecification;
CoilArray=coil_array(CoilSpecification,1);
```

C.1 Function coilspecfication

```
function spec=coilspecification()

global optdata
x=optdata;
spec(1).name='layer1';
spec(1).ArraySize=[2 2];
spec(1).ArrayPosition=[-0.75*(20e-3+x(1)) -0.75*(20e-3+x(1)) -2.5e-3];
   %x, y,z
spec(1).Spacing=[10e-3+x(1)/2-x(2) 10e-3+x(1)/2-x(2)];
spec(1).CoilSize=[20e-3+x(1)+x(2) 20e-3+x(1)+x(2) 5e-3 x(2) x(2)+1e-3];
   %l, b, h, th, fr
spec(1).Turns=[1];
```

C.2 Function coil_array

```
function array=coil_array(data, plt);

for n=1:length(data)
    s(n,1)=data(n).ArraySize(1);
    s(n,2)=data(n).ArraySize(2);
end
xdim=max(s(:,1));
ydim=max(s(:,2));

for n=1:length(data)
    array(:,:,n)=build_coil_array(data(n), xdim, ydim);
```

```
end


%plots coil array
if plt==1
 s=size(array);
 if length(s)==2;
     s(3)=1;
 end
 for nz=1:s(3)
  %figure;
   title(sprintf('Coil Layer %d', nz));
   hold on;
   for nx=1:s(1)
    for ny=1:s(2)
      if array(nx, ny, nz).Name ~= -1
       l=array(nx, ny, nz).Size(1);
       w=array(nx, ny, nz).Size(2);
       th=array(nx, ny, nz).Size(4);
       fr=array(nx, ny, nz).Size(5);
       ox=array(nx, ny, nz).Position(1);
       oy=array(nx, ny, nz).Position(2);
       line([ox-0.5*l+fr ox+0.5*l-fr], [oy-0.5*w oy-0.5*w]);
       line([ox+0.5*l ox+0.5*l], [oy-0.5*w+fr oy+0.5*w-fr]);
       line([ox+0.5*l-fr ox-0.5*l+fr], [oy+0.5*w oy+0.5*w]);
       line([ox-0.5*l ox-0.5*l], [oy+0.5*w-fr oy-0.5*w+fr]);
       line([ox-0.5*l+fr ox+0.5*l-fr], [oy-0.5*w+th oy-0.5*w+th]);
       line([ox+0.5*l-th ox+0.5*l-th], [oy-0.5*w+fr oy+0.5*w-fr]);
       line([ox+0.5*l-fr ox-0.5*l+fr], [oy+0.5*w-th oy+0.5*w-th]);
       line([ox-0.5*l+th ox-0.5*l+th], [oy+0.5*w-fr oy-0.5*w+fr]);

       text(ox, oy, array(nx, ny, nz).Name);
       segments=5;
       for n=1:segments
        dphi=0.5*pi/segments;
        line([ox+0.5*l-fr + fr*sin(dphi*(n-1)) ox+0.5*l-fr + fr*sin(dphi*n)],
           [oy-0.5*w+fr - fr*cos(dphi*(n-1)) oy-0.5*w+fr - fr*cos(dphi*n)]);
        line([ox+0.5*l-fr + fr*cos(dphi*(n-1)) ox+0.5*l-fr + fr*cos(dphi*n)],
           [oy+0.5*w-fr + fr*sin(dphi*(n-1)) oy+0.5*w-fr + fr*sin(dphi*n)]);
        line([ox-0.5*l+fr - fr*sin(dphi*(n-1)) ox-0.5*l+fr - fr*sin(dphi*n)],
           [oy+0.5*w-fr + fr*cos(dphi*(n-1)) oy+0.5*w-fr + fr*cos(dphi*n)]);
        line([ox-0.5*l+fr - fr*cos(dphi*(n-1)) ox-0.5*l+fr - fr*cos(dphi*n)],
           [oy-0.5*w+fr - fr*sin(dphi*(n-1)) oy-0.5*w+fr - fr*sin(dphi*n)]);
        line([ox+0.5*l-fr + (fr-th)*sin(dphi*(n-1)) ox+0.5*l-fr +
           (fr-th)*sin(dphi*n)], [oy-0.5*w+fr - (fr-th)*cos(dphi*(n-1))
           oy-0.5*w+fr - (fr-th)*cos(dphi*n)]);
        line([ox+0.5*l-fr + (fr-th)*cos(dphi*(n-1)) ox+0.5*l-fr +
           (fr-th)*cos(dphi*n)], [oy+0.5*w-fr + (fr-th)*sin(dphi*(n-1))
           oy+0.5*w-fr + (fr-th)*sin(dphi*n)]);
        line([ox-0.5*l+fr - (fr-th)*sin(dphi*(n-1)) ox-0.5*l+fr -
           (fr-th)*sin(dphi*n)], [oy+0.5*w-fr + (fr-th)*cos(dphi*(n-1))
           oy+0.5*w-fr + (fr-th)*cos(dphi*n)]);
        line([ox-0.5*l+fr - (fr-th)*cos(dphi*(n-1)) ox-0.5*l+fr -
           (fr-th)*cos(dphi*n)], [oy-0.5*w+fr - (fr-th)*sin(dphi*(n-1))
           oy-0.5*w+fr - (fr-th)*sin(dphi*n)]);
        end
       end
      end
     end
    end
    figure
    hold on;
    for nz=1:s(3)
     for nx=1:s(1)
      ny=1;
      if array(nx, ny, nz).Name ~= -1
       ox=array(nx, ny, nz).Position(1);
       oz=array(nx, ny, nz).Position(3);
       l=array(nx, ny, nz).Size(1);
       h=array(nx, ny, nz).Size(3);
       fr=array(nx, ny, nz).Size(5);
       line((ox-0.5*w)*[1 1],oz+0.5*h*[-1 1]);
       line((ox-0.5*w+fr)*[1 1],oz+0.5*h*[-1 1]);
       line((ox+0.5*w-fr)*[1 1],oz+0.5*h*[-1 1]);
       line((ox+0.5*w)*[1 1],oz+0.5*h*[-1 1]);
       line([ox-0.5*w ox-0.5*w+fr],oz+0.5*h*[1 1]);
       line([ox-0.5*w+fr ox+0.5*w-fr],oz+0.5*h*[1 1]);
       line([ox+0.5*w-fr ox+0.5*w],oz+0.5*h*[1 1]);
       line([ox-0.5*w ox-0.5*w+fr],oz+0.5*h*-[1 1]);
       line([ox-0.5*w+fr ox+0.5*w-fr],oz+0.5*h*-[1 1]);
       line([ox+0.5*w-fr ox+0.5*w],oz+0.5*h*-[1 1]);
     end
  end
 end
end
```

C.3 Function build_coil_array

```
function array=build_coil_array(data, xdim, ydim)


s=data.ArraySize;
l=data.CoilSize(1);
w=data.CoilSize(2);
sx=data.Spacing(1);
sy=data.Spacing(2);

if mod(s(1), 2)==0
```

```
    xdir=[-(s(1)-1)*(l+sx)/2:l+sx:(s(1)-1)*(l+sx)/2];
else
    xdir=[-(s(1)-1)/2*(l+sx):l+sx:(s(1)-1)/2*(l+sx)];
end


if mod(s(2), 2)==0
    ydir=[-(s(2)-1)*(w+sy)/2:w+sy:(s(2)-1)*(w+sy)/2];
else
    ydir=[-(s(2)-1)/2*(w+sy):w+sy:(s(2)-1)/2*(w+sy)];
end

xdir=xdir+data.ArrayPosition(1);
ydir=ydir+data.ArrayPosition(2);
zdir=data.ArrayPosition(3);

for nx=1:xdim
    for ny=1:ydim
        if nx > s(1) | ny > s(2)
            array(nx, ny).Name=-1;
            array(nx, ny).Position=[0 0 0];
            array(nx, ny).Size=[0 0 0 0 0];
            array(nx, ny).Turns=0;
            array(nx, ny).Current=0;
        else
            array(nx, ny).Name=sprintf('%d,%d', nx, ny);
            array(nx, ny).Position=[xdir(nx) ydir(ny) zdir];
            array(nx, ny).Size=data.CoilSize;
            array(nx, ny).Turns=data.Turns;
            array(nx, ny).Current=0;
        end
    end
end
```

Appendix D

Matlab code for the force and torque calculation

Author: J.W. Jansen

D.1 Function force

```
function [ForceTorque, Carpet]=force(CoilArray, MagnetArray, Carpet);

s=size(CoilArray);
if length(s)==2;
    s(3)=1;
end
ForceTorque=[0 0 0 0 0 0];
for nz=1:s(3)
    for ny=1:s(2)
        for nx=1:s(1)
            if CoilArray(nx, ny, nz).Name~=-1
                ForceTorque=ForceTorque+forcecoil2(CoilArray(nx, ny, nz),
                   MagnetArray, Carpet);
            end
        end
    end
end

%code for transient simulations using Matlab Aerospace Blockset code,
%extracted with real-time workshop!!!!!
%Ts=0.0001;
%Force(1)=ForceTorque(1);
%Force(2)=ForceTorque(2);
%Force(3)=ForceTorque(3);
%Torque(1)=ForceTorque(4);
%Torque(2)=ForceTorque(5);
%Torque(3)=ForceTorque(6);

%vbo=Carpet.SpeedBody;
%roto=Carpet.Rotation;
```

```
%Eulero=[Carpet.MassCenter(4), Carpet.MassCenter(5), Carpet.MassCenter(6)];
%veo=Carpet.SpeedObserver;
%xeo=[Carpet.MassCenter(1), Carpet.MassCenter(2), Carpet.MassCenter(3)];

%vb=vbo+Ts*(cross(vbo, roto)+Force/Carpet.Mass);
%rot=roto'+Ts*inv(Carpet.Inertia)*(Torque'-cross(roto',
     Carpet.Inertia*roto'));

%Euler=Eulero'+Ts*DOFsub_dot(roto, Eulero) ;

%ax=DOFsub_AXIS(Eulero);
%ve=vbo*ax;
%xe=xeo+Ts*ve;

%Carpet.MassCenter=[xe(1) xe(2) xe(3) Euler(1) Euler(2) Euler(3)];
%Carpet.BFieldCalc=xe+Carpet.vec_MC_Bfield;
%Carpet.BfieldCalc(4)=Euler(1);
%Carpet.BfieldCalc(5)=Euler(2);
%Carpet.BfieldCalc(6)=Euler(3);
%Carpet.SpeedObserver=ve;
%Carpet.SpeedBody=vb;
%Carpet.Rotation=rot';
```

D.2 Function force_coil2

```
function FT=forcecoil(Coil, MagnetArray, Carpet)
%global segments
ox=Coil.Position(1);
oy=Coil.Position(2);
oz=Coil.Position(3);
l=Coil.Size(1);
w=Coil.Size(2);
h=Coil.Size(3);
th=Coil.Size(4);
fr=Coil.Size(5);
nt=Coil.Turns;
I=Coil.Current;


J=I*nt/(th*h); %current density



%forces on the retangular parts of the coil
F=[0 0 0];
T=[0 0 0];

if abs(J)>1e-9;
nseg=10; %number of segments in straight part of coil
```

```
[wp,ww,hp,hw]=coilmatrix(th, h, (l-2*fr)/nseg);


%figure
%hold on

%             s3
%       -------<-------
%   c2 /               \ c1
%     /                 \
%     |                   |
%     |                   |
% s4 v                   ^ s2
%     |                   |
%     |                   |
%     \                 /
%   c3 \_______>_______/ c4
%             s1
%

%straigt segment 1
dl=(l-2*fr)/nseg;
for nx=1:nseg
    x=ox-0.5*l+fr+(nx-0.5)*dl;
    for ny=1:length(wp)
        y=oy-0.5*w+0.5*th+wp(ny);
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([x, y, z], Carpet));
            I=mc_tfd_coil([J 0 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dl*ww(ny)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

%straigt segment 3
for nx=1:nseg
    x=ox-0.5*l+fr+(nx-0.5)*dl;
    for ny=1:length(wp)
        y=oy+0.5*w-0.5*th+wp(ny);
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([x, y, z], Carpet));
            I=mc_tfd_coil([-J 0 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dl*ww(ny)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

%straigt segment 4
dl=(w-2*fr)/nseg;
for ny=1:nseg
    y=oy-0.5*w+fr+(ny-0.5)*dl;
    for nx=1:length(wp)
        x=ox-0.5*l+0.5*th+wp(nx);
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([x, y, z], Carpet));
            I=mc_tfd_coil([0 -J 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dl*ww(nx)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

%straigt segment 2
for ny=1:nseg
    y=oy-0.5*w+fr+(ny-0.5)*dl;
    for nx=1:length(wp)
        x=ox+0.5*l-0.5*th+wp(nx);
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([x, y, z], Carpet));
            I=mc_tfd_coil([0 J 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dl*ww(nx)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end


%forces on the circular parts of the coil
%nseg=10;
elemnt=(l-2*fr)/nseg;
nseg=ceil((2.5e-3*0.5*pi)/elemnt);
dphi=pi/2/nseg;
```

```
%circular segment 1
for p=1:nseg
    phi=dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([ox+0.5*l-fr+r*cos(phi),
               oy+0.5*w-fr+r*sin(phi), z], Carpet));
            I=mc_tfd_coil(J*[-sin(phi), cos(phi), 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dr*ww(nr)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

%circular segment 2
for p=1:nseg
    phi=0.5*pi+dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([ox-0.5*l+fr+r*cos(phi),
               oy+0.5*w-fr+r*sin(phi), z], Carpet));
            I=mc_tfd_coil(J*[-sin(phi), cos(phi), 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dr*ww(nr)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

%circular segment 3
for p=1:nseg
    phi=pi+dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([ox-0.5*l+fr+r*cos(phi),
                oy-0.5*w+fr+r*sin(phi), z], Carpet));
            I=mc_tfd_coil(J*[-sin(phi), cos(phi), 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dr*ww(nr)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

%circular segment 4
for p=1:nseg
    phi=3/2*pi+dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            B=field_array_pm(MagnetArray, mc_tf_coil([ox+0.5*l-fr+r*cos(phi),
                oy-0.5*w+fr+r*sin(phi), z], Carpet));
            I=mc_tfd_coil(J*[-sin(phi), cos(phi), 0], Carpet);
            Fs=-cross(I, B);
            Fs=Fs*dr*ww(nr)*hw(nz)*th*h;
            Ts=cross(mc_tft_coil([x y z], Carpet), Fs);
            F=F+Fs;
            T=T+Ts;
        end
    end
end

end
FT=[F(1) F(2) F(3) T(1) T(2) T(3)];
```

D.3 Function coilmatrix

```
function [wp,ww,hp,hw]=coilmatrix(cw, ch, res)
%coilmatrix calcultaes the points for the numerical
%integration using a enhanced Abramowitz method.
%ch: height opf the coil
%cw: width of the coil
%wp, hp: the positions of the points in the width/height
%ww, hw: the weight of the points in the width/height


wn=fix(cw/res)-1;
hn=fix(ch/res)-1;
if wn<1
    wn=1;
```

```
end
if hn<1
    hn=1;
end

ww(1)=5;
for n=2:wn+1
    ww(n)=8;
end
ww(n+1)=5;
ww=ww/sum(ww);

h=cw/((length(ww)-2)*8+10);
wp(1)=-cw/2+h*9-0.5*sqrt(3/5)*h*18;
wp(2)=-cw/2+h*9;
for n=3:length(ww)-1
    wp(n)=wp(n-1)+h*8;
end
l=length(wp);
wp(l+1)=wp(l)+0.5*sqrt(3/5)*h*18;


hw(1)=5;
for n=2:hn+1
    hw(n)=8;
end
hw(n+1)=5;
hw=hw/sum(hw);

h=ch/((length(hw)-2)*8+10);
hp(1)=-ch/2+h*9-0.5*sqrt(3/5)*h*18;
hp(2)=-ch/2+h*9;
for n=3:length(hw)-1
    hp(n)=hp(n-1)+h*8;
end
l=length(hp);
hp(l+1)=hp(l)+0.5*sqrt(3/5)*h*18;


plt=0;
if plt==1
    figure
    hold on
    plot([-cw/2 cw/2],[-ch/2 -ch/2]);
    plot([-cw/2 cw/2],[ch/2 ch/2]);
    plot([-cw/2 -cw/2],[ch/2 -ch/2]);
    plot([cw/2 cw/2],[ch/2 -ch/2]);
    for na=1:length(wp)
        for nb=1:length(hp)
            plot(wp(na), hp(nb), 'o')
        end
    end
end
```

D.4 Function mc_tf_coil

```
function mc=mc_tf_coil(coil, carpet)


Rotpsi=    [1  0                             0                            0;
            0  cos(carpet.BFieldCalc(4))    -sin(carpet.BFieldCalc(4))    0;
            0  sin(carpet.BFieldCalc(4))     cos(carpet.BFieldCalc(4))    0;
            0  0                             0                            1];

Rottheta= [ cos(carpet.BFieldCalc(5))   0    sin(carpet.BFieldCalc(5))  0;
            0                           1    0                          0;
           -sin(carpet.BFieldCalc(5))   0    cos(carpet.BFieldCalc(5))  0;
            0                           0    0                          1];

Rotphi=    [cos(carpet.BFieldCalc(6)) -sin(carpet.BFieldCalc(6))   0   0
            sin(carpet.BFieldCalc(6))  cos(carpet.BFieldCalc(6))   0   0
            0                          0                           1   0
            0                          0                           0   1];

Rot=Rotphi*Rottheta*Rotpsi;


Trans=     [1     0   0    carpet.BFieldCalc(1);
            0     1   0    carpet.BFieldCalc(2);
            0     0   1    carpet.BFieldCalc(3);
            0     0   0    1];


mc=inv(Trans*Rot)*[coil(1) coil(2) coil(3) 1]';
mc=[mc(1) mc(2) mc(3)];
```

D.5 Function mc_tfd_coil

```
function mc=mc_tfd_coil(coil, carpet)


Rotpsi=    [1  0                             0                            0;
            0  cos(carpet.MassCenter(4))    -sin(carpet.MassCenter(4))    0;
            0  sin(carpet.MassCenter(4))     cos(carpet.MassCenter(4))    0;
            0  0                             0                            1];

Rottheta= [ cos(carpet.MassCenter(5))   0    sin(carpet.MassCenter(5))  0;
```

```
           0                            1   0                        0;
          -sin(carpet.MassCenter(5))    0   cos(carpet.MassCenter(5))  0;
           0                            0   0                        1];

Rotphi=   [cos(carpet.MassCenter(6)) -sin(carpet.MassCenter(6))   0   0
           sin(carpet.MassCenter(6))  cos(carpet.MassCenter(6))   0   0
           0                          0                           1   0
           0                          0                           0   1];

Rot=Rotphi*Rottheta*Rotpsi;

mc=inv(Rot)*[coil(1) coil(2) coil(3) 1]';
mc=[mc(1) mc(2) mc(3)];
```

D.6 Function mc_tft_coil

```
function mc=mc_tft_coil(coil, carpet)


Rotpsi=   [1  0                           0                          0;
           0  cos(carpet.MassCenter(4))  -sin(carpet.MassCenter(4))  0;
           0  sin(carpet.MassCenter(4))   cos(carpet.MassCenter(4))  0;
           0  0                           0                          1];

Rottheta= [ cos(carpet.MassCenter(5))   0   sin(carpet.MassCenter(5))  0;
            0                           1   0                          0;
           -sin(carpet.MassCenter(5))   0   cos(carpet.MassCenter(5))  0;
            0                           0   0                          1];

Rotphi=   [cos(carpet.MassCenter(6)) -sin(carpet.MassCenter(6))   0   0
           sin(carpet.MassCenter(6))  cos(carpet.MassCenter(6))   0   0
           0                          0                           1   0
           0                          0                           0   1];

Rot=Rotphi*Rottheta*Rotpsi;


Trans=    [1    0   0   carpet.MassCenter(1);
           0    1   0   carpet.MassCenter(2);
           0    0   1   carpet.MassCenter(3);
           0    0   0   1];


mc=inv(Trans*Rot)*[coil(1) coil(2) coil(3) 1]';
mc=[mc(1) mc(2) mc(3)];
```

D.7 Function DOFsub_AXIS

```
function out=DOFsub_AXIS(Euler)

out(1,1)=cos(Euler(3))*cos(Euler(2));
out(1,2)=sin(Euler(3))*cos(Euler(2));
out(1,3)=-sin(Euler(2));
out(2,1)=cos(Euler(3))*sin(Euler(2))*sin(Euler(1))
         -sin(Euler(3))*cos(Euler(1));
out(2,2)=sin(Euler(3))*sin(Euler(2))*sin(Euler(1))
         +cos(Euler(3))*cos(Euler(1));
out(2,3)=cos(Euler(2))*sin(Euler(1));
out(3,1)=cos(Euler(3))*sin(Euler(2))*cos(Euler(1))
         +sin(Euler(3))*sin(Euler(1));
out(3,2)=sin(Euler(3))*sin(Euler(2))*cos(Euler(1))
         -cos(Euler(3))*sin(Euler(1));
out(3,3)=cos(Euler(2))*cos(Euler(1));
```

D.8 Function DOFsub_dot

```
function out=DOFsub_dot(roto, Eulero)

out(1,1)=roto(1)+roto(2)*sin(Eulero(1))*tan(Eulero(2))+
         roto(3)*cos(Eulero(1))*tan(Eulero(2));
out(2,1)=roto(2)*cos(Eulero(1))-roto(3)*sin(Eulero(1));
out(3,1)=roto(2)*sin(Eulero(1))/cos(Eulero(2))+roto(3)
         *cos(Eulero(1))/cos(Eulero(2));
```

Appendix E

Matlab code for the inductance calculation

The function `inductance` can be used to calculate the self and mutual inductance of two structs of `CoilArray`. The first struct is the excited coil, the second the linked coil.
Author: J.W. Jansen

```
inductance(CoilArray(1), CoilArray(2))
```

E.1 Function inductance

```
function L=inductance(Coil1, Coil2)

ox=Coil1.Position(1);
oy=Coil1.Position(2);
oz=Coil1.Position(3);
l=Coil1.Size(1);
w=Coil1.Size(2);
h=Coil1.Size(3);
th=Coil1.Size(4);
fr=Coil1.Size(5);
nt=Coil1.Turns;

global nseg
%number of segments in straight part of coil
[wp,ww,hp,hw]=coilmatrix(th, h, (l-2*fr)/nseg);

%            s3
%       -------<-------
%   c2 /             \ c1
%     /               \
%    |                  |
%    |                  |
% s4 v                  ^ s2
%    |                  |
%    |                  |
%     \                 /
%   c3 \_______>_______/ c4
%             s1
%
L=0;

%straigt segment 1
dl=(l-2*fr)/nseg;
for nx=1:nseg
    x=ox-0.5*l+fr+(nx-0.5)*dl;
    for ny=1:length(wp)
        y=oy-0.5*w+0.5*th+wp(ny);
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, x, y, z);
            n=[1 0 0];
            A=dot(A,n);
            Ls=A*dl*ww(ny)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end


%straigt segment 3
for nx=1:nseg
    x=ox-0.5*l+fr+(nx-0.5)*dl;
    for ny=1:length(wp)
        y=oy+0.5*w-0.5*th+wp(ny);
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, x, y, z);
            n=[-1 0 0];
            A=dot(A,n);
            Ls=A*dl*ww(ny)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end

%straigt segment 4
dl=(w-2*fr)/nseg;
for ny=1:nseg
    y=oy-0.5*w+fr+(ny-0.5)*dl;
    for nx=1:length(wp)
        x=ox-0.5*l+0.5*th+wp(nx);
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, x, y, z);
            n=[0 -1 0];
            A=dot(A,n);
            Ls=A*dl*ww(nx)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end

%straigt segment 2
for ny=1:nseg
    y=oy-0.5*w+fr+(ny-0.5)*dl;
    for nx=1:length(wp)
        x=ox+0.5*l-0.5*th+wp(nx);
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, x, y, z);
            n=[0 1 0];
            A=dot(A,n);
            Ls=A*dl*ww(nx)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end


%forces on the circular parts of the coil
%nseg=10;
elemnt=(l-2*fr)/nseg;
nseg=ceil((2.5e-3*0.5*pi)/elemnt);
dphi=pi/2/nseg;

%circular segment 1
for p=1:nseg
    phi=dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, ox+0.5*l-fr+r*cos(phi),
               oy+0.5*w-fr+r*sin(phi), z);
            n=[-sin(phi), cos(phi), 0];
            A=dot(A,n);
            Ls=A*dr*ww(nr)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end

%circular segment 2
for p=1:nseg
```

```
    phi=0.5*pi+dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, ox-0.5*l+fr+r*cos(phi),
               oy+0.5*w-fr+r*sin(phi), z);
            n=[-sin(phi), cos(phi), 0];
            A=dot(A,n);
            Ls=A*dr*ww(nr)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end

%circular segment 3
for p=1:nseg
    phi=pi+dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, ox-0.5*l+fr+r*cos(phi),
               oy-0.5*w+fr+r*sin(phi), z);
            n=[-sin(phi), cos(phi), 0];
            A=dot(A,n);
            Ls=A*dr*ww(nr)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
end

%circular segment 4
for p=1:nseg
    phi=3/2*pi+dphi*(p-1)+0.5*dphi;
    for nr=1:length(wp)
        r=fr-0.5*th+wp(nr);
        dr=0.5*pi*r/nseg;
        for nz=1:length(hp)
            z=oz+hp(nz);
            A=vecpot_coil2(Coil2, ox+0.5*l-fr+r*cos(phi),
               oy-0.5*w+fr+r*sin(phi), z);
            n=[-sin(phi), cos(phi), 0];
            A=dot(A,n);
            Ls=A*dr*ww(nr)*hw(nz)*th*h;
            L=L+Ls;
        end
    end
```

```
end

L=L*nt/(th*h);
```

E.2 Function coilmatrix

See D.3.

E.3 Function vecpot_coil2

```
function A=vecpot_coil2(Coil, xp, yp, zp)

ox=Coil.Position(1);
oy=Coil.Position(2);
oz=Coil.Position(3);
l=Coil.Size(1);
w=Coil.Size(2);
h=Coil.Size(3);
th=Coil.Size(4);
fr=Coil.Size(5);

global nseg2;
 %number of segments in straight part of coil
[wp,ww,hp,hw]=coilmatrix(th, h, (l-2*fr)/nseg2);

A=0;
for nz=1:length(hp)
    z=oz+hp(nz);
    for nw=1:length(wp)
        A=A+vecpotlinecoil(ox, oy, oz, 2*(0.5*l-0.5*th+wp(nw)),
           2*(0.5*w-0.5*th+wp(nw)), z, th, 0.5*th+wp(nw)+fr-th, 1,
           1*ww(nw)*hw(nz), xp, yp, zp);
    end
end
```

E.4 Function vecpot_linecoil

```
function A=vecpotlinecoil(ox, oy, oz, l, w, h, th, fr, nt, I, x, y, z)
%ox, oy, oz: origin coil
%l, w, h: lenght, width, height of the outside of the coil
%th: thickness
%fr: fillet radius
%nt: number of turns
%I: current (in one turn)
%x, y, z: evaluation point

A=0;
```

```
segments=5;

%straight segments
A = A + vecpotlc(ox-0.5*l+fr, oy-0.5*w, oz+h, ox+0.5*l-fr,
    oy-0.5*w, oz+h, I*nt, x, y, z);
A = A + vecpotlc(ox+0.5*l, oy-0.5*w+fr, oz+h, ox+0.5*l,
    oy+0.5*w-fr, oz+h, I*nt, x, y, z);
A = A + vecpotlc(ox+0.5*l-fr, oy+0.5*w, oz+h, ox-0.5*l+fr,
    oy+0.5*w, oz+h, I*nt, x, y, z);
A = A + vecpotlc(ox-0.5*l, oy+0.5*w-fr, oz+h, ox-0.5*l,
    oy-0.5*w+fr, oz+h, I*nt, x, y, z);

%round segments
for n=1:segments
    dphi=0.5*pi/segments;
    A = A + vecpotlc(ox+0.5*l-fr + fr*sin(dphi*(n-1)), oy-0.5*w+fr -
        fr*cos(dphi*(n-1)), oz+h, ox+0.5*l-fr + fr*sin(dphi*n), oy-0.5*w+fr -
        fr*cos(dphi*n), oz+h, I*nt, x, y, z);
    A = A + vecpotlc(ox+0.5*l-fr + fr*cos(dphi*(n-1)), oy+0.5*w-fr +
        fr*sin(dphi*(n-1)), oz+h, ox+0.5*l-fr + fr*cos(dphi*n), oy+0.5*w-fr +
        fr*sin(dphi*n), oz+h, I*nt, x, y, z);
    A = A + vecpotlc(ox-0.5*l+fr - fr*sin(dphi*(n-1)), oy+0.5*w-fr +
        fr*cos(dphi*(n-1)), oz+h, ox-0.5*l+fr - fr*sin(dphi*n), oy+0.5*w-fr +
        fr*cos(dphi*n), oz+h, I*nt, x, y, z);
    A = A + vecpotlc(ox-0.5*l+fr - fr*cos(dphi*(n-1)), oy-0.5*w+fr -
        fr*sin(dphi*(n-1)), oz+h, ox-0.5*l+fr - fr*cos(dphi*n), oy-0.5*w+fr -
        fr*sin(dphi*n), oz+h, I*nt, x, y, z);
end

plt=0;
if plt==1
    %figure;
    hold on;
    line([ox-0.5*l+fr ox+0.5*l-fr], [oy-0.5*w oy-0.5*w]);
    line([ox+0.5*l ox+0.5*l], [oy-0.5*w+fr oy+0.5*w-fr]);
    line([ox+0.5*l-fr ox-0.5*l+fr], [oy+0.5*w oy+0.5*w]);
    line([ox-0.5*l ox-0.5*l], [oy+0.5*w-fr oy-0.5*w+fr]);
    for n=1:segments
        dphi=0.5*pi/segments;
        line([ox+0.5*l-fr + fr*sin(dphi*(n-1)) ox+0.5*l-fr + fr*sin(dphi*n)],
           [oy-0.5*w+fr - fr*cos(dphi*(n-1)) oy-0.5*w+fr - fr*cos(dphi*n)]);
        line([ox+0.5*l-fr + fr*cos(dphi*(n-1)) ox+0.5*l-fr + fr*cos(dphi*n)],
           [oy+0.5*w-fr + fr*sin(dphi*(n-1)) oy+0.5*w-fr + fr*sin(dphi*n)]);
        line([ox-0.5*l+fr - fr*sin(dphi*(n-1)) ox-0.5*l+fr - fr*sin(dphi*n)],
           [oy+0.5*w-fr + fr*cos(dphi*(n-1)) oy+0.5*w-fr + fr*cos(dphi*n)]);
        line([ox-0.5*l+fr - fr*cos(dphi*(n-1)) ox-0.5*l+fr - fr*cos(dphi*n)],
           [oy-0.5*w+fr - fr*sin(dphi*(n-1)) oy-0.5*w+fr - fr*sin(dphi*n)]);
    end
    grid on;
end
```

E.5 Function vecpotlc

```
function A=vecpotlc(x1, y1, z1, x2, y2, z2, I, x, y, z)
%function calculates the vector potential caused by a
%line-current in the point x, y, z.
%The current flows from point x1, y1, z1 in a straight line to x2, y2, z2.

mu=4e-7*pi;

ax=x1-x2;
ay=y1-y2;
az=z1-z2;

bx=x-x1;
by=y-y1;
bz=z-z1;

cx=x2-x;
cy=y2-y;
cz=z2-z;

RS=sqrt(ax^2+ay^2+az^2);
TR=sqrt(bx^2+by^2+bz^2);
TS=sqrt(cx^2+cy^2+cz^2);

la=ax/RS;
ma=ay/RS;
na=az/RS;

RP=la*bx + ma*by + na*bz;
PS=la*cx + ma*cy + na*cz;
TP = sqrt(TR^2 - RP^2);

%G=sqrt((az*cy-ay*cz)^2+(ax*cz-az*cx)^2+(ay*cx-ax*cy)^2);
%if G~=0
%    lb=(az*cy-ay*cz)/G
%    mb=(ax*cz-az*cx)/G
%    nb=(ay*cx-ax*cy)/G
%else
%    lb=0;
%    mb=0;
%    nb=0;
%    warning('Division by zero')
%end

if (TP~=0&TR~=0&TS~=0)
    A=mu*I/(4*pi)*(asinh(-RP/-TP)-asinh(-PS/TP));
else
    warning('Division by zero')
    A=0;
```

```
end
Ax=la*A;
Ay=ma*A;
Az=na*A;
A=[Ax, Ay, Az];
```

PIONEER CORPORATION
4-1, MEGURO 1-CHOME, MEGURO-KU, TOKYO 153-8654, JAPAN

PIONEER ELECTRONICS (USA) INC.
P.O. Box 1540, Long Beach, California 90801-1540, U.S.A.
TEL: (800) 421-1404

PIONEER EUROPE NV
Haven 1087, Keetberglaan 1, B-9120 Melsele, Belgium
TEL: (0) 3/570.05.11

PIONEER ELECTRONICS ASIACENTRE PTE. LTD.
253 Alexandra Road, #04-01, Singapore 159936
TEL: 65-472-1111

PIONEER ELECTRONICS AUSTRALIA PTY. LTD.
178-184 Boundary Road, Braeside, Victoria 3195, Australia
TEL: (03) 9586-6300

PIONEER ELECTRONICS OF CANADA, INC.
300 Allstate Parkway, Markham, Ontario L3R OP2, Canada
TEL: (905) 479-4411

PIONEER ELECTRONICS DE MEXICO, S.A. de C.V.
San Lorenzo 1009 3er. Piso Desp. 302
Col. Del Valle Mexico, D.F. C.P. 03100
TEL: 5-688-52-90

Printed in Thailand
Imprimé en Thaïlande

Appendix F

Finite element models

F.1 FLUX3D

Figure F.1 shows the top view of the permanent magnets and the coils in the planar motor model created in FLUX3D. The coils are built of four rectangular segments and four cylindrical segments. Figure F.2 shows a perspective view on the coils.

Figure F.3 shows the mesh on the magnet array. A tetrahedral mesh has been applied. A line mesh generator has been created to assign the mesh to the lines of the magnets. The mesh division on the lines is 2 mm.

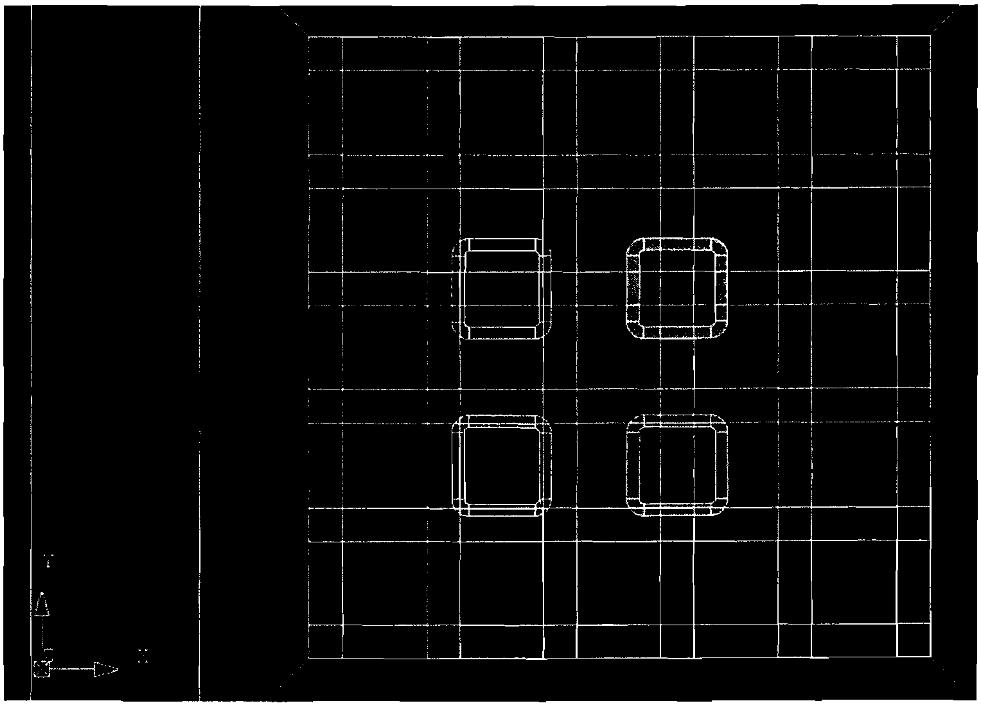


Figure F.1. *Top view on the coils and the magnet array in FLUX3D.*

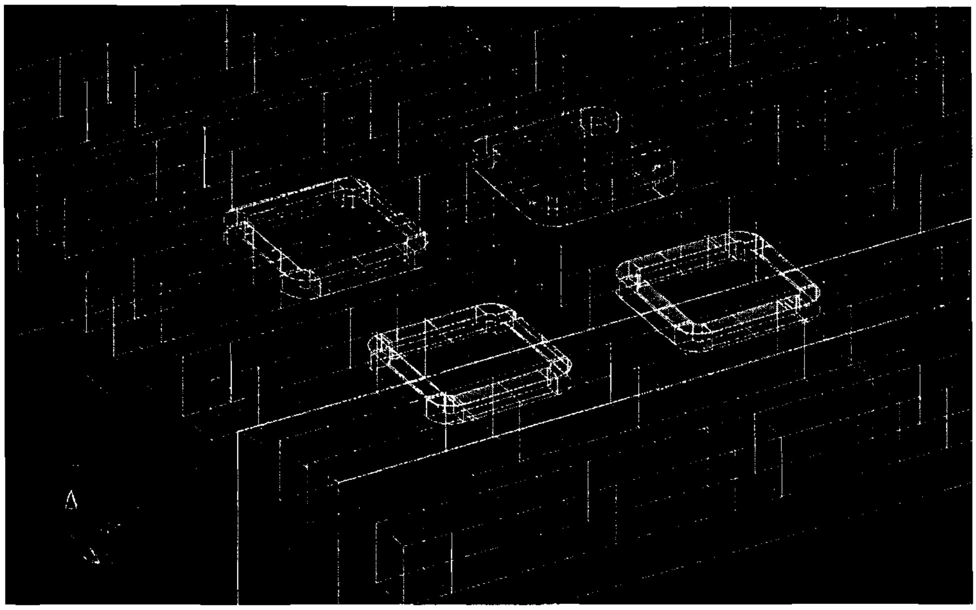


Figure F.2. *Perspective view on the coils and the magnet array in FLUX3D.*

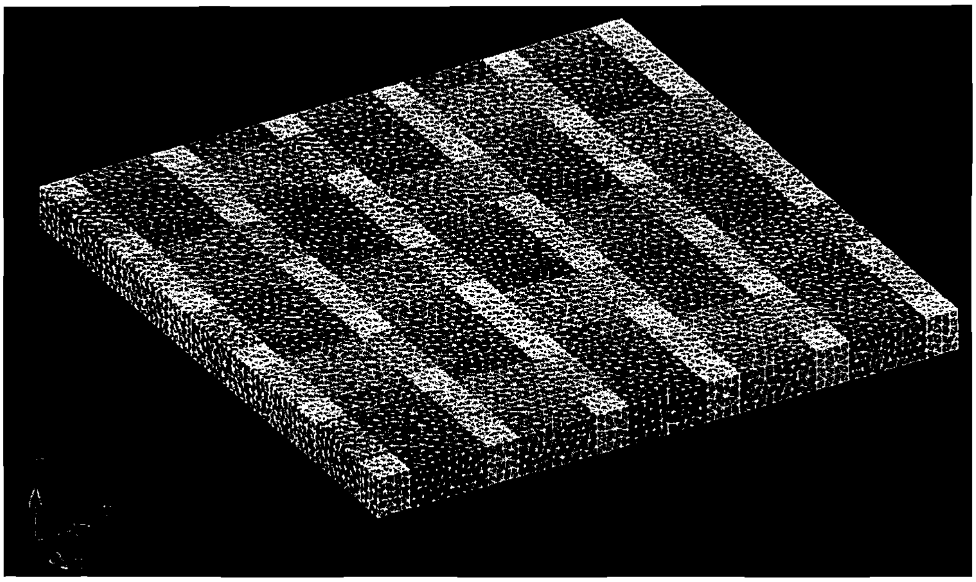


Figure F.3. *Perspective view on the coils and the magnet array in FLUX3D.*

F.2 Maxwell 3D

Figure F.4 shows the bottom view of the permanent magnets and the coils (dark green) in the planar motor model created in Maxwell 3D. The coils are constructed of single volumes.

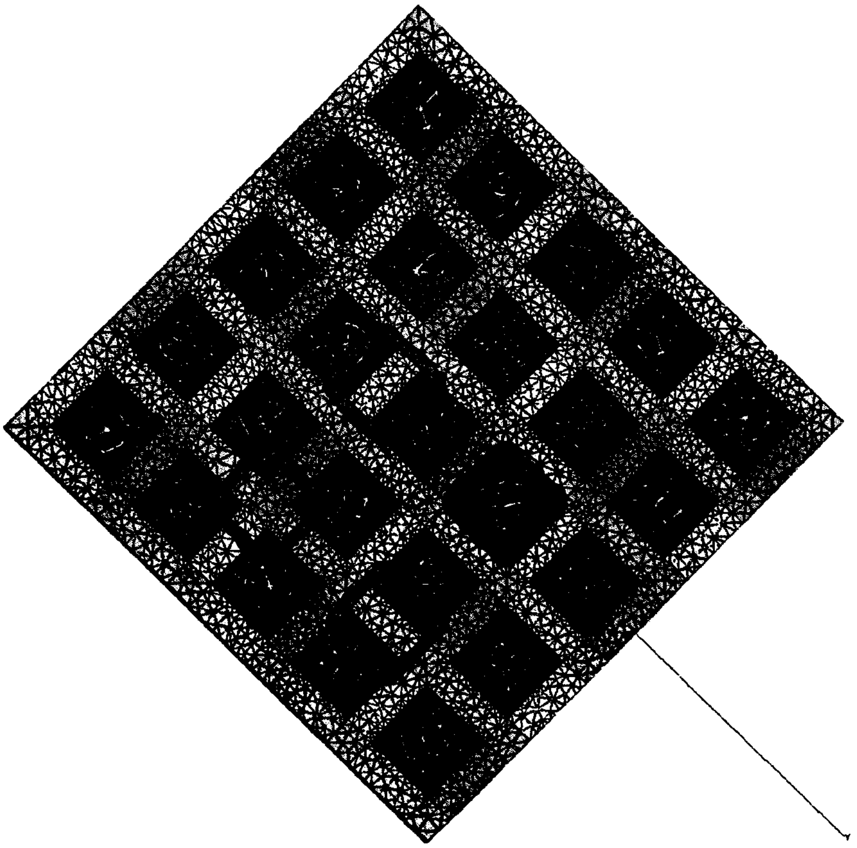


Figure F.4. *Bottom view on the coils and the magnet array in Maxwell 3D.*

Figure F.5 a detail of the tetrahedral mesh of the magnets and a coil is shown. The automatic mesh generator created a mesh, which is denser near the edges of the magnets. Therefore, the magnetic field density errors are smaller near the edges of the magnets than in FLUX3D. However, the mesh on the coils is extremely dense on the round corners, because of small radius.

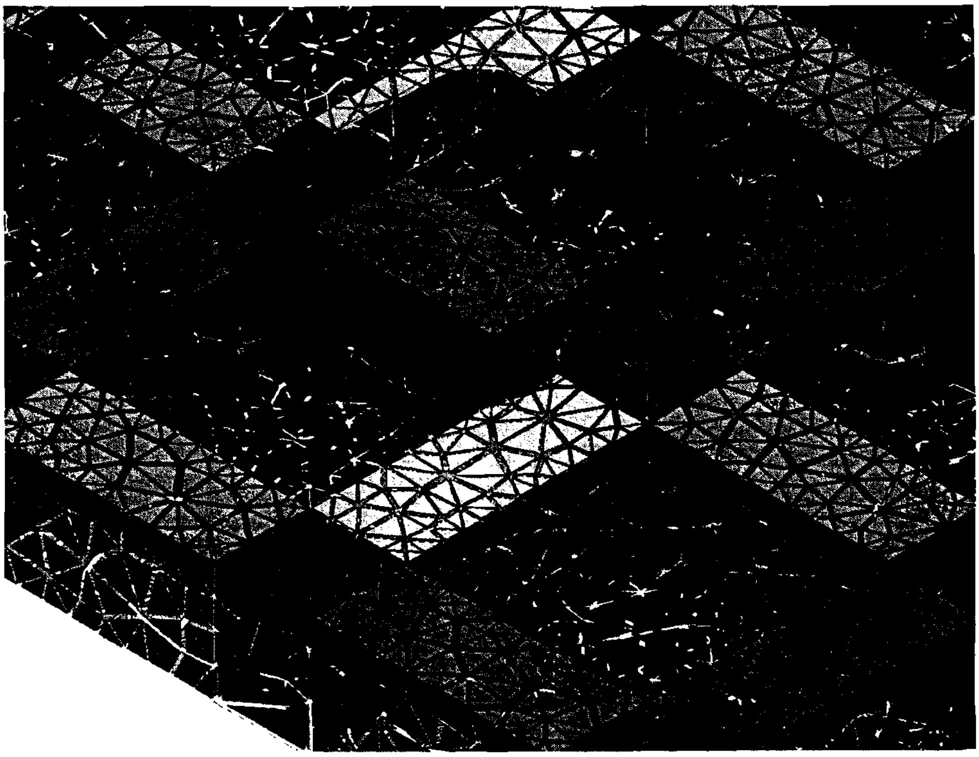


Figure F.5. *Detail of the mesh of a coil and the magnet array in Maxwell 3D.*