Modelling of disturbance forces of a x-y manipulator on a floating platform

Zuurendonk, B.P.

Award date:
2007

Modelling of Disturbance Forces of a x-y Manipulator on a Floating Platform

B.P. Zuurendonk

DCT 2007.028

Master's thesis

Coach(es): J. de Boeij

Supervisor: M. Steinbuch

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Technology Group

Eindhoven, March 05, 2007

Modelling of Disturbance Forces of a x-y Manipulator on a Floating Platform

Bastiaan Zuurendonk*, Jeroen de Boeij**, Maarten Steinbuch*, Elena Lomonova**

Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

* Department of Mechanical Engineering

** Department of Electrical Engineering

***Abstract*—This paper describes a multi-body model for predicting disturbance forces and torques caused by a multi-stage manipulator (2 DOF) on a floating platform (6 DOF). The manipulator consists of a beam attached to two parallel linear ironless actuators and a rotary arm underneath the beam. The model prediction can be used as a feed-forward in the control of the magnetic bearings of the platform.**

Lagrange's equations are used for the analysis, which provide a convenient way for deriving a multi-body model without the need for separating all bodies and calculation of all interacting forces and torques.

The model is verified with measurements. The experimental setup consists of a manipulator on an aluminium platform which is connected to the fixed world via a 6 DOF force-torque sensor which is used to verify the model.

I. Introduction

Most high-precision machines consist of several positioning stages which are often a cascaded set of long-stroke actuators with low precision and short-stroke actuators for high precision positioning [1], [2], [3]. In order to decrease production costs and time there is an increased demand for higher productivity of such machines. Several options are available in order to tackle this problem:

1) Faster machines could be built which would need more powerful actuators and, therefore, lead to increased mechanical and thermal stresses, resulting in a bulkier design.
2) The batch size of the production process could be increased which would lead to actuators with a longer stroke.
3) Parallel processing could be used. In this case another task will be performed while positioning. This way it is possible to improve performance without the need for increased machine size.

The first two options do not only result in heavier machines, but also compromise the accuracy of the machines. Parallel processing however does not have this drawback. At Eindhoven University of Technology such a parallel machine is currently under development (see Fig. 1). The goal of the project is to build a contactless planar actuator with a manipulator on top of it. Robots on the fixed world can place products on the planar actuator, which in turn can be used for transportation and positioning of the same product. While moving, the manipulator can be used for e.g. inspection or calibration of the product. Increased reliability and dynamics will result from removing all cables which connect it to the fixed world. Therefore, three different contactless techniques should be realised in this project:

1) Contactless movement of the planar actuator by using magnetic bearings and propulsion.
2) Contactless energy transfer by using inductive coupling.
3) Wireless control of the manipulator using a wireless low-latency data link.

The energy, which is necessary to operate the manipulator, is provided by the contactless inductive coupling. Furthermore, the manipulator is controlled from the ground via the wireless data link.

II. Mechanical design of the manipulator

The planar actuator consists of an array of stationary coils, above which an ironless platform with permanent magnets is floating. The manipulator on top of the platform is an H-drive with two ironless linear actuators attached to a beam. In the centre of the beam a rotary motor is assembled with an arm attached to it. The tip of this arm can be positioned anywhere in the x-y plane between the two horizontal linear legs by combining the translation of the linear actuators and the rotary movement of the arm.

The linear actuators are brushless 3-phase ironless actuators. Therefore, they have no cogging. The rotary drive is a 3-phase slotless motor, and therefore, also has no cogging. The actuator properties are listed in Table I. The position of each linear actuator is measured with incremental encoders which have a 1 μm resolution. The angle of the rotary motor is measured

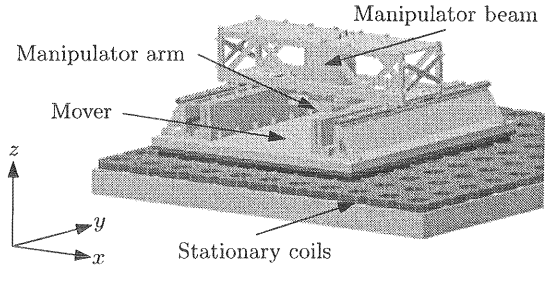

Fig. 1. Contactless planar actuator with manipulator

TABLE I
ACTUATOR PROPERTIES

Property	Linear act.	Rotary motor
Motor constant	11.4 [N/A_{rms}]	0.144 [Nm/A_{rms}]
Peak current	3.1 [A_{rms}]	10.8 [A_{rms}]
Continuous current	0.87 [A_{rms}]	1.92 [A_{rms}]
Continuous force	10 [N]	-
Continuous torque	-	0.28 [Nm]

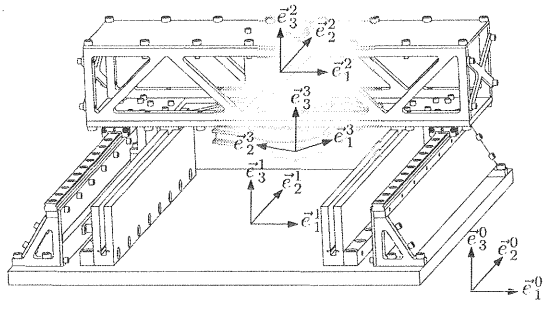

Fig. 2. Coordinate systems in the manipulator

with a 40 μrad resolution. The beam which connects the linear actuators is sufficiently stiff in order to consider it rigid. Therefore, movements of the beam in directions other than the movement direction of the linear actuators are considered impossible.

III. Model derivation using Lagrange's equations

The derivation of the multi-body model is done by using Lagrange's equations [4], [5], [6]. These equations eliminate the need for computing all interacting forces between different bodies. Only external forces and constraint forces of interest have to be taken into account.

A. Coordinate systems

The manipulator consists of three separate bodies. The platform, magnet tracks and bearings together are the first body with mass, m_1. The second body is the beam and has a mass, m_2. The rotor of the rotary motor together with the arm form the last body with mass, m_3. Each body has a local coordinate system attached to it (see Fig. 2), which is used for describing the position and orientation with respect to the global coordinate system. Each coordinate system consists of three mutually orthogonal unit vectors:

$$\underline{\vec{e}}^i = \begin{bmatrix} \vec{e}_1^i & \vec{e}_2^i & \vec{e}_3^i \end{bmatrix}^T \text{ for } i = 0, \ldots, 3. \quad (1)$$

$\underline{\vec{e}}^0$ is fixed to the world and can, therefore, not move, $\underline{\vec{e}}^1$ is placed in the centre of the manipulator platform, $\underline{\vec{e}}^2$ is located in the centre of the beam and $\underline{\vec{e}}^3$, finally, is attached to the rotary arm.

B. Position and orientation of the bodies

The position of the center of mass of each body with respect to the fixed world is written as:

$$\vec{r}_{CM_i} = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix} \underline{\vec{e}}^0. \quad (2)$$

The orientation of each body is described by means of Tait-Bryant angles. The orientation of a body is the result of subsequent rotations θ_i, ψ_i and ϕ_i about, respectively, the local $\vec{e}_1^i$, $\vec{e}_2^i$ and $\vec{e}_3^i$ axis. By the use of rotation matrices, $\underline{A}^{ij}$ the transformation from one coordinate system to another can now be easily made:

$$\vec{e}_j = \underline{A}^{ji} \vec{e}_i. \quad (3)$$

Rotation matrices and their properties are discussed in more detail in appendix A.

C. Generalised coordinates

In general, a body has 6 degrees of freedom (DOF) if it is not subject to any constraints. In presence of constraints, each constraint removes one degree of freedom. The minimum required set of coordinates to describe the position and orientation of each body are a set of n generalised coordinates, $\underline{q}$. So $\underline{q}$ is a $(n \times 1)$ column. The floating platform has 6 DOF due to its complete freedom with respect to the fixed world. Furthermore, the manipulator adds 1 DOF due to the translation of the beam and 1 DOF due to the rotational movement of the rotary motor. The beam is considered rigid and roller bearings on each side do not allow other movements of the beam than the one in $\vec{e}_2^1$-direction. Therefore, only 1 DOF is added by the beam. So a column of generalised coordinates for the manipulator on the floating platform is:

$$\underline{q} = \begin{bmatrix} x_1 & y_1 & z_1 & \theta_1 & \psi_1 & \phi_1 & y_{LM} & \phi_{RM} \end{bmatrix}^T, \quad (4)$$

where y_{LM} and ϕ_{RM} denote, respectively, the movement of the beam and rotation of the arm. Now the position vectors, $\vec{r}_{CM_i}$, can be written in terms of $\underline{q}$.

D. Kinetic and potential energy

The next step in the Lagrangian approach is to define the total energy available in the system. The total kinetic energy of the system is the sum of the translational and rotational kinetic energy of each body:

$$T = \frac{1}{2} \sum_{i=1}^{3} \left(m_i \dot{\vec{r}}_{CM_i} \cdot \dot{\vec{r}}_{CM_i} + {}^{i0}\vec{\omega} \cdot {}^{i}_{CM_i}\mathbf{J} \cdot {}^{i0}\vec{\omega} \right), \quad (5)$$

where m_i denotes the mass of body i, ${}^{i0}\vec{\omega}$ is the angular velocity of body i with respect to the fixed world and ${}^{i}_{CM_i}\mathbf{J}$ is the inertia tensor of body i with respect to its centre of mass.

The total potential energy of the system is the sum of the potential of the gravitational forces acting on each body and the energy stored in the system in spring-elements. Because there are no springs in the manipulator and no flexibility is included in the model the potential energy equation reduces to:

$$V = -\sum_{i=1}^{3} m_i \vec{g} \cdot \vec{r}_{CM_i}, \quad (6)$$

with $\vec{g}$ the gravitational acceleration vector.

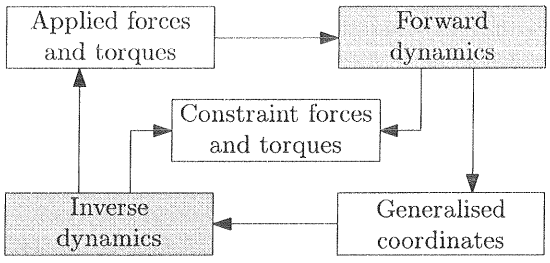

Fig. 3. Forward and inverse dynamics

E. Forward and inverse dynamic analysis

Once the equations of motion are derived, a dynamic analysis can be performed. This can be done by forward or inverse dynamic analysis (see Fig. 3). In the first case a column of external forces and torques, $\underline{\tau}$, is the input from which the trajectories of the generalised coordinates, $\underline{q}$, and constraint forces, $\underline{\lambda}$, are calculated. For an inverse dynamic analysis, trajectories for all generalised coordinates are the inputs from which the external forces and constraint forces are computed.

F. External forces and torques

In case a forward dynamic analysis is performed, the external forces and torques are the inputs for the model. The platform with manipulator is subject to several external forces and torques. Because the floating platform can move in three directions and rotate about three axes, an equal amount of forces and torques is necessary to control all these degrees of freedom. Furthermore there is the force of the linear motors which acts between the beam and the platform. The forces generated by the two linear motors will be treated as a single force in the middle of the beam. Furthermore, there is the torque generated by the rotary motor which acts between the arm and beam. Finally, friction forces in both the linear and rotary actuators are added as external forces.

In order to incorporate the external forces in the model they have to be rewritten as generalised non-conservative forces as described in [4]. Therefore, for each applied force an exertion point, $\vec{r}_i(\underline{q})$ is defined as well as a magnitude vector $\vec{F}_i^{nc}$. A similar approach is used for the applied torques, which results in the definition of a column with rotation parameters, $\underline{\theta}_j$, a row of axial vectors, $\underline{\vec{w}}(\underline{\theta}_j)$ about which the rotation is performed and a magnitude vector, $\vec{T}_j$. Now the generalised non-conservative forces can be written as:

$$\underline{Q}^{nc} = \sum_{i=1}^{n_F} \left(\frac{\partial \vec{r}_i}{\partial \underline{q}} \right)^T \cdot \vec{F}_i^{nc} + \sum_{j=1}^{n_T} \left(\frac{\partial \underline{\theta}_j}{\partial \underline{q}} \right)^T \underline{\vec{w}}(\underline{\theta}_j) \cdot \vec{T}_j^{nc}, \quad (7)$$

with n_F and n_T, respectively, the total number of external forces and external torques.

G. Constraint equations

In case an inverse dynamic analysis is performed the external forces and torques, which are related to the generalised coordinates, are substituted by a set of constraint equations. So only the damping forces remain as external forces. The platform is forced by m constraints to follow a certain trajectory. From these constraint equations, the constraint forces and torques necessary to follow the trajectory can be computed. All constraints will be included as velocity constraints and can therefore be written as:

$$\underline{h}_{nh}(\underline{\dot{q}}, \underline{q}, t) = \underline{0}, \quad (8)$$

where $\underline{h}_{nh}(\underline{\dot{q}}, \underline{q}, t)$ is a $(m \times 1)$ column with all constraint equations. Because all velocity components appear as terms which are linear in the generalised coordinates, the constraints are rewritten as:

$$\underline{W}^T(\underline{q}, t)\underline{\dot{q}} + \underline{\tilde{w}}(\underline{q}, t) = \underline{0}, \quad (9)$$

where $\underline{W}(\underline{q}, t)$ is a $(m \times n)$ matrix which represents the velocity dependent components, and $\underline{\tilde{w}}(\underline{q}, t)$, a $(m \times 1)$ column with the remaining non-velocity dependent components.

H. Lagrange multipliers

In order to incorporate the constraint equations in the equations of Lagrange, a $(n \times 1)$ column, $\underline{\lambda}$, of so called Lagrange multipliers is introduced. The Lagrange multipliers represent the constraint forces and torques. By writing the constraints now as $\underline{W\lambda}$ they can be treated as generalised forces in the equations of Lagrange.

I. Equations of Lagrange

The equations of motion now follow from the extended equations of Lagrange as described in [4]:

$$\left(\frac{d}{dt} \left(\frac{\partial T}{\partial \underline{\dot{q}}} \right) - \frac{\partial T}{\partial \underline{q}} + \frac{\partial V}{\partial \underline{q}} \right)^T = \underline{Q}^{nc} + \underline{W\lambda}. \quad (10)$$

The equations of Lagrange together with the constraint equations (9) now completely describe the dynamics of the system. In order to solve the equations they are rewritten and combined. Therefore, first the equations of Lagrange without constraints (i.e. $\underline{W\lambda} = \underline{0}$) are put in the following form:

$$\underline{M}(\underline{q})\underline{\ddot{q}} + \underline{H}(\underline{q}, \underline{\dot{q}}) = \underline{S}(\underline{q})\underline{\tau}, \quad (11)$$

where $\underline{M}(\underline{q})$ is the mass-matrix, $\underline{H}(\underline{q}, \underline{\dot{q}})$, a matrix with centripetal, Coriolis and gravitational terms and $\underline{S}(\underline{q})$, the distribution of external forces and torques, $\underline{\tau}$. The next step is differentiating the constraint equations (9) with respect to time:

$$\underline{W}^T(\underline{q}, t)\underline{\ddot{q}} + \underline{\bar{w}}(\underline{q}, \underline{\dot{q}}, t) = \underline{0}, \quad (12)$$

where

$$\underline{\bar{w}}(\underline{q}, \underline{\dot{q}}, t) = \frac{\partial \underline{\tilde{w}}(\underline{q}, t)}{\partial t} + \left(\frac{\partial \underline{W}^T(\underline{q}, t)}{\partial t} + \frac{\partial \underline{W}^T(\underline{q}, t)\underline{\dot{q}}}{\partial \underline{q}} + \frac{\partial \underline{\tilde{w}}(\underline{q}, t)}{\partial \underline{q}} \right) \underline{\dot{q}}. \quad (13)$$

The total dynamics is now written as:

$$\begin{bmatrix} \underline{M}(\underline{q}) & -\underline{W}(\underline{q}, t) \\ \underline{W}^T(\underline{q}, t) & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{\ddot{q}} \\ \underline{\lambda} \end{bmatrix} + \begin{bmatrix} \underline{H}(\underline{q}, \underline{\dot{q}}) \\ \underline{\bar{w}}(\underline{q}, \underline{\dot{q}}, t) \end{bmatrix} = \begin{bmatrix} \underline{S}(\underline{q})\underline{\tau} \\ \underline{0} \end{bmatrix}. \quad (14)$$

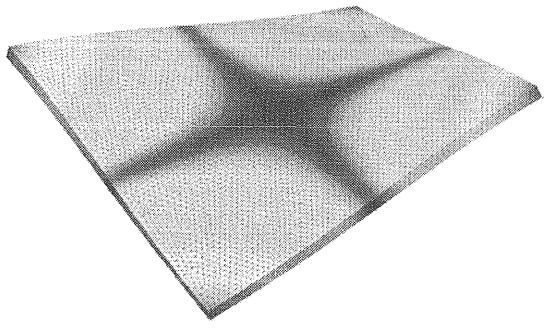

Fig. 4. First eigenmode of the plate using FEA

As the goal of the model is to predict the forces and torques on the platform, now an expression is derived for the Lagrange multipliers. For a certain trajectory of the platform and manipulator, the Lagrange multipliers namely contain the forces and torques which keep the platform balanced. From (14) an expression for the generalised accelerations, $\ddot{\underline{q}}$, is now derived as:

$$\ddot{\underline{q}} = \underline{M}^{-1}(\underline{q}) \left(\underline{S}(\underline{q})\underline{\tau} - \underline{H}(\underline{q}, \dot{\underline{q}}) + \underline{W}(\underline{q}, t)\underline{\lambda} \right). \quad (15)$$

Using this expression in (9) and solving for $\underline{\lambda}$ results in:

$$\begin{aligned} \underline{\lambda} = &\left(\underline{W}^T(\underline{q}, t)\underline{M}^{-1}(\underline{q})\underline{W}(\underline{q}, t) \right)^{-1} \\ &\left(\underline{W}^T(\underline{q}, t)\underline{M}^{-1}(\underline{q}) \left(\underline{H}(\underline{q}, \dot{\underline{q}}) - \underline{S}(\underline{q})\underline{\tau} \right) \right). \end{aligned} \quad (16)$$

So an expression for $\underline{\lambda}$ is now available in terms of $\underline{q}$, $\dot{\underline{q}}$, τ and t.

IV. Model validation

A. Finite element analysis

The platform is modelled as a rigid body. This assumption is only valid up to a certain frequency. Therefore, a finite element analysis (FEA) of the isolated plate is performed. Because both the linear actuators and the bearing support add stiffness to the plate, the isolated plate is, therefore, a worst case scenario. The baseplate is modelled as a 300×300×10 mm aluminium plate. The first eigenmode of the plate is predicted to occur at 350 Hz (see Fig. 4). Because the frequency range in which the manipulator will stay below 80 Hz, the plate can be considered rigid.

B. Experimental setup

A real manipulator is built and used for verification purposes. It is placed on a 6 DOF sensor (see Fig. 5) which can measure forces and torques in three directions, respectively. As a result of placing the platform on a very stiff sensor its location is considered fixed to the world. The measured reaction forces and torques are equal in magnitude to the forces and torques necessary to stabilise the platform in case it is floating.

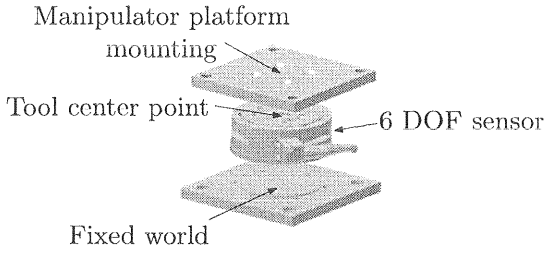

Fig. 5. Mounting of 6 DOF sensor

C. Impulse response

Impulse response measurements were performed in order to characterise the eigenfrequencies in the manipulator. Thereto an impulse hammer, with built-in force sensor, is used to excite the platform, while the reaction forces and torques on the platform are measured with the 6 DOF sensor. The measurements were performed with the beam in both outer positions as well as in the centre. From the impulse responses (see Figs. 6-11, note: all figures share the same legend) we can now conclude most dynamics appear above 80 Hz.

D. Transfer functions

The manipulator beam is supposed to be rigid. A rigid beam allows the total force produced by the two linear actuators to be modelled as a single force in the middle of the beam. One way to validate this assumption is to measure the transfer functions from one linear actuator to itself and its cross terms from the other linear actuator and compare the results. The control scheme of the legs of the manipulator is shown in Fig. 12. So each manipulator leg has its own controller, C, which is a 10 Hz low bandwidth PD-controller. The controller is implemented in such a way that no crosstalk effects are taken into account. The transfer function of each linear actuator as well as the transfer function of the cross terms is measured using noise injection after the controller. Noise is injected at w_1 and w_2 and measured at v_1 and v_2. Now the transfer function is computed as:

$$H_{ij} = \frac{v_j}{w_i} = \begin{cases} \dfrac{1}{1 + C_i P_{ij}} & \text{for } i = j \\ \dfrac{-C_j P_{ij}}{1 + C_j P_{jj}} & \text{for } i \neq j \end{cases}. \quad (17)$$

Using these transfer function definitions the plant transfer functions are computed as:

$$P_{ij} = \begin{cases} \dfrac{1 - H_{ij}}{H_{ij} C_i} & \text{for } i = j \\ -\dfrac{H_{ij}(1 + C_j P_{jj})}{C_j} & \text{for } i \neq j \end{cases}. \quad (18)$$

Measurement data is collected by making the manipulator follow a simple trajectory while injecting band limited white noise. The transfer functions (see Figs. 13 and 14) are now computed using an averaging procedure. In these figures only

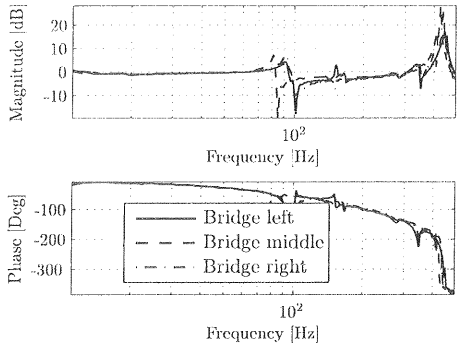

Fig. 6. Impulse response of force in x-direction

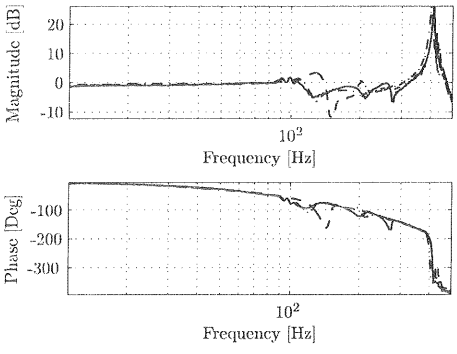

Fig. 7. Impulse response of force in y-direction

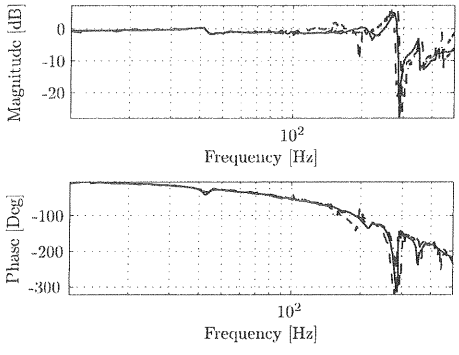

Fig. 8. Impulse response of force in z-direction

the results are shown for one leg, P_{11}, as the other leg has similar results. It becomes clear that the cross term, P_{21}, is almost identical to the direct transfer function. From this we can conclude that a very stiff connection must be present between the two legs. Therefore, the assumption of considering the beam rigid is valid. Note that this also shows that a MIMO controller, which takes care of the cross terms, might achieve much better control results.

E. Friction force

The motor currents as well as the position of both linear actuators are measured. From this data the friction force is estimated, using the fact that for a constant velocity the acceleration is zero. Therefore, the motor currents at constant

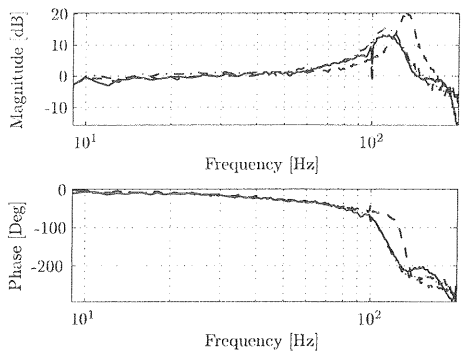

Fig. 9. Impulse response of torque about x-axis

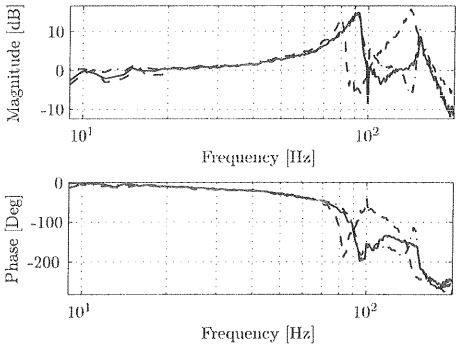

Fig. 10. Impulse response of torque about y-axis

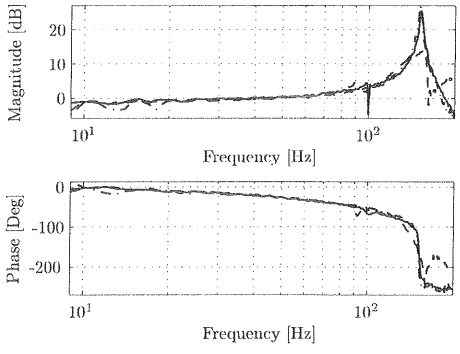

Fig. 11. Impulse response of torque about z-axis

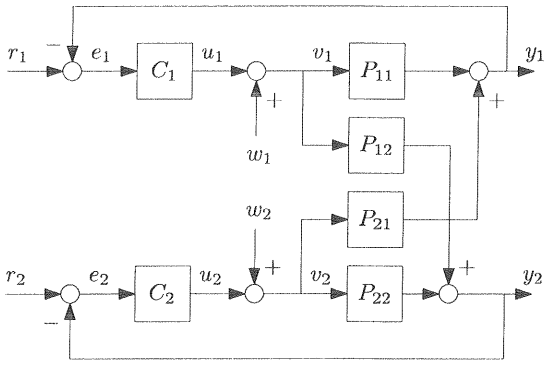

Fig. 12. Control scheme in manipulator

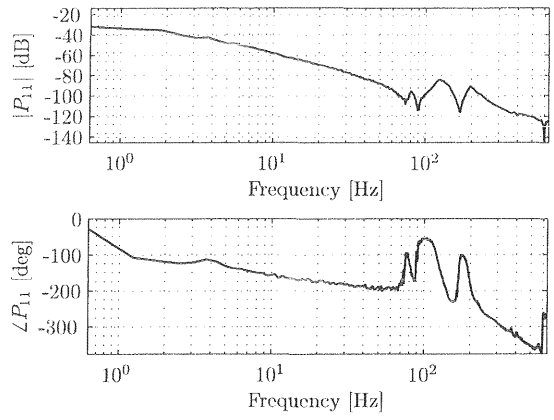

Fig. 13. Measured cross terms in manipulator

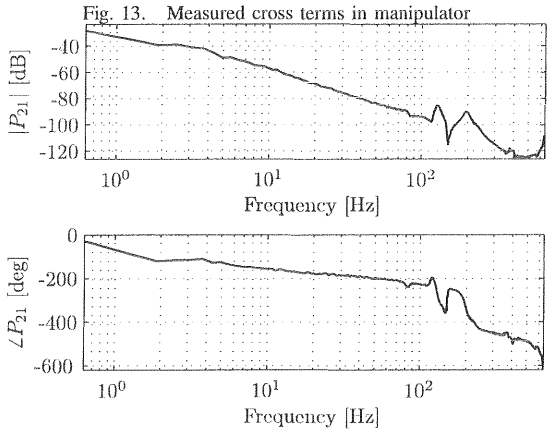

Fig. 14. Measured cross terms in manipulator

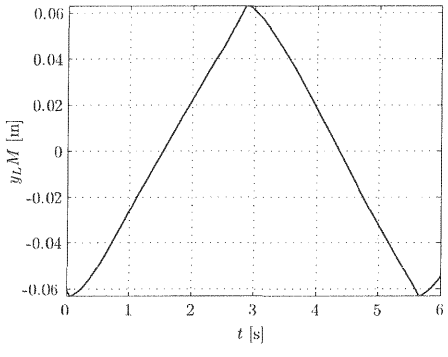

Fig. 15. Constant velocity trajectory

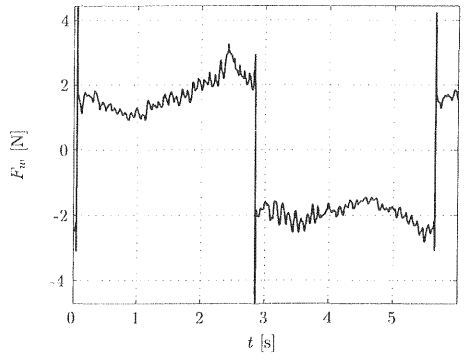

Fig. 16. Friction force in linear actuators

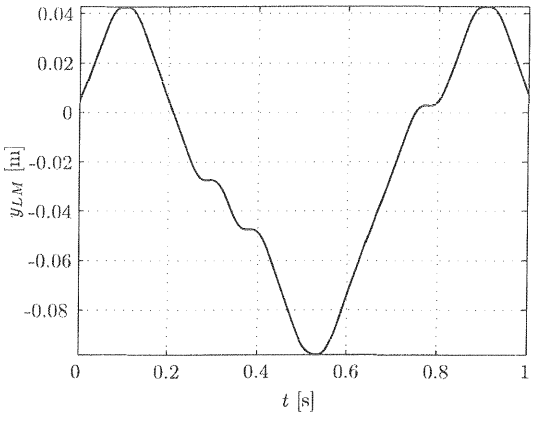

Fig. 17. Movement profile of the beam

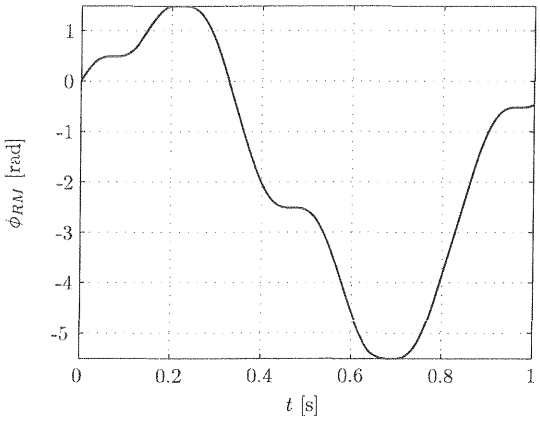

Fig. 18. Movement profile of the arm

velocity should be zero if no friction is present. In Fig. 15 the constant velocity trajectory is plotted and in Fig. 16 the reconstructed friction from the motor currents is shown. The experiment was repeated for several speeds, but no significant velocity dependence was found, therefore the friction can be estimated as Coulomb friction of approximately 2 N, which is about ten percent of the total available actuator force.

F. Inverse dynamic analysis

Now the model assumptions are checked, the model itself is validated. A movement profile (see Figs. 17 and 18) is followed by the manipulator and measurement data is collected. The movement profile is used as an constraint input for the model and the constraint forces are computed.

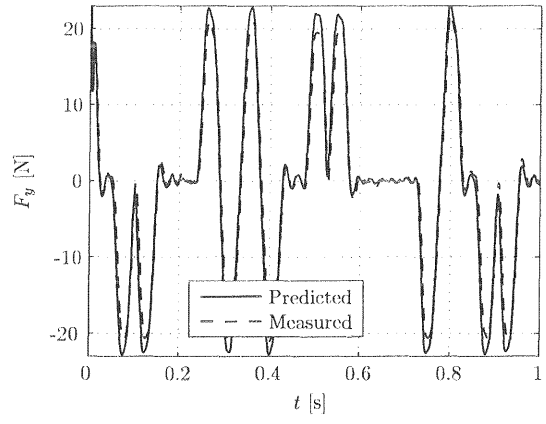

Fig. 19. Force on platform in $\vec{e}_0^2$-direction

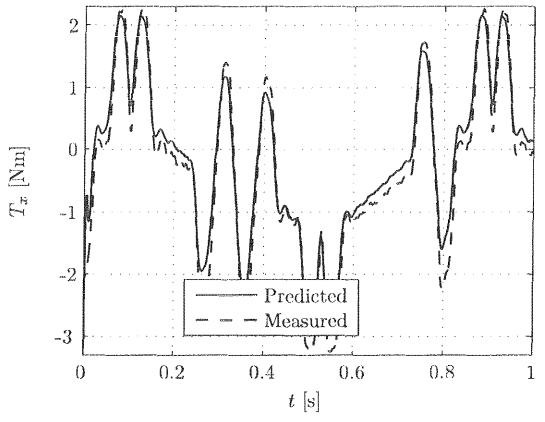

Fig. 20. Torque on platform about $\vec{e}_0^1$

V. Results

In Figs. 19-21 the results of the simulation as well as the measured forces and torques are shown. The used controller for the linear motor consists of a lead-lag up to 60 Hz and has a lowpass filter at 70 Hz in order to suppress following unmodelled dynamics. The measured forces and torques are filtered with a 4^{th} order Butterworth filter with a cut-off frequency of 80 Hz. F_y and T_x are predicted quite well, however some offsets are present. This offset is due to calibration problems of the 6 DOF sensor. The predicted torque about $\vec{e}_0^3$ shows globally the same peaks as the measured force, but is not very accurate. The reasons for this deviation are the existence of a noise component in the unfiltered signal, which has a much larger amplitude than the signal itself, as well as the previously mentioned calibration problem of the 6 DOF sensor. Therefore further fine-tuning of the model parameters is delayed until the sensor is better calibrated.

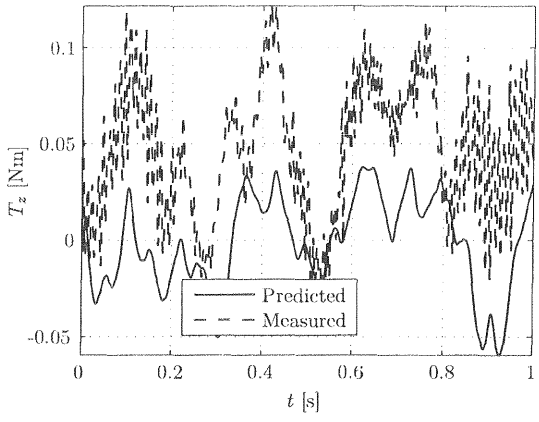

Fig. 21. Torque on platform about $\vec{e}_0^3$

VI. Conclusions and Recommendations

A multi-body model was derived for a manipulator on a floating platform. It will be used for predicting disturbance forces and torques on the platform. A manipulator was built and placed on a 6 DOF sensor for verification purposes. The multi-body model was verified with measurements. The rigid body behaviour of the manipulator is predicted very well, however some unmodelled dynamics, make the model unreliable for frequencies above 80 Hz. Better results could be achieved by recalibrating the 6 DOF sensor, which would allow more reliable measurements, which in turn would allow better fine-tuning of the model parameters. Finally, further investigation of the crosstalk between the linear actuators, and implementation of a MIMO controller could lead to better control results.

Acknowledgements

The author would in the first place like to thank his supervisor, Jeroen de Boeij, for the great amount of help and good advise during the whole project. A lot of thanks also go to Marijn Uyt de Willigen for his assistance with the experimental setup. Finally, thanks go to Elena Lomonova for her very useful comments and suggestions for this article.

References

[1] Compter, J.C. *Electro-Dynamic Planar Motor* Precision Engineering, Vol 28, Issue 2, pp. 171-180, April 2004.

[2] Jansen, J.W., Lierop, C.M.M. van, Lomonova, E. and Vandenput, A.J.A., *Moving-Magnet Planar Actuator with Integrated Active Magnetic Bearing*, ASPE 21st annual meeting, Monterey CA, October 2006, CDROM.

[3] Boeij, J. de, Lomonova, E., and Vandenput, A.J.A., *Modeling Ironless Permanent-Magnet Planar Actuator Structures*, IEEE Trans. on Magnetics, Vol. 42, No. 8, pp. 2009-2016, August 2006.

[4] Kraker, B. de and Campen, D.H. van, *Mechanical Vibrations*, Maastricht, The Netherlands: Shaker Publishing, 2001.

[5] Wouw, N. van de, *Multibody Dynamics - Lecture notes*, Eindhoven, The Netherlands, 2003.

[6] Lyshevski, S. E., *Electromechanical Systems, Electric Machines, and Applied Mechatronics*, Boca Raton: CRC Press, 2000.

APPENDIX A
ROTATION MATRICES

A rotation matrix is a function of the rotation parameters of a body and allows the transformation from one coordinate system to another. A total of three basic rotations exists, one about each axis. If a body is rotated using the Tait-Bryant sequence, it is rotated first about the local $\vec{e}_1$-axis, followed by a rotation about the local $\vec{e}_2$-axis and finally a rotation about $\vec{e}_3$ is performed. Therefore the total rotation sequence can be described by an initial coordinate system, two intermediate systems and a final coordinate orientation. The rotation about the $\vec{e}_1^0$-axis is written as:

$$\underline{\vec{e}}^1 = \underline{A}^{10}\underline{\vec{e}}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \underline{\vec{e}}^0. \tag{19}$$

About the $\vec{e}_2^1$-axis:

$$\underline{\vec{e}}^2 = \underline{A}^{21}\underline{\vec{e}}^1 = \begin{bmatrix} \cos(\psi) & 0 & -\sin(\psi) \\ 0 & 1 & 0 \\ \sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \underline{\vec{e}}^1. \tag{20}$$

And finally about the $\vec{e}_3^2$-axis:

$$\underline{\vec{e}}^3 = \underline{A}^{32}\underline{\vec{e}}^2 = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{\vec{e}}^2. \tag{21}$$

So the rotation matrix, which describes the complete Tait-Bryant sequence can be constructed as:

$$\begin{aligned} \underline{\vec{e}}^3 = \underline{A}^{32}\underline{A}^{21}\underline{A}^{10} &= \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\psi) & 0 & -\sin(\psi) \\ 0 & 1 & 0 \\ \sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \underline{\vec{e}}^0 \\ &= \begin{bmatrix} \cos(\psi)\cos(\phi) & \cos(\theta)\sin(\phi) + \sin(\theta)\sin(\psi)\cos(\phi) & \sin(\theta)\sin(\phi) - \cos(\theta)\sin(\psi)\cos(\phi) \\ -\cos(\psi)\sin(\phi) & \cos(\theta)\cos(\phi) - \sin(\theta)\sin(\psi)\sin(\phi) & \sin(\theta)\cos(\phi) + \cos(\theta)\sin(\psi)\sin(\phi) \\ \sin(\psi) & -\sin(\theta)\cos(\psi) & \cos(\theta)\cos(\psi) \end{bmatrix} \underline{\vec{e}}^0 \end{aligned}. \tag{22}$$

So it is simply a multiplication of the rotation matrices related to respectively a rotation about the local $\vec{e}_3$, $\vec{e}_2$ and $\vec{e}_1$ axis.

APPENDIX B
FULL MULTI-BODY MODEL DERIVATION

A. Positions and orientations

The column of generalised coordinates for the complete multi-body derivation is chosen as:

$$\underline{q} = \begin{bmatrix} x_p & y_p & z_p & \theta_p & \psi_p & \phi_p & y_{LM} & \phi_{RM} \end{bmatrix}^T, \tag{23}$$

which is a slightly different choice than in the main article. Here x_p, y_p and z_p denote the position of the geometric centre of the platform instead of the previously used centre of mass of the platform (which lies slightly higher than the geometric centre). Also the rotation parameters, θ_p, ψ_p and ϕ_p are now directly related to the geometric centre of the platform, instead of the centre of mass. The realised rotation is exactly the same, but the notation, however, is more consequent. The choice for x_p, y_p and z_p is very straightforward, as this position is exactly known, in contrast to, the centre of mass, which is estimated. The centre of mass of the platform is now rewritten as:

$$\begin{aligned} \vec{r}_{CM_1} &= \vec{r}_p + \vec{r}_{1_{lcl}} \\ &= \begin{bmatrix} x_p & y_p & z_p \end{bmatrix} \underline{\vec{e}}^0 + \begin{bmatrix} x_{1_{lcl}} & y_{1_{lcl}} & z_{1_{lcl}} \end{bmatrix} \underline{\vec{e}}^{1'} \end{aligned} \tag{24}$$

where $\vec{r}_p$ is the position of the geometric centre of the platform and $\vec{r}_{1_{lcl}}$ is a body fixed vector (see Fig. 22), which describes the position of the centre of mass of the platform with respect to its geometric centre. The centre of mass of body 2 and 3 can be rewritten in a similar way, which results in:

$$\begin{aligned} \vec{r}_{CM_2} &= \vec{r}_p + \vec{r}_{21} + \begin{bmatrix} 0 & y_{LM} & 0 \end{bmatrix} \underline{\vec{e}}^1 + \vec{r}_{2_{lcl}} \\ &= \begin{bmatrix} x_p & y_p & z_p \end{bmatrix} \underline{\vec{e}}^0 + \begin{bmatrix} x_{21} & y_{21} + y_{LM} & z_{21} \end{bmatrix} \underline{\vec{e}}^1 + \begin{bmatrix} x_{2_{lcl}} & y_{2_{lcl}} & z_{2_{lcl}} \end{bmatrix} \underline{\vec{e}}^{2'} \end{aligned} \tag{25}$$

and

$$\begin{aligned} \vec{r}_{CM_3} &= \vec{r}_p + \vec{r}_{21} + \begin{bmatrix} 0 & y_{LM} & 0 \end{bmatrix} \underline{\vec{e}}^1 + \vec{r}_{32} + \vec{r}_{3_{lcl}} \\ &= \begin{bmatrix} x_p & y_p & z_p \end{bmatrix} \underline{\vec{e}}^0 + \begin{bmatrix} x_{21} & y_{21} + y_{LM} & z_{21} \end{bmatrix} \underline{\vec{e}}^1 + \begin{bmatrix} x_{32} & y_{32} & z_{32} \end{bmatrix} \underline{\vec{e}}^2 + \begin{bmatrix} x_{3_{lcl}} & y_{3_{lcl}} & z_{3_{lcl}} \end{bmatrix} \underline{\vec{e}}^{3'} \end{aligned} \tag{26}$$

where $\vec{r}_{ji}$ are body fixed vectors from body i to body j. The rotation matrices are also rewritten in terms of $\underline{q}$:

$$\underline{A}^{10}(\theta_1, \psi_1, \phi_1) \Rightarrow \underline{A}^{10}(\theta_p, \psi_p, \phi_p) \tag{27}$$

$$\underline{A}^{20}(\theta_2, \psi_2, \phi_2) \Rightarrow \underline{A}^{20}(\theta_p, \psi_p, \phi_p) \tag{28}$$

$$\underline{A}^{30}(\theta_3, \psi_3, \phi_3) \Rightarrow \underline{A}^{30}(\theta_p, \psi_p, \phi_p + \phi_{RM}) \tag{29}$$

B. Angular velocity

The angular velocity, ${}^{ji}\vec{\omega}$, of coordinate system $\underline{\vec{e}}^j$ with respect to $\underline{\vec{e}}^i$, is determined using the additive property. The angular velocity of a body can be written as the sum of the angular velocities caused by its rotations. So for body 1 this results in:

$$\begin{aligned} {}^{10}\vec{\omega} &= \dot{\theta}_p \vec{e}_1^0 + \dot{\psi}_p \vec{e}_2^{\alpha} + \dot{\phi}_p \vec{e}_3^{\beta} = \\ &= \begin{bmatrix} \dot{\theta}_p + \dot{\phi}_p \sin(\psi_p) & \dot{\psi}_p \cos(\theta_p) - \dot{\phi}_p \cos(\psi_p) \sin(\theta_p) & \dot{\phi}_p \cos(\psi_p) \cos(\theta_p) + \dot{\psi}_p \sin(\theta_p) \end{bmatrix} \underline{\vec{e}}^{0'} \end{aligned} \tag{30}$$

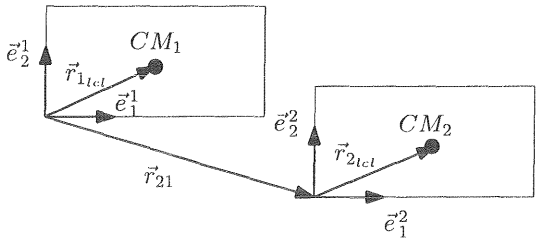

Fig. 22. Body fixed vectors

where $\underline{\vec{e}}^{\alpha}$ and $\underline{\vec{e}}^{\beta}$ denote intermediate coordinate systems. Body 2 can not rotate with respect to body 1, therefore, the angular velocity is the same, so:

$$^{20}\vec{\omega} = {}^{10}\vec{\omega}. \tag{31}$$

And finally body 3 rotates about $\vec{e}_3^2$ with an angle ϕ_{RM} so:

$$\begin{aligned} {}^{30}\vec{\omega} &= \dot{\theta}_p\vec{e}_1^0 + \dot{\psi}_p\vec{e}_2^{\alpha} + \dot{\phi}_p\vec{e}_3^{\beta} + \dot{\phi}_{RM}\vec{e}_3^2 = \\ &= \begin{bmatrix} \dot{\theta}_p + (\dot{\phi}_p + \dot{\phi}_{RM})\sin(\psi_p) \\ \dot{\psi}_p\cos(\theta_p) - (\dot{\phi}_p + \dot{\phi}_{RM})\cos(\psi_p)\sin(\theta_p) \\ (\dot{\phi}_p + \dot{\phi}_{RM})\cos(\psi_p)\cos(\theta_p) + \dot{\psi}_p\sin(\theta_p) \end{bmatrix}^T \underline{\vec{e}}^0. \end{aligned} \tag{32}$$

C. Kinetic and potential energy

Now all terms in (5) are known the kinetic energy, T, can be computed. Also all terms for the potential energy, V, are known. The resulting equations for the complete 3D-model including all degrees of freedom, however, are very elaborate and are therefore not written down here. In Appendix C a simplified 2D-model is described, including the corresponding expressions for T and V.

D. Constraints

Each degree of freedom has a constraint related to it. The constraints on the platform are very straightforward, as the platform can not move, nor rotate. Therefore the velocities, $\dot{x}_p$, $\dot{y}_p$ and $\dot{z}_p$, as well as the angular velocities, $\dot{\theta}_p$, $\dot{\psi}_p$, $\dot{\phi}_p$, are zero. The position of the beam, y_{LM}, follows a certain trajectory, $u_1(t)$, therefore the velocity constraint on the beam can be written as $\dot{y}_{LM} - \dot{u}_1(t) = 0$. Furthermore, the angle of the rotary motor, ϕ_{RM}, is defined by $u_2(t)$, which results in the constraint $\dot{\phi}_{RM} - \dot{u}_2(t)$. So now the column of constraints is written as:

$$\underline{h}_{nh} = \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{z}_p \\ \dot{\theta}_p \\ \dot{\psi}_p \\ \dot{\phi}_p \\ \dot{y}_{LM} - \dot{u}_1(t) \\ \dot{\phi}_{RM} - \dot{u}_2(t) \end{bmatrix} = \underline{0}, \tag{33}$$

or using the format in (9) the constraints are written as:

$$\underline{W}^T(\underline{q},t)\dot{\underline{q}} + \underline{\tilde{w}}(\underline{q},t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{z}_p \\ \dot{\theta}_p \\ \dot{\psi}_p \\ \dot{\phi}_p \\ \dot{y}_{LM} \\ \dot{\phi}_{RM} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\dot{u}_1(t) \\ -\dot{u}_2(t) \end{bmatrix} = \underline{0}. \tag{34}$$

All expressions in the equations of Lagrange are now known and the constraint forces can be computed.

APPENDIX C
2D-MODEL

Because the full 3D-model results in too elaborate expressions, a 2D-model is introduced here to give the reader an idea of the resulting equations. The 2D-model consists of a platform and a beam. The rotary arm is removed in this case. The platform can move in two directions and rotate about the third axis. The beam can still not rotate with respect to platform, but can only move in $\vec{e}_1^1$ direction. Therefore in this case, a column of generalised coordinates is:

$$\underline{q} = \begin{bmatrix} y_p & z_p & \theta_p & y_{LM} \end{bmatrix}^T. \tag{35}$$

The orientation of $\underline{\vec{e}}^1$ with respect to $\underline{\vec{e}}^0$ is now only determined by θ_p and therefore:

$$\underline{\vec{e}}^1 = \underline{A}^{10}(\theta_p)\underline{\vec{e}}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_p) & \sin(\theta_p) \\ 0 & -\sin(\theta_p) & \cos(\theta_p) \end{bmatrix} \underline{\vec{e}}^0. \tag{36}$$

The centres of mass and geometric centre of the bodies are chosen to coincide in order to simplify the resulting equations. Therefore the position of body 1 can be written as:

$$\vec{r}_{CM_1} = \begin{bmatrix} 0 & y_p & z_p \end{bmatrix} \underline{\vec{e}}^0, \tag{37}$$

and the position of body 2:

$$\begin{aligned} \vec{r}_{CM_2} &= \begin{bmatrix} 0 & y_p & z_p \end{bmatrix} \underline{\vec{e}}^0 + \begin{bmatrix} 0 & y_{LM} & z_{21} \end{bmatrix} \underline{\vec{e}}^1 = \begin{bmatrix} 0 & y_p & z_p \end{bmatrix} \underline{\vec{e}}^0 + \begin{bmatrix} 0 & y_{LM} & z_{21} \end{bmatrix} \underline{A}^{10} \underline{\vec{e}}^0 \\ &= \begin{bmatrix} 0 & y_p + y_{LM}\cos(\theta_p) - z_{21}\sin(\theta_p) & z_p + y_{LM}\sin(\theta_p) + z_{21}\cos(\theta_p) \end{bmatrix} \underline{\vec{e}}^0 \end{aligned}. \tag{38}$$

The expressions for the angular velocities are also much simpler now, because only one angle is involved, therefore:

$${}^{10}\vec{\omega} = {}^{20}\vec{\omega} = \begin{bmatrix} \dot{\theta}_p & 0 & 0 \end{bmatrix} \underline{\vec{e}}^0. \tag{39}$$

Now the kinetic energy expression, T, is computed:

$$\begin{aligned} T = \frac{1}{2} \Big(& m_1 \left(\dot{y}_p^2 + \dot{z}_p^2 \right) + m_2 \Big(\Big(\dot{y}_p + \dot{y}_{LM}\cos(\theta_p) + \dot{\theta}_p \left(-y_{LM}\sin(\theta_p) - z_{21}\cos(\theta_p) \right) \Big)^2 \\ & + \Big(\dot{z}_p + \dot{y}_{LM}\sin(\theta_p) + \dot{\theta}_p \left(y_{LM}\cos(\theta_p) - z_{21}\sin(\theta_p) \right) \Big)^2 \Big) + \left(J_1 + J_2 \right) \dot{\theta}_p^2 \Big) \end{aligned}. \tag{40}$$

And the potential energy, V:

$$V = -g \left(m_1 z_p + m_2 \left(z_p + y_{LM}\sin(\theta_p) + z_{21}\cos(\theta_p) \right) \right). \tag{41}$$

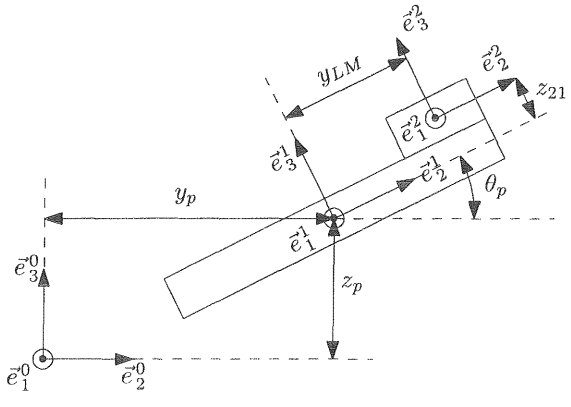

Fig. 23. 2D-model

In this 2D-model also the number of constraint equations is only four. Namely:

$$\underline{h}_{nh} = \begin{bmatrix} \dot{y}_p \\ \dot{z}_p \\ \dot{\theta}_p \\ \dot{y}_{LM} - \dot{u}_1(t) \end{bmatrix} = \underline{0}, \tag{42}$$

or again using the format in (9) the constraints are written as:

$$\underline{W}^T(\underline{q},t)\dot{\underline{q}} + \underline{\tilde{w}}(\underline{q},t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_p \\ \dot{z}_p \\ \dot{\theta}_p \\ \dot{y}_{LM} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\dot{u}_1(t) \end{bmatrix} = \underline{0}. \tag{43}$$

After differentiating and combining all energy equations in (10), and rewriting the result together with the constraints in the format of (14), the following symmetric, mass matrix, $\underline{M}(\underline{q})$, is obtained:

$$\underline{M}(\underline{q}) = \begin{bmatrix} m_1 + m_2 & 0 & m_2\cos(\theta_p) & -m_2\left(y_{LM}\sin(\theta_p) + z_{21}\cos(\theta_p)\right) \\ \cdots & m_1 + m_2 & m_2\sin(\theta_p) & m_2\left(y_{LM}\cos(\theta_p) - z_{21}\sin(\theta_p)\right) \\ \cdots & \cdots & m_2 & -z_{21}m_2 \\ \cdots & \cdots & \cdots & J_1 + J_2 + m_2\left(y_{LM}^2 + z_{21}^2\right) \end{bmatrix}, \tag{44}$$

where the symmetrical entries are skipped. And for $\underline{H}(\underline{q},\dot{\underline{q}})$ is found:

$$\underline{H}(\underline{q},\dot{\underline{q}}) = \begin{bmatrix} -m_2\dot{\theta}_p\left(\left(2\dot{y}_{LM} - z_{21}\dot{\theta}_p\right)\sin(\theta_p) + \dot{\theta}_p y_{LM}\cos(\theta_p)\right) \\ m_2\dot{\theta}_p\left(2\dot{y}_{LM}\cos(\theta_p) - \dot{\theta}_p\left(y_{LM}\sin(\theta_p) + z_{21}\cos(\theta_p)\right)\right) - g\left(m_1 + m_2\right) \\ -m_2\left(\dot{\theta}_p^2 y_{LM} + g\sin(\theta_p)\right) \\ m_2\left(2\dot{y}_{LM}\dot{\theta}_p y_{LM} + g\left(z_{21}\sin(\theta_p) - y_{LM}\cos(\theta_p)\right)\right) \end{bmatrix}. \tag{45}$$

Note that $\underline{S}(\underline{q})\underline{\tau}$ in this case is $\underline{0}$ as there are no external forces defined.

APPENDIX D
MATLAB: EQUATIONCOMPUTATION3D.M

```
% Clear screen, memory and close figures
clear all
close all
clc

% Define the used symbolic expressions
syms x_p y_p z_p theta_p psi_p phi_p y_LM phi_RM
syms x_pdot y_pdot z_pdot theta_pdot psi_pdot phi_pdot...
    y_LMdot phi_RMdot
syms x_pddot y_pddot z_pddot theta_pddot psi_pddot...
    phi_pddot y_LMddot phi_RMddot

% Load used parameters from parameterfile
loadparams

% Make tfuns a global variable
global tfuns
tfuns = [x_p y_p z_p theta_p psi_p phi_p y_LM phi_RM];

% Define the column of generalised coordinates and derivatives
q = [x_p y_p z_p theta_p psi_p phi_p y_LM phi_RM].';
qdot = tdiff(q);
qddot = tdiff(qdot);

% Define rotation matrices
A01alpha_0 = [1 0 0
    0 cos(theta_p) sin(theta_p)
    0 -sin(theta_p) cos(theta_p)];
A01beta_01alpha = [cos(psi_p) 0 -sin(psi_p)
    0 1 0
    sin(psi_p) 0 cos(psi_p)];
A1_01beta = [cos(phi_p) sin(phi_p) 0
    -sin(phi_p) cos(phi_p) 0
    0 0 1];
A01beta_0 = A01beta_01alpha*A01alpha_0;
A1_0 = A1_01beta*A01beta_01alpha*A01alpha_0;
A2_0 = A1_0;
A3_0 = subs(A1_0, phi_p, phi_p+phi_RM);

% Define position vectors and derivatives
r_CM_1 = ([x_p y_p z_p] + [x_lcl_1 y_lcl_1 z_lcl_1]*A1_0).';
r_CM_2 = ([x_p y_p z_p] + [x_21 y_21+y_LM z_21]*A1_0 + ...
    [x_lcl_2 y_lcl_2 z_lcl_2]*A2_0).';
r_CM_3 = ([x_p y_p z_p] + [x_21 y_21+y_LM z_21]*A1_0 + ...
    [x_32 y_32 z_32]*A2_0 + [x_lcl_3 y_lcl_3 z_lcl_3]*A3_0).';
r_CM_1dot = simple(tdiff(r_CM_1));
r_CM_2dot = simple(tdiff(r_CM_2));
r_CM_3dot = simple(tdiff(r_CM_3));

% Define angular velocity
omega_10_0 = simple([theta_pdot 0 0] + [0 psi_pdot 0]*A01alpha_0 + ...
    [0 0 phi_pdot]*A01beta_0).';
omega_20_0 = omega_10_0;
omega_30_0 = simple([theta_pdot 0 0] + [0 psi_pdot 0]*A01alpha_0 + ...
    [0 0 phi_pdot+phi_RMdot]*A1_0).';

% Compute kinetic energy
H_CM1_0 = simple(A1_0.'*J_CM1_1*A1_0*omega_10_0);
H_CM2_0 = simple(A2_0.'*J_CM2_2*A2_0*omega_20_0);
H_CM3_0 = simple(A3_0.'*J_CM3_3*A3_0*omega_30_0);
T = 0;
T = T+.5*m_1*r_CM_1dot.'*r_CM_1dot+.5*omega_10_0.'*H_CM1_0;
T = T+.5*m_2*r_CM_2dot.'*r_CM_2dot+.5*omega_20_0.'*H_CM2_0;
T = T+.5*m_3*r_CM_3dot.'*r_CM_3dot+.5*omega_30_0.'*H_CM3_0;
```

```
T = simple(T);

% Compute potential energy
V = 0;
V = V-m_1*[0 0 g]*r_CM_1;
V = V-m_2*[0 0 g]*r_CM_2;
V = V-m_3*[0 0 g]*r_CM_3;
V = simple(V);

% Compute several derivatives
ddtT_qdot = simple(tdiff(rowdiff(T, qdot)));
T_q = simple(rowdiff(T, q));
V_q = simple(rowdiff(V, q));

% Set Qnc to zero as no external forces are used in this case
Qnc = zeros(8,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - This part is a demonstration of how %
% external forces could be included - %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Define tau
% tau = [F_xext F_yext F_zext].';
%
% % Define forces and the location of action
% r{1} = [x_p y_p z_p];
% F{1} = [F_xext 0 0]*A1_0;
%
% r{2} = [z_p y_p z_p];
% F{2} = [0 F_yext 0]*A1_0;
%
% r{3} = [x_p y_p z_p];
% F{3} = [0 0 F_zext]*A1_0;
%
% % Compute Qnc
% Qnc = sym(0)*zeros(length(q), 1);
% for i = 1:length(F)
%
%     Qnc = Qnc + rowdiff(r{i}, q).'*F{i}.';
%
% end

% Define the constraint on y_LM and phi_RM
syms u_1 u_1dot u_2 u_2dot

h = [];
h = [h, x_pdot];
h = [h, y_pdot];
h = [h, z_pdot];
h = [h, theta_pdot];
h = [h, psi_pdot];
h = [h, phi_pdot];
h = [h, y_LMdot-u_1dot];
h = [h, phi_RMdot-u_2dot];

% Compute W
W = sym(0)*ones(length(qdot), length(h));
for i = 1:length(h)

    W(:, i) = rowdiff(h(i), qdot).';

end

% Compute wtilde
wtilde = simple(h.'-W.'*qdot);

% - Here everything is rewritten! -
```

```
% Lefthand is the lefthand part of the Lagrange equations
lefthand = simple(ddtT_qdot - T_q + V_q).';

% Compute M
M = sym(0)*ones(length(lefthand));
for i = 1:length(lefthand)

    for j = 1:length(lefthand)

        eval(['M(' num2str(i) ',' num2str(j) ...
            ') = findddot(lefthand(' num2str(i) ...
            '), qddot(' num2str(j) '));']);

    end

end
M = simple(M);

% Substract M from lefthand part of equations
eqns = lefthand - Qnc;
eqnsrest = simple(eqns - M*qddot);

% Set tau and S to 0 in this case
tau = sym(0);
S = sym(0)*ones(length(eqnsrest), length(tau));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - Note that in case external forces are included this part should %
% be used instead -                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% eqns = lefthand - Qnc;
% eqnsrest = simple(eqns - M*qddot);
%
% S = sym(0)*ones(length(eqnsrest), length(tau));
% for i = 1:length(eqnsrest)
%
%     for j = 1:length(tau)
%
%         eval(['S(' num2str(i) ',' num2str(j) ') = ...
%   -findtau(eqnsrest(' num2str(i) '), tau(' num2str(j) '));']);
%
%     end
%
% end

% Compute H
H = simple(eqnsrest + S*tau);
```

APPENDIX E

MATLAB: EQUATIONCOMPUTATION2D.M

```
% Clear screen, memory and close figures
clear all
close all
clc

% Define the used symbolic expressions
syms y_p z_p y_LM theta_p
syms y_pdot z_pdot y_LMdot theta_pdot
syms y_pddot z_pddot y_LMddot theta_pddot
syms J_1 J_2 m_1 m_2
syms g z_21

% Make tfuns a global variable
global tfuns
tfuns = [y_p z_p y_LM theta_p];

% Define the column of generalised coordinates and derivatives
q = [y_p z_p y_LM theta_p].';
qdot = tdiff(q);
qddot = tdiff(qdot);

% Define rotation matrices
A1_0 = [1 0 0
        0 cos(theta_p) sin(theta_p)
        0 -sin(theta_p) cos(theta_p)];
A2_0 = A1_0;

% Define position vectors and derivatives
r_CM_1 = ([0 y_p z_p]).';
r_CM_2 = ([0 y_p z_p]+[0 y_LM z_21]*A1_0).';
r_CM_1dot = simple(tdiff(r_CM_1));
r_CM_2dot = simple(tdiff(r_CM_2));

% Define angular velocity
omega_10_0 = [theta_pdot 0 0].';
omega_20_0 = [theta_pdot 0 0].';

% Define moments of inertia
J_CM1_1 = [J_1 0 0
    0 0 0
    0 0 0];
J_CM2_2 = [J_2 0 0
    0 0 0
    0 0 0];

% Compute kinetic energy
H_CM1_0 = simple(A1_0.'*J_CM1_1*A1_0*omega_10_0);
H_CM2_0 = simple(A2_0.'*J_CM2_2*A2_0*omega_20_0);
T = 0;
T = T+.5*m_1*r_CM_1dot.'*r_CM_1dot+.5*omega_10_0.'*H_CM1_0;
T = T+.5*m_2*r_CM_2dot.'*r_CM_2dot+.5*omega_20_0.'*H_CM2_0;
T = simple(T);

% Compute potential energy
V = 0;
V = V-m_1*[0 0 g]*r_CM_1;
V = V-m_2*[0 0 g]*r_CM_2;
V = simple(V);

% Compute several derivatives
ddtT_qdot = simple(tdiff(rowdiff(T, qdot)));
T_q = simple(rowdiff(T, q));
V_q = simple(rowdiff(V, q));

```

```
% Set Qnc to zero as no external forces are used in this case
Qnc = zeros(4,1);

% Define the constraint on y_LM
syms u_1 u_1dot

h = [];
h = [h, y_pdot];
h = [h, z_pdot];
h = [h, y_LMdot-u_1dot];
h = [h, theta_pdot];

% Compute W
W = sym(0)*ones(length(qdot), length(h));
for i = 1:length(h)

    W(:, i) = rowdiff(h(i), qdot).';

end

% Compute wtilde
wtilde = simple(h.'-W.'*qdot);

% - Here everything is rewritten! -
% Lefthand is the lefthand part of the Lagrange equations
lefthand = simple(ddtT_qdot - T_q + V_q).';

% Compute M
M = sym(0)*ones(length(lefthand));
for i = 1:length(lefthand)

    for j = 1:length(lefthand)

        eval(['M(' num2str(i) ',' num2str(j) ') = ...
            findddot(lefthand(' num2str(i) '), qddot(' num2str(j) '));']);

    end

end
M = simple(M);

% Substract M from lefthand part of equations
eqns = lefthand - Qnc;
eqnsrest = simple(eqns - M*qddot);

% Set tau and S to 0 in this case
tau = sym(0);
S = sym(0)*ones(length(eqnsrest), length(tau));

% Compute H
H = simple(eqnsrest + S*tau);
```

APPENDIX F
MATLAB: TDIFF.M

```
% R = tdiff(S)
%
% The tdiff function differentiates a symbolic function S w.r.t. time
% thereby taking care of possible time dependent variables or functions.
%
% The global variable tfuns is used to denote which functions are
% explicit functions of time and therefore have derivatives which can not
% be neglected.
%
% Example:
%
% >> global tfuns
% >> syms x a b
% >> tfuns = [x];
% >> tdiff(a*x^2+b)
%
% ans =
%
% 2*a*x*xdot
%
% Note: The maximum possible derivative for a function is its second
% derivative denoted by a ddot suffix.
function R = tdiff(S)

    % Define the tfuns global variable
    global tfuns tfunsdot tfunsddot

    % Define the tfunsdot and tfunsddot functions
    expandtfuns();
    ttest = [tfuns tfunsdot tfunsddot];

    % Initialise the result and define tl
    R = sym(0);
    tl = length(ttest);

    % Here the actual computation
    for i = 1:2*tl/3

        P = diff(S, ttest(i));
        R = R + P*ttest(i+tl/3);

    end

end

% expandtfuns() generates the tfunsdot and tfunsddot variables
function [] = expandtfuns()

    % Define the tfuns global variable
    global tfuns tfunsdot tfunsddot

    % Initialise tfunsdot and tfunsddot
    tfunsdot = [];
    tfunsddot = [];

    % Fill tfunsdot and tfunsddot rows
    for j = 1:length(tfuns)

        tfunsdot = [tfunsdot sym([findsym(tfuns(j)) 'dot'])];
        tfunsddot = [tfunsddot sym([findsym(tfuns(j)) 'ddot'])];

    end

end
```

APPENDIX G
MATLAB: ROWDIFF.M

```
% R = rowdiff(S, q)
%
% Differentiates a column or row, S, with respect to all variables given in
% q. Example:
%
% >> syms x y
% >> S = [x^2, x*y^2];
% >> q = [x y];
% >> rowdiff(S, q)
% ans =
%
% [   2*x,     0]
% [   y^2, 2*x*y]
function R = rowdiff(S, q)

    % Initialise R
    R = [];

    % Some very basic input check
    s = size(S);
    if s(1) > 1 && s(2) > 1

        error('Invalid input: S should be 1-dimensional');

    elseif s(2) > 1

        S = S.';

    end

    % Compute all derivatives
    for i = 1:length(q)

        R = [R diff(S, q(i))];

    end

end
```

APPENDIX H
MATLAB: FINDDDOT.M

```
% a = findddot(s, m)
%
% Function used for extracting double derivatives (*ddot-variables) in a
% symbolic expression for rewriting Lagrange's equations in a format:
% M*qddot+H=S*tau
%
% Usage example:
%
% >> syms xddot yddot
% >> findddot(3*xddot, [xddot])
%
% ans =
%
% 3
%
% So it returns the xddot term in the expression. Another example:
%
% >> syms xddot yddot
% >> findddot(3*xddot*yddot, [xddot])
% ??? Error using ==> findddot
% Cross term found in ddot!
%
% Here an error is generated as in the resulting expression still a
% *ddot-variable is found and rewriting in the right format is impossible.
% This normally indicates something was wrong with the input.
function a = findddot(s, m)

    % Differentiate the input equation w.r.t. the specified variable
    k = diff(s, m);

    % The resultafter substracting the found term from the original
    % expression is analysed for any further existance of *ddot-variables
    h = simplify(s - k*m);
    k_m = findstr(char(k), char(m));
    k_ddot = findstr(char(k), 'ddot');
    h_m = findstr(char(h), char(m));

    % Output errors if applicable
    if length(k_m) ~= 0

        error('No clear term could be found');

    elseif length(k_ddot) ~= 0

        error('Cross term found in ddot!');

    elseif length(h_m) ~= 0

        error('No clear term could be found');

    % Or output the result
    else

        a = simplify(k);

    end

end
```