A STUDY OF ROLL INDUCED BY CRANE MOTION ON SHIPS: A CASE STUDY USING MOVING FRAMES AND SE(3)

ANDREAS NORDVIK
NATALIA BATOOL KHAN
ROBERTO ANDREI BURCĂ

Bachelor's thesis in Marine & Mechanical Engineering
Bergen, Norway 2017

Høgskulen på Vestlandet

A STUDY OF ROLL INDUCED BY CRANE MOTION ON SHIPS: A CASE STUDY USING MOVING FRAMES AND SE(3)

Andreas Nordvik
Natalia Batool Khan
Roberto Andrei Burcă

Department of Mechanical and Marine Engineering
Western Norway University of Applied Sciences
NO-5063 Bergen, Norway

IMM 2017-M2

Høgskulen på Vestlandet
Avdeling for ingeniør- og økonomifag
Institutt for maskin- og marinfag
Inndalsveien 28,
NO-5063 Bergen, Norge

Title:	*A Study of Roll Induced by Crane Motion on Ships: A Case Study Using Moving Frames and SE(3)*

Author(s), student ID:	Andreas Nordvik (h146696) Natalia Batool Khan (h147169) Roberto Andrei Burcă (h146626)

Study program:	Mechanical Engineering, Marine Engineering
Date:	May 2017
Report number:	IMM 2017-M2 (M2: Group ID number)
Supervisor at HVL:	Professor Thomas J. Impelluso
Assigned by:	Western Norway Uni. of Applied Sciences
Contact person:	Professor Thomas J. Impelluso

Number of files delivered digitally: 1

Preface and acknowledgements

This report is the result of a bachelor's thesis at the Department of Mechanical and Marine Engineering (IMM) at Western Norway University of Applied Sciences (HVL). The bachelor's thesis is an obligatory part of the study plan to obtain a degree in Mechanical Engineering and Marine Engineering, worth 20 credits (ECTS).

The authors of this thesis would like to thank Professor Thomas J. Impelluso for his encouragement, willingness to help and assist at all times, and contagious interest in the field of dynamics. We greatly appreciate his valuable time.

We would also like to thank Professor Hidenori Murakami, the developer of the moving frame method. It has been a wonderful journey to learn and comprehend the method, as well as implement it in our work. Thank you for sharing your intellect with the world.

While working on this thesis, the authors used Matlab for calculations and Microsoft Word for the writing of the report. They have proven to be useful tools for us, and our thanks go to the developers of these programs for making our work easier.

This thesis is also written as a conference paper for IMECE2017. The paper will be published in Tampa, Florida, USA in November 2017, with article number IMECE2017-70111.

Date and place: 24.05.2017 Western Norway University of Applied Sciences HVL

Signed:

Andreas Nordvik

Natalia Batool Khan

Roberto Andrei Burcă

Copyright statement

Summary - Sammendrag

En ny metode i dynamikk – The Moving Frame Method (MFM) – er brukt for å analysere hvordan kranbevegelser på et skip påvirker skipets bevegelser.

MFM, som er basert på Lie Group Theory, Cartans Moving Frames og en kompakt notasjon fra geometrisk fysikk, muliggjør denne raskere utvinningen av de aktuelle bevegelsesligningene. Deretter utnyttes Special Euclidean Group SE(3) og en begrenset variasjon som skal brukes i Hamilton's Principle for å trekke ut bevegelsesligningene. Den matematiske modellen blir så forenklet for å få et tydeligere bilde av parametrene som påvirker kranens bevegelse.

Ligningene av interesse løses numerisk ved å bruke Runge-Kutta-metoden for å oppnå de spesifikke dataene for bevegelsen indusert av kranen. Deretter brukes Cayley-Hamilton-teoremet og Rodriquezs formel til å rekonstruere rotasjonsmatrisen.

For å supplere rapporten, er en nettside kodet med en modell av kranen og skipet, for å grafisk visualisere bevegelsen i 3D. Det er viktig å merke seg at der det er mange tilnærminger til dynamikk, presenterer MFM en konsistent metode, fra 2D til 3D, og på tvers av underdisipliner. Forenklingen er det som har gjort det mulig for studentene å gjennomføre dette prosjektet.

Abstract

This paper presents a new method in dynamics—The Moving Frame Method (MFM)—and applies it to analyze the roll, yaw and pitch of a ship at sea, as induced by an onboard moving crane. The MFM, founded on Lie Group Theory, Cartan's Moving Frames and a compact notation from geometrical physics, enables this expedited extraction of the equations of motion. Next, the method deploys the power of the special Euclidean Group SE(3) and a restricted variation to be used in Hamilton's Principle, to extract the equations of motion. The mathematical model is then simplified to get a clearer picture of the parameters that impact the motion of the crane. The equations of interest are numerically solved by using fourth order Runge-Kutta method to obtain the specific data for the motion induced by the crane. Then, The Cayley-Hamilton theorem and Rodriquez's formula is used to reconstruct the rotation matrix. To supplement the paper, a webpage is coded with a model of the crane and ship, to graphically visualize the motion in 3D. It is imperative to note that while there are many approaches to dynamics, the MFM presents a consistent method, from 2D to 3D, and across sub-disciplines. The simplification is what has enabled undergraduate students to undertake this project.

Table of contents

Preface and acknowledgements.....3
Copyright statement.....5
Summary 7
Abstract.....9
Table of contents.....11
Nomenclature.....13
1. Introduction.....16
2. Practical.....17
3. Method.....18
4. Results.....30
5. Conclusion.....36
References.....37
Appendix A.....38
Appendix B.....42
Appendix C.....43
Appendix D.....47
Appendix E.....60
Appendix F.....79

Nomenclature

$[B]$	B-matrix
C	Center of mass
$[D]$	Combined angular velocity matrix
E	Frame connection matrix
$\dot{E}$	Time derivative of frame connection matrix
e	Frame
$\{F^*\}$	Generalized force
$\{F\}$	Force and moment list
$\mathbf{H}$	Angular momentum
{H}	Generalized momenta
I_3	3x3 Identity matrix
$J_C^{(\alpha)}$	3x3 Mass moment of inertia matrix
K	Kinetic energy
{L}	Linear momentum
L	Lagrangian
[M]	Mass matrix
[M*]	Reduced mass matrix
[N*]	Reduced nonlinear velocity matrix
m	Mass
q	Generalized position
$\dot{q}$	Generalized velocity
$\ddot{q}$	Generalized acceleration
$\{\dot{q}\}$	General velocity variable list
$\{\ddot{q}\}$	Generalized acceleration variable list
R	Rotation matrix
$\dot{R}$	Time derivative of rotation matrix
r	Absolute position vector
s	Position vector
t	Time
U	Potential energy
δU	Variation of potential energy
δW	Virtual work
$\{\dot{X}\}$	Velocity list
$\{\tilde{X}\}$	Virtual displacements
x	Position vector

$\dot{x}$	Linear velocity vector
α	Index counter
δ	Variation
$\delta\Pi$	Variation of frame connection matrix
θ	Angle
$\overleftrightarrow{\delta\pi}$	Virtual rotational displacement
Ω	Time rate of the frame connection matrix
ω	Angular velocity vector
$\overleftrightarrow{\omega}$	Skew symmetric angular velocity matrix
$\underbrace{0}_{3x3}$	3x3 matrix of zeroes
0^T	1x3 matrix of zeroes
0	1x1 matrix of zeroes
1	1x1 matrix of ones

1. Introduction

The rolling of offshore systems induced by ocean waves results in undesirable motion [1]. Some companies lower an object into the sea through moon pools. A moon pool is a section of the ship that allows sea access but restricts rough waves. Essentially, it is a section of the ship in which a hole is cut through the bottom center of the boat. Water will not enter the vessel because it´s contained inside the moon pool's walls. However, a moon pool, as an opening in the floor or base of the ship, introduces stress concentrations in the hull. Thus, there is a preferred use of cranes.

Figure 1. A ship installed with a stiff boom crane

However, the use of a crane on a ship induces ship motion. This stems from unwanted reactionary moments from the driving torques. The movement of such large crane masses also impacts the ship motion.

Crane motion must be controlled more delicately as it is more impacted by the splash zone when lowering objects into the sea. One company trying to control crane motion is Palfinger [2]. Using sensors, the crane base is stabilized with hydraulic motion compensation for roll, yaw and pitch of the boat.
This paper provides an analysis that can be incorporated into such compensation systems. The research presented here develops a general mathematical model for the roll induced by crane motion on ships. Through the use of the moving frame method, this paper presents a systematic derivation of the equations of motion [3, 4].

This study obtains a system of equations that accounts crane induced ship movement. Initially, coupled nonlinear equations of motion are obtained. The mathematical model is then simplified to more clearly examine the role of stiff boom crane parameters and its effect on roll. The simplified model is then solved numerically for the proposed methods of rocking through the use of the fourth order Runge-Kutta method and a reconstruction formula for the rotations.

2. Practical

CRANE

Marine cranes are designed to meet the safety standards and extreme environmental conditions of the maritime industry. Stiff boom marine cranes are most suitable for general cargo handling and for service on board ships and offshore vessels where there is ample deck space. The cranes are available in the range from 141–30000 kNm lifting moment. Stiff boom cranes are used in dock, harbor conditions or on fixed installations. The crane studied in this analysis is Palfinger Marine´s DK2000 [5].

MODEL CRANE AND FRAME

The *idealized* physical model of the system is shown in Fig. 2. The major components of the system to be included in the analysis will be the outer ship body, a (yellow) crane tower and a (blue) boom. A red frame is placed at the center of mass of the ship. The crane rotates about the vertical (3)-axis, and a black frame is placed on the crane. The boom rotates about the (2)-axis; a yellow frame is placed on the boom.

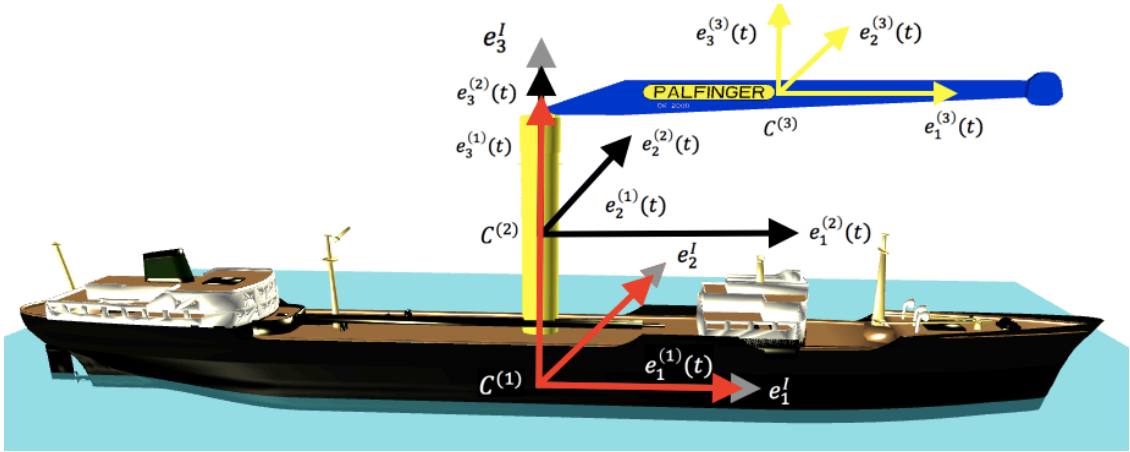


Figure 2. Model System Description

3. Method

THE MOVING FRAME METHOD

The moving frame method uses Lie Group Theory, Cartan's Moving Frames and Frankel's compact notation to facilitate the kinetic analysis. The complete background is presented elsewhere: [5-7]. Here, it will be summarized.

An inertial Cartesian coordinate system $x_C^{(\alpha)}(t) = \left\{x_1^{(\alpha)} \;\; x_2^{(\alpha)} \;\; x_3^{(\alpha)}\right\}^T$, along with the inertial vector basis $\mathbf{e}^I = (\mathbf{e}_1^I \;\; \mathbf{e}_2^I \;\; \mathbf{e}_3^I)$, is defined. The selection of an inertial frame implies the declaration of zero velocity at the origin as well as a zero position vector at the origin. Also, "x" is used to designate coordinates in an inertial frame.

Beginning with the first link and continuing outward, body numbers are assigned starting with α=1: ship (α=1); crane (α=2) and boom (α=3).

A Cartesian coordinate system $s^{(\alpha)}(t) = \left\{s_1^{(\alpha)} \;\; s_2^{(\alpha)} \;\; s_3^{(\alpha)}\right\}^T$ is attached at the center of mass, $C^{(\alpha)}$, of each body. This provides the vector basis $\mathbf{e}^{(\alpha)}(t) = \left(e_1^{(\alpha)}(t)\; e_2^{(\alpha)}(t)\; e_3^{(\alpha)}(t)\right)$. Also, "s" is used to designate coordinates of a moving frame.

To express the translation of body-α from an inertial frame perspective, the position vector $\mathbf{r}_C^{(\alpha)}(t)$ is expressed as:

$$\mathbf{r}_C^{(\alpha)}(t) = \mathbf{e}^I x_C^{(\alpha)}(t). \tag{1}$$

However, when expressing the position of body-α+1 relative to another body-(α) in the system, the *relative position vector* is defined as below using "**s**" instead of "**r**." Also, the alpha frame is used.

$$\boldsymbol{s}^{(\alpha+1/\alpha)}(t) = \mathbf{e}^{(\alpha)}(t) s^{(\alpha+1/\alpha)}(t). \tag{2}$$

Using Eqs. (1) and (2), position vectors can be expressed between the different components of the system and the inertial frame. Fig. 3 further demonstrates the notation used to differentiate between a relative position vector and a position vector from an inertial frame perspective.

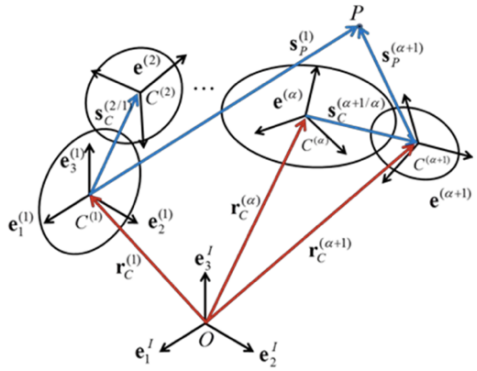


Figure 3. Visual representation of the notation

The orientation of $\mathbf{e}^{(\alpha)}(t)$ measured from $\mathbf{e}^I$ is expressed using a 3 × 3 rotation matrix $R^{(\alpha)}(t)$ (a member of the SO(3)group) as:

$$\mathbf{e}^{(\alpha)}(t) = \mathbf{e}^I R^{(\alpha)}(t). \tag{3}$$

To describe the orientation of body-(α + 1) with respect to body-(α), the superscript on $R^{(\alpha+1/\alpha)}(t)$ indicates a *relative* rotation and is presented in Eqn. (4):

$$\mathbf{e}^{(\alpha+1)}(t) = \mathbf{e}^{\alpha} R^{(\alpha+1/\alpha)}(t). \tag{4}$$

If the orientation of the $\mathbf{e}^{(\alpha+1)}(t)$ body is desired with respect to the inertial frame, Eq. (3) can be substituted into Eq. (4) as follows:

$$\mathbf{e}^{(\alpha+1)}(t) = \mathbf{e}^I R^{(\alpha)}(t) R^{(\alpha+1/\alpha)}(t) = \mathbf{e}^I R^{(\alpha+1)}. \tag{5}$$

Since the frames are related through rotation matrices, and rotation matrices are orthogonal and members of a group, then $R^{-1}(t) = R(t)^T$.

To describe both the translation and orientation of a moving frame *compactly* from an inertial frame, the 4 × 4 relative frame connection matrix $E^{(\alpha)}(t)$ of body-α can be defined as follows:

$$\left(\boldsymbol{e}^{(\alpha)}(t)\ \boldsymbol{r}_C^{(\alpha)}(t)\right) =$$

$$(\boldsymbol{e}^I\ \ \mathbf{0}) E^{(\alpha)}(t) = (\boldsymbol{e}^I\ \ \mathbf{0}) \begin{bmatrix} R^{(\alpha)}(t) & x_C^{(\alpha)}(t) \\ 0^T & 1 \end{bmatrix} \tag{6}$$

In the above, $0^T = [0\ \ 0\ \ 0]$.

While such homogenous transformation matrices were espoused by others [8], the Moving Frame Method explicitly recognizes that these are members of the special Euclidean Group, SE(3). From the paraphernalia of such group theory, the kinetic equations of motion are easily extracted.

Continuing, the *relative* frame connection matrix can be expressed for body-(α + 1) with respect to body-α

$$\left(\boldsymbol{e}^{(\alpha+1)}(t) \quad \boldsymbol{r}_C^{(\alpha+1)}(t)\right) =$$

$$(\boldsymbol{e}^{(\alpha)}(t) \quad \boldsymbol{r}_C^{(\alpha)}(t)) \begin{bmatrix} R^{((\alpha+1)/\alpha)}(t) & s_C^{(\alpha)}(t) \\ 0^T & 1 \end{bmatrix}. \tag{7}$$

Similar to individual rotation matrices, the following recursive relation exists for the frame connection matrices:

$$E^{(\alpha+1)}(t) = E^{(\alpha)}(t)E^{(\alpha+1/\alpha)}(t). \tag{8}$$

The kinematics of the system presented in Fig. 2 will be obtained, beginning with the outer most component, the ship, and moving up to the crane and the boom.

As the first step, a body attached coordinate system and corresponding coordinate frame is attached to all of the components. The goal of this section is to obtain the frame connection matrix for every component, from which the velocity and angular velocity vectors can be obtained for the kinetics section that follows.

Ship Kinematics

The ship's frame connection matrix can be stated directly with respect to the inertial frame using Eq. (6) above

$$\left(\boldsymbol{e}^{(1)}(t) \quad \boldsymbol{r}_C^{(1)}(t)\right) = (\boldsymbol{e}^{\boldsymbol{I}} \quad \boldsymbol{0}) \begin{bmatrix} R^{(1)}(t) & x_C^{(1)}(t) \\ 0^T & 1 \end{bmatrix}. \tag{9}$$

To obtain the velocity and angular velocity vectors, the time derivative of the above is taken, and then expressed with the ship's moving frame using the inverse of Eq. (9):

$$\left(\dot{\boldsymbol{e}}^{(1)}(t) \quad \dot{\boldsymbol{r}}_C^{(1)}(t)\right) = \left(\boldsymbol{e}^{(1)}(t) \quad \boldsymbol{r}_C^{(1)}(t)\right)\Omega^{(1)}(t) \tag{10}$$

where the 4 × 4 time rate of the ship frame connection matrix is expressed as:

$$\Omega^{(1)}(t) = \left(E^{(1)}(t)\right)^{-1} \dot{E}^{(1)}(t). \tag{11}$$

The matrix $\Omega^{(1)}(t)$ explicitly stated below, contains the velocity and angular velocity vectors with respect to the ship coordinate frame

$$\Omega^{(1)}(t) = \begin{bmatrix} \overleftrightarrow{\omega^{(1)}(t)} & \left(R^{(1)}(t)\right)^T \dot{x}_C^{(1)}(t) \\ 0^T & 0 \end{bmatrix}. \tag{12}$$

The upper left square quadrant of the above matrix presents the skew symmetric angular velocity matrix:

$$\overleftrightarrow{\omega^{(1)}(t)} = \left(R^{(1)}(t)\right)^T \dot{R}^{(1)} \tag{13}$$

where $\overleftrightarrow{\omega^{(1)}(t)}$ is used to express the time-rate of the ship coordinate frame with respect to its own frame

$$\dot{\boldsymbol{e}}^{(1)}(t) = \boldsymbol{e}^{(1)}(t)\overleftrightarrow{\omega^{(1)}(t)} = \begin{bmatrix} 0 & -\omega_3^{(1)}(t) & \omega_2^{(1)}(t) \\ \omega_3^{(1)}(t) & 0 & -\omega_1^{(1)}(t) \\ \omega_2^{(1)}(t) & \omega_1^{(1)}(t) & 0 \end{bmatrix} \tag{14}$$

The superposed double arrow indicates a skew symmetric angular-velocity matrix $\overleftrightarrow{\omega^{(1)}(t)}$. From $\overleftrightarrow{\omega^{(1)}(t)}$, the angular velocity components can be obtained, but in the moving frame.

$$\boldsymbol{\omega}^{(1)}(t) = \boldsymbol{e}^{(1)}(t)\omega^{(1)}(t) = \boldsymbol{e}^{(1)}(t)\begin{Bmatrix} \omega_1^{(1)}(t) \\ \omega_2^{(1)}(t) \\ \omega_3^{(1)}(t) \end{Bmatrix}. \tag{15}$$

The upper right quadrant of Eqn. (12) contains the velocity vector of the ship. However, it is preferred to write this term with respect to the inertial frame so that the forces in the kinetics section can be expressed in the inertial frame as well. Thus, the velocity vector can be expressed as:

$$\dot{\boldsymbol{r}}_C^{(1)}(t) = \boldsymbol{e}^I \dot{x}_C^{(1)}(t). \tag{16}$$

Equations (15) and (16) are the first two terms needed in this analysis. Attention is now shifted to the crane.

Crane Kinematics

Similar to the ship, the crane relative frame connection matrix and velocity and angular velocity vectors must be obtained. The fixed relative position vectors for the center of mass of the crane, using the ship frame, can thus be expressed as follows:

$$\boldsymbol{s}_c^{(2/1)} = \boldsymbol{e}^{(1)}(t)s_c^{(2/1)} = \boldsymbol{e}^{(1)}(t)\begin{Bmatrix} h_1^{(2/1)} \\ h_2^{(2/1)} \\ h_3^{(2/1)} \end{Bmatrix}. \tag{17}$$

Note that the relative position vector components are constant with respect to the ship frame. However, nothing prevents the coordinates in Eqn. (17) to also be time dependent.

The relation between the ship and crane frames can be expressed by an elementary rotation matrix

$$\boldsymbol{e}^{(2)}(t) = \boldsymbol{e}^{(1)}R^{(2/1)}(t). \tag{18}$$

Since the crane is rotating around a single axis, one notes: $\boldsymbol{e}_3^{(2)}(t) = \boldsymbol{e}_3^{(1)}(t)$. The rotation matrix can be expressed as follows:

$$R^{(2/1)}(t) = \begin{bmatrix} cos\phi(t) & -sin\phi(t) & 0 \\ sin\phi(t) & cos\phi(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{19}$$

The relative crane connection matrix can now be assembled:

$$\begin{aligned} \left(\boldsymbol{e}^{(2)}(t) \;\; \boldsymbol{r}_C^{(2)}(t)\right) &= \left(\boldsymbol{e}^{(1)}(t) \;\; \boldsymbol{r}_C^{(1)}(t)\right)\begin{bmatrix} R^{(2/1)}(t) & s_C^{(2/1)} \\ 0 & 1 \end{bmatrix} \\ &= \left(\boldsymbol{e}^{(1)}(t) \;\; \boldsymbol{r}_C^{(1)}(t)\right)E^{(2/1)}(t). \end{aligned} \tag{20}$$

The above expressions in Eq. (20) will be written with respect to the inertial frame:

$$\left(\boldsymbol{e}^{(2)}(t) \;\; \boldsymbol{r}_C^{(2)}(t)\right) = \; (\boldsymbol{e}^I \;\; 0)E^{(1)}(t)E^{(2/1)}(t) = (\boldsymbol{e}^I \;\; 0)E^{(2)}(t). \tag{21}$$

The time derivative taken from the above Eq. (21) produces $\Omega^{(2)}(t)$.

As with the ship, the angular velocity vector can be extracted from the upper left quadrant as:

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t) + \dot{\phi}^{(2)}e_3 \tag{22}$$

In the above equations, the relative angular velocity of the crane from the ship (the second term on the right) has been simplified using $e_3 = (0 \;\; 0 \;\; 1)^T$— a

column vector isomorphic to the third basis of the corresponding Lie Algebra.

The velocity vector components can be obtained from the second columns. Expressing the component with respect to the inertial frame results in Eq. (23) below

$$\dot{x}_C^{(2)} = \dot{x}_C^{(1)} + R^{(1)}(t)\left(\overleftrightarrow{s_C^{(2/1)}}^T\right)\omega^{(1)}. \tag{23}$$

Equations (22) and (23) are the second two terms needed in this analysis. Attention is now shifted to the last link of the crane.

Boom Kinematics

The kinematic study for the boom is similar to the crane and the ship. The relative position vectors for the boom attached to the crane can thus be expressed as follows:

$$\boldsymbol{s}_c^{(3/2)} = \boldsymbol{e}^{(2)}(t)s_c^{(3/2)} = \boldsymbol{e}^{(2)}(t)\begin{Bmatrix} d_1^{(3/2)} \\ d_2^{(3/2)} \\ d_3^{(3/2)} \end{Bmatrix}. \tag{24}$$

The rotation of the boom, similar to the crane, is only about a single axis $\boldsymbol{e}^{(2)}(t)$; therefore, the relation between the crane and boom frames can be expressed as follows:

$$\boldsymbol{e}^{(3)}(t) = \boldsymbol{e}^{(3)}(t)R^{(3/2)}(t). \tag{25}$$

The elementary rotation matrix can be expressed as below

$$R^{(3/2)}(t) = \begin{bmatrix} cos\xi(t) & 0 & sin\xi(t) \\ 0 & 1 & 0 \\ -sin\xi(t) & 0 & cos\xi(t) \end{bmatrix}. \tag{26}$$

The boom/frame connection matrix is now assembled and expressed with respect to the inertial frame by using Eq. (21).

$$\left(\boldsymbol{e}^{(3)}(t) \;\; \boldsymbol{r}_C^{(3)}(t)\right) =$$

$$(\boldsymbol{e}^I \;\; 0)E^{(3)}(t) = (\boldsymbol{e}^I \;\; 0)E^{(1)}E^{(2/1)}E^{(3/2)}. \tag{27}$$

Taking the time derivative of Eq. (27) enables one to extract the following angular velocity components

$$\omega^{(3)}(t) =$$

$$\left(R^{(3/1)}(t)\right)^T \omega^{(1)}(t) + \left(R^{(3/2)}(t)\right)^T \dot{\phi}^{(2)} e_3 + \dot{\xi}^{(2)} e_2. \tag{28}$$

where $R^{(3/1)}$ is expressed as the following:

$$R^{(3/1)}(t) = R^{(2/1)}(t) R^{(3/2)}(t). \tag{29}$$

Once again, the last term in Eq. (28) represents a relative rotation from the crane and can be simplified using: $e_2 = (0 \;\; 1 \;\; 0)^T$.

Since the boom and the crane center of mass coincide, the velocity components are the same for both and can be obtained

$$\dot{x}_C^{(3)} = R^{(1)}(t) R^{\left(\frac{2}{1}\right)}(t) R^{\left(\frac{3}{2}\right)}(t) \begin{pmatrix} \overleftrightarrow{s_C^{\left(\frac{3}{2}\right)}}^T \end{pmatrix} \omega^{(3)} +$$

$$R^{(1)}(t) R^{(2/1)}(t) \begin{pmatrix} \overleftrightarrow{s_C^{(2/1)}}^T \end{pmatrix} \omega^{(2)} + R^{(1)}(t) \begin{pmatrix} \overleftrightarrow{s_C^{(2/1)}}^T \end{pmatrix} \omega^{(1)} + \dot{x}_C^{(1)} \tag{30}$$

Expressions for the velocity and angular velocity of all three components have now been obtained.

The next goal is to formulate these expressions in terms of a minimal set of generalized coordinates.

Generalized Essential Coordinates

The system consists of three major components, or three moving frames, each with a corresponding velocity and angular velocity vector. They can collectively be expressed in a 18 × 1 column matrix $\{\dot{X}(t)\}$, referred to as the generalized velocities. The generalized velocities can be more simply expressed by the set of independent, essential generalized velocities, which are assembled in a 8×1 column matrix $\{\dot{q}(t)\}$

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(2)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(3)}(t) \\ \omega^{(3)}(t) \end{Bmatrix}, \quad \{\dot{q}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(3)} \end{Bmatrix}. \tag{31}$$

These generalized velocities are, respectively, the translational velocity of the ship, the angular velocity of the ship, the rotational velocity of the crane and the

rotational velocity of the boom. Reviewing (16), (17), (22), (23), (28) and (30), the generalized velocities are linearly related to the essential generalized velocities through the use of the $[B(t)]$matrix.

$$\{\dot{X}(t)\} = [B(t)]\{\dot{q}(t)\} \tag{32}$$

$[B(t)]$ is a 18 × 8 matrix assembled from the velocity and angular velocity vectors derived in the kinematics section and shown in Eqn. (33):

$$[B(t)] = \begin{bmatrix} I_3 & 0_3 & 0_1 & 0_1 \\ 0_3 & I_3 & 0_1 & 0_1 \\ I_3 & R^{(1)}(t)\left(\overleftarrow{s_C^{(2/1)}}^{T}\right) & 0_1 & 0_1 \\ 0_3 & \left(R^{(2/1)}(t)\right)^T & e_3 & 0_1 \\ I_3 & B_{52} & B_{53} & B_{54} \\ 0_3 & \left(R^{(3/1)}(t)\right)^T & \left(R^{(3/2)}(t)\right)^T e_3 & e_2 \end{bmatrix}. \tag{33}$$

In the above:

$$0_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad 0_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{34}$$

The derivation of [B(t)] concludes the kinematics of the system.

Kinetics

In this section, the equations of motion for the system are derived through the use of the principal of virtual work obtained from Hamilton's Principle. To derive the principle of virtual work, the Lagrangian $\hat{L}$ is defined as the difference between the kinetic energy K and the potential energy U

$$\hat{L} = K - U. \tag{35}$$

The kinetic energy consists of both translational and rotational energy with respect to the center of mass of each of the components. To define the total kinetic energy of the system, expressions for the linear momentum, $\boldsymbol{L}_C^{(\alpha)}$, and angular $\boldsymbol{H}_C^{(\alpha)}$, are needed.

$$\boldsymbol{L}_C^{(\alpha)} = \boldsymbol{e}^I L_C^{(\alpha)} = \boldsymbol{e}^I m^\alpha \dot{x}_C^{(\alpha)}. \tag{36}$$

$$\boldsymbol{H}_C^{(\alpha)} = \boldsymbol{e}^{(\alpha)} H_c^{(\alpha)} = \boldsymbol{e}^{(\alpha)} J_c^{(\alpha)} \omega^{(\alpha)}. \tag{37}$$

The total kinetic energy can be more compactly expressed in matrix form as:

$$K = \frac{1}{2}\{\dot{X}(t)\}^T\{H\} = \frac{1}{2}\{\dot{X}(t)\}^T[M]\{\dot{X(t)}\}, \tag{38}$$

where $\{H(t)\}$, the generalized momenta, contains both linear and angular momentum

$$\{H\} = \begin{pmatrix} L_C^{(1)} \\ H_C^{(1)} \\ L_C^{(2)} \\ H_C^{(2)} \\ L_C^{(3)} \\ H_C^{(3)} \end{pmatrix} = [M]\{\dot{X}(t)\}, \tag{39}$$

and $[M]$, the generalized mass matrix, is diagonal and contains the masses and moments of inertia of the bodies in the system.

$$[M] = \begin{bmatrix} m^{(1)}I_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & J_C^{(1)} & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & m^{(2)}I_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & J_C^{(2)} & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & m^{(3)}I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & J_C^{(3)} \end{bmatrix}. \tag{40}$$

For the moments of inertia above, it has been assumed that the coordinate systems, located in the center of mass of each body, coincide with the principal axes, resulting in diagonal $J_C^{(\alpha)}$ matrices.

The potential energy U of each body is due to gravitational forces

$$U^{(\alpha)} = m^{(\alpha)}gx_{3C}^{(\alpha)}. \tag{41}$$

To take the variation of the Lagrangian, the variation of the generalized velocities needs to be obtained. The variation of a body-α frame connection matrix is defined by Murakami [4]

$$\delta\Pi^{(\alpha)} = \begin{bmatrix} \overleftrightarrow{\delta\pi^{(\alpha)}(t)} & \left(R^{\alpha}(t)\right)^T\delta x_C^{(\alpha)}(t) \\ 0^T & 0 \end{bmatrix}. \tag{42}$$

The virtual rotational displacements, $\overleftrightarrow{\delta\pi^{(\alpha)}(t)}$, are defined as follows:

$$\overleftrightarrow{\delta\pi^{(\alpha)}(t)} = \left(R^{\alpha}(t)\right)^T\delta R^{(\alpha)}(t), \tag{43}$$

$$\delta\boldsymbol{\pi}^{(\alpha)}(t) = \boldsymbol{e}^{(\alpha)}(t)\delta\pi^{(\alpha)}(t). \tag{44}$$

The virtual generalized displacements are assembled in a column matrix $\{\delta\tilde{X}\}$:

$$\{\delta\tilde{X}(t)\} = \begin{pmatrix} \delta x_C^{(1)}(t) \\ \delta\pi_C^{(1)}(t) \\ \delta x_C^{(2)}(t) \\ \delta\pi_C^{(2)}(t) \\ \delta x_C^{(3)}(t) \\ \delta\pi_C^{(3)}(t) \end{pmatrix}. \tag{45}$$

The critical part comes by imposing the commutativity of time differentiation, d/dt- operator, and the variation of the frame, δ-operator during the integration of the Lagrangian. This results in the following restriction on the variation of the generalized angular velocities and velocities [9].

$$\delta\omega^{(\alpha)} = \frac{d}{dt}\delta\pi^{(\alpha)}(t) + \overleftrightarrow{\omega^{(\alpha)}(t)}\delta\pi^{(\alpha)}(t) \tag{46}$$

$$\frac{d}{dt}\delta x_C^{(\alpha)}(t) = \delta\dot{x}_C^{(\alpha)}(t). \tag{47}$$

Placing the above Eqs. (46) and (47) in matrix form for use in the variation of the Lagrangian results in the following compact expression:

$$\left\{\delta\dot{X}\right\} = \left\{\delta\dot{\tilde{X}}\right\} + [D]\{\delta\tilde{X}\}, \tag{48}$$

where the skew symmetric matrix [D] is expressed as:

$$[D] = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & \overleftrightarrow{\omega^{(1)}(t)} & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & \overleftrightarrow{\omega^{(2)}(t)} & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & \overleftrightarrow{\omega^{(3)}(t)} \end{bmatrix}. \tag{49}$$

The variation of the Lagrangian thus becomes

$$\delta L = \left\{\delta\dot{X}(t)\right\}^T[M]\left\{\dot{X}(t)\right\} - \{\delta\tilde{X}(t)\}^T\{G\} \tag{50}$$

where $\{G\}$ represents the assembled gravitational forces:

$$\{G\} = \begin{pmatrix} m^{(1)}ge_3 \\ 0_1 \\ m^{(2)}ge_3 \\ 0_1 \\ m^{(3)}ge_3 \\ 0_1 \end{pmatrix} \tag{51}$$

Before proceeding, and to simplify the analysis and the number of equations, the essential virtual displacements are obtained.

$$\{\delta q(t)\} = \begin{pmatrix} \delta x_C^{(1)}(t) \\ \delta \pi^{(1)}(t) \\ \delta \varphi^{(2)}(t) \\ \delta \psi^{(3)}(t) \end{pmatrix} \tag{52}$$

As done with the generalized velocities in Eq. (30), a similar relation can be obtained that relates the virtual generalized displacements, $\{\delta \tilde{X}(t)\}$, and the essential virtual displacements, $\{\delta q(t)\}$.

$$\{\delta \tilde{X}(t)\} = [B]\{\delta q(t)\} \tag{53}$$

Holding this in abeyance, the applied forces will now be assembled as:

$$\{Q(t)\} = \begin{pmatrix} F_C^{(1)I}(t) \\ M_C^{(1)}(t) \\ F_C^{(2)I}(t) \\ M_C^{(2)}(t) \\ F_C^{(3)I}(t) \\ M_C^{(3)}(t) \end{pmatrix} = \begin{pmatrix} F^{(w)I}(t) + F_b^I - m^{(1)} g e_3 \\ M^{(w)}(t) - M_m^{(c)}(t) e_3 \\ -m^{(2)} g e_3 \\ M_m^{(c)}(t) e_3 - M_m^{(b)}(t) e_2 \\ -m^{(3)} g e_3 \\ M_m^{(b)}(t) e_2 \end{pmatrix} \tag{54}$$

where

$$F^{(w)I} = Force\ from\ waves$$

$$F_b^I = Force\ from\ buoyance$$

$$m^{(1)} g e_3 = Force\ down\ from\ gravity\ on\ ship$$

$$M^{(w)}(t) = Moment\ from\ waves$$

$$-M_m^{(c)}(t) e_3 = Reverse\ moment\ on\ ship\ from\ motor\ that\ turns\ the\ crane$$

$$-m^{(2)} g e_3 = Force\ gravity\ down\ on\ the\ crane$$

$$M_m^{(c)}(t) e_3 = Moment\ that\ turns\ the\ crane$$

$$-M_m^{(b)}(t) e_2 = Reverse\ moment\ on\ crane\ from\ boom$$

$$-m^{(3)} g e_3 = Force\ gravity\ down\ on\ boom$$

$$M_m^{(b)}(t) e_2 = Moment\ on\ crane\ that\ turns\ the\ boom$$

The virtual work done by the external forces, couples, and actuator couples, needs to be considered. The virtual work done by the physical forces, where moments and virtual rotations are the natural pair, is expressed using:

$$\delta W = \{\delta \tilde{X}(t)\}^T \{Q(t)\} \tag{55}$$

Eq. (55) states that the variation of the work is such that the forces are taken through a virtual displacement and the moments are conjugate to the variation: $\delta\pi$.

Recognizing Eq. (53), Eq. (55) can be established as:

$$\delta W = \{\delta q(t)\}^T \{F^*(t)\} \tag{56}$$

Where the essential force $\{F^*(t)\}$ is defined as a $n^* \times 1$ matrix, where n^* defines the number of essential generalized coordinates:

$$\{F^*(t)\} = [B(t)]^T \{Q(t)\} \tag{57}$$

In the above equation, $F^{(w)I}(t)$ and $M^{(w)}(t)$ are the wave forces and couples acting on the ship, respectively.

The superscript *I* indicates that the components of the forces are with respect to the inertial frame. The action and reaction couples ${M^{(c)}}_m(t)$ and ${M^{(b)}}_m(t)$ are due to the motors operating the crane and the boom respectively.

Hamilton's principle is now stated. Here the middle term accounts for the restriction of the angular velocity, evinced by the D matrix, earlier:

$$\int_{t_0}^{t_1} \left\{\delta \dot{\tilde{X}}\right\}^T \{H\} + \{\delta \tilde{X}\}([D]^T \{H\} + \{Q(t)\}) dt = 0 \tag{58}$$

Integrating by parts, and noting that the virtual displacements vanish at t_0 and t_1, yields the principle of virtual work that holds for all time and must be zero:

$$\{\delta \tilde{X}\}^T \left[\{\dot{H}(t)\} + [D]\{H(t)\} - \{Q(t)\}\right] \tag{59}$$

The generalized momenta derived in Eq. (38), can be written with respect to the essential generalized velocities:

$$\{H(t)\} = [M][B(t)]\{\dot{q}(t)\} \tag{60}$$

Substituting Eqs. (59) and (60) back into the principal of virtual work, results in the equation below:

$$[B(t)]^T [M][B(t)]\{\ddot{q}(t)\} + [B(t)]^T \left([M]\left[\dot{B}(t)\right] + [D(t)][M][B(t)]\right)\{\dot{q}(t)\} - [B(t)]^T \{Q(t)\} = \{0\}. \tag{61}$$

Equation (61) represents the general formula for deriving the equations of motion through the use of essential generalized velocities. The framework for deriving the equations of motion of a ship with a stiff boom crane has been set. The solution of the above equation will be a 8×1 column vector containing the equations of motion for each corresponding essential generalized velocity.

The proper solution method entails obtaining all terms above at each time step and then conducting the matrix multiplication and necessary inversion to update the generalized coordinates. However, in the following analysis, a reduced form of the equations are studied.

With the use of symbolic computational software, most commonly available in MATLAB, the equations can be obtained in terms of the system parameters.

4. Results

SIMPLIFIED MODEL

It is imperative to state, at the outset, that Eqn. (61) can be readily solved using standard numerical methods. However, the equations of motion will be simplified for this student project. This will be done to allow one to see the effect of several of the system parameters. Also, this will enable the team of undergraduate students to complete the task and demonstrate that the Moving Frame Method makes 3D Dynamics accessible to undergraduates using a notation that is consistent across the sub-disciplines of Dynamics.

To simplify the mathematical model previously derived above, several assumptions will be made. First, the ship will be assumed stationary, thus the body will not translate but only rotate around its center of mass

$$\delta x_C^{(1)} = 0. \tag{62}$$

Second, the moments of inertia of the crane and boom will be calculated using a 3D simulation software (CreoParametric 3.0) and using a prescribed constant angular velocity.

The mass and inertia of the ship are prescribed and the inertia is assumed to be a solid cuboid.

$$J_1^{(1)} = \frac{1}{12} m(h^2 + d^2)$$

$$J_2^{(1)} = \frac{1}{12} m(w^2 + d^2)$$

$$J_3^{(1)} = \frac{1}{12} m(w^2 + h^2) \tag{63}$$

The crane´s mass and inertia are assumed to be zero. Thus the crane will only be

a boom-carrier. Again, it is possible to solve (2) directly, but the goal here is to create a first pass analysis.

The angular rate will be prescribed and there is no angular acceleration.

$$J_1^{(2)} = J_2^{(2)} = J_3^{(2)} = 0$$

$$\delta\dot{\phi} = 0. \tag{64}$$

Third, the boom will be assumed to have a prescribed mass and harmonic rate and there is no angular acceleration.

$$\delta\dot{\xi} = 0. \tag{65}$$

The above assumptions leave only three coupled equations of motion corresponding to the angular velocity of the ship.

$$\frac{d\omega_1^{(1)}}{dt} = f\left(\omega^{(1)}, \omega^{(2)}, \omega^{(3)}, J^{(1)}, J^{(2)}, J^{(3)}, t\right) \quad \text{(a)}$$

$$\frac{d\omega_2^{(1)}}{dt} = f\left(\omega^{(1)}, \omega^{(2)}, \omega^{(3)}, J^{(1)}, J^{(2)}, J^{(3)}, t\right) \quad \text{(b)}$$

$$\frac{d\omega_3^{(1)}}{dt} = f\left(\omega^{(1)}, \omega^{(2)}, \omega^{(3)}, J^{(1)}, J^{(2)}, J^{(3)}, t\right) \quad \text{(c)}$$

(66)

where $\frac{d\omega_1^{(1)}}{dt}$, $\frac{d\omega_2^{(1)}}{dt}$, $\frac{d\omega_3^{(1)}}{dt}$ can be found in appendix.

These equations of motion are a suite of coupled nonlinear first order differential equations dependent on the mass properties of the ship and the boom to be stabilized, the crane's constant angular velocity ($\phi(t)$, and the position and angular velocity of the boom ($\xi(t)$). The controllable parameters or inputs of the system are thus the angular velocity of the crane and the excitation of the boom. The focus is now on the effect of the above parameters and their effect on the amplitude of oscillation on the ship.

The distance between the center of masses to the ship and the other links are assumed to be zero. The result of this is that $\dot{x}_C^{(2)}(t) = 0$ and $\dot{x}_C^{(3)}(t) = 0$. This can be justified due to its small contribution to the linear momentum, Eq. (39).

RECONSTRUCTION OF ROTATION MATRIX

The orientation of the ship is dependent on the rotation matrix, $R^{(1)}(t)$. Thus, the rotation matrix must be known at every time step. The descriptive equation for the rotation matrix $R^{(1)}(t)$ at each time step after integrating the equations of motion is given by the following (which is a reformulation of Eqn. (13)):

$$\dot{R}^{(1)}(t) = R^{(1)}(t)\overleftrightarrow{\omega^{(1)}(t)}. \tag{67}$$

Equation (67) can be integrated analytically from t to $t + \Delta t$ if the angular velocity is constant. Unfortunately, that is not the case with the problem under analysis. To accommodate for changing angular velocities, the reconstruction can be performed at every time step of the Runge-Kutta integration, assuming the angular velocity stays constant from t to $t + \Delta t$. The mid-point integration method is adopted, using the mean value of the angular velocity. The reconstruction formula, by integrating Eq. (67) is obtained below:

$$R(t + \Delta t) = R(t)exp\{\Delta t\omega(t + \Delta t/2\}. \tag{68}$$

where the matrix exponential can be calculated using the following expression:

$$exp\{t\omega_0\} = \quad I_3 + \frac{\omega_0}{\|\omega_0\|}\sin(t\|\omega_0\|) + \left(\frac{\omega_0}{\|\omega_0\|}\right)^2 \left(1 - cos(t\|\omega_0\|)\right). \tag{69}$$

Numerical Solution

Equation (59) will be solved numerically to see the effect of the several controllable parameters. The moment of inertias of the boom will be calculated using a 3D simulation software (CreoParametric 3.0)

$$J_1^{(3)} = 1.1763498 * 10^5\ tonne * m^2$$

$$J_2^{(3)} = 1.1800369 * 10^7 tonne * m^2$$

$$J_3^{(3)} = 1.1739636 * 10^7\ tonne * m^2$$

The mass moment of inertia of the ship are calculated with Eq. (63) and are given these parameters:

$$m^{(1)} = 4100\ tonne$$

$$w = 22\ m\ , \qquad h = 10\ m, \qquad d = 85m.$$

Third, a sinusoidal method of excitation for the boom will be prescribed. The boom's angular displacement will thus be described by Eq. (70) below.

$$\xi(t) = Asin(\omega t) \tag{70}$$

To ignore any confusion, A above represents the amplitude of oscillation of the

boom in radians and ω is the angular frequency. For this first numerical test, A is set to be 1.414 radians and $\omega = 2\pi f$, where f is set to be 0.05 second. The angular velocity $\xi(t)$ can thus easily be obtained from Eq. (60). The angular velocity of the crane is kept constant at $\phi = 0.15\ rad/s$. Substituting in the mass parameters and the sinusoidal excitation back into Eq. (54), the equation of motion can be numerically solved. Naturally, the boom angular velocity suggests it passes through the crane. Thus, it is best to ignore the results after one half a cycle. However all the data is presented just to observe the predicted results.

Through the use of the 4^{th} order Runge-Kutta method, these equations are solved with time steps of 0.05 seconds to 150 seconds. The plots are presented below in Fig. 4, 5 and 6, where the solid line represents the roll, pitch and yaw, and the dashed line is the harmonic rate of the boom.

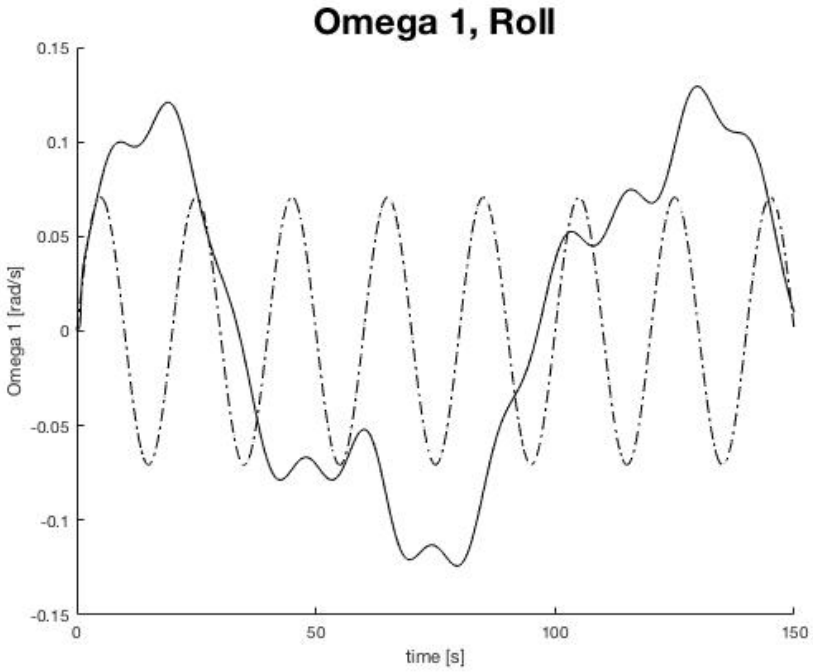


Figure 4. Plot of the output (solid line) and input (dotted line) angular velocity obtained from the numerical solution.

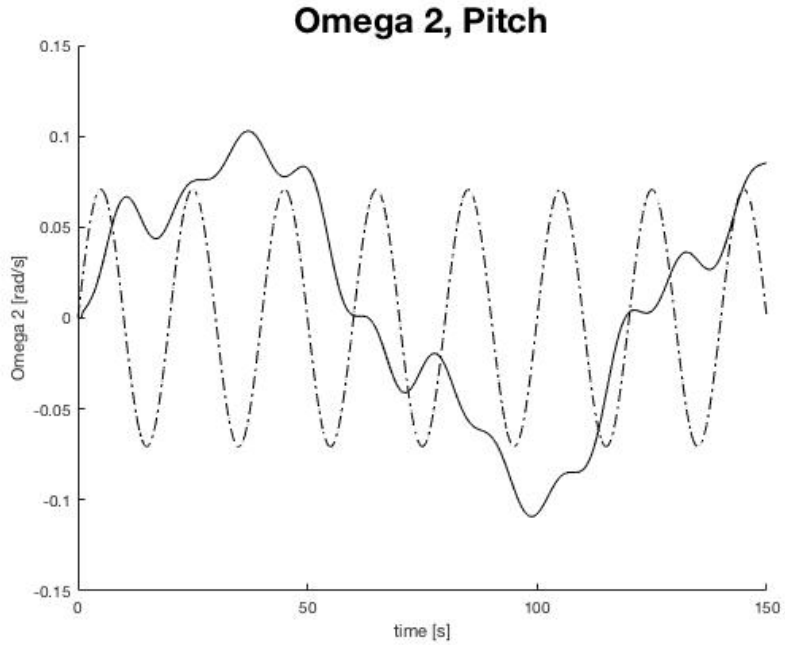


Figure 5 - Plot of the angular velocity about the second axis.

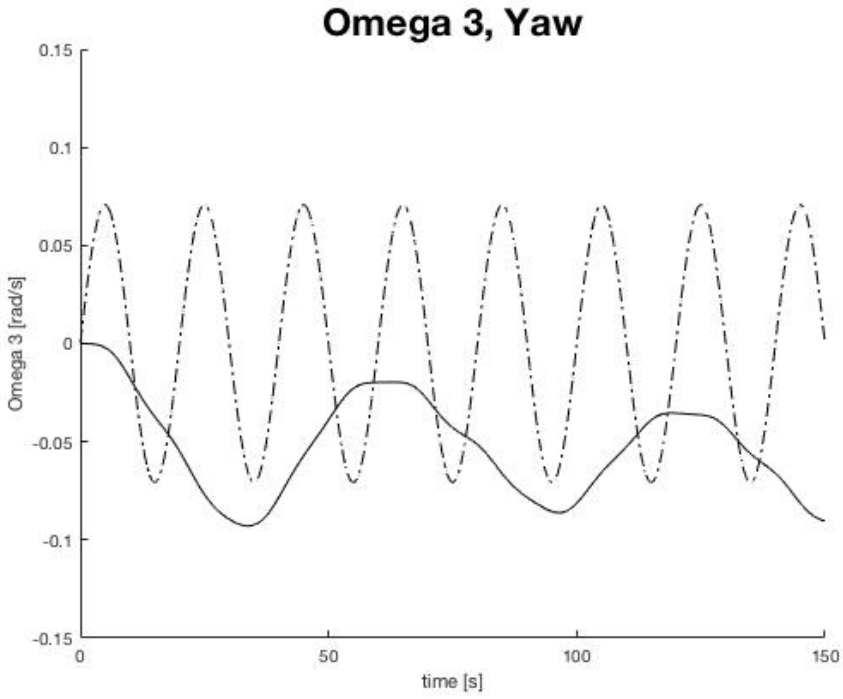


Figure 6 - Plot of the prescribed angular velocity about the third axis.

In these plots, one observes the induced angular velocity due to the input angular velocity of the crane and boom. The response of the ship closely matches the input angular velocities of the two links. However, the reader is reminded that many simplifications were undertaken. The work ignored wave forces, added mass and the input torque generator on the two arms. However, these can be readily added and these results do confirm the qualitative response.

More informative, perhaps, is obtained by observing the induced rocking on the ship, itself. While this has not been plotted in 2D, the authors point the reader to a 3D web page (viewable on most mobile devices) (control box in upper right corner of web page):

http://home.hib.no/prosjekter/dynamics/2017/crane/

On this web page, one can alter the input parameters and observe the rocking motion of the craft. Left, middle and right mouse buttons translate as: zoom, pan and rotate.

5. CONCLUSION

Primarily, this work needs experimental confirmation. This work is next and will utilize a wave tank at HVL.

The mathematical model of the ship installed with a crane and the boom was obtained using the moving frame method. The equations of motion were obtained and solved. The work was conducted by undergraduate students.

The team took recourse to Matab's symbolic manipulator. This enabled us to extract three equations for the angular velocity. However, the price paid was a host of over simplifications.

Clearly, this is not advisable. The proper way would be to solve Eq. (61) and also employ the reconstruction of the rotation matrix from the angular velocity. In this way, all effects can be included: eccentric geometry, added mass of fluid, drag forces, internal moments that drive the crane and the turret, along with all masses.

The fact remains that the MFM enables an immediate extraction of the equations of motion. Also the notation used in the method is identical to that in an undergraduate introductory dynamics class.

REFERENCES

[1] Yuzhe, Q., Yongchun, F., 2016 "Dynamics Analysis of an Offshore Ship-mounted Crane Subject to Sea Wave Disturbances," *Proceedings of the 12th World Congress on Intelligent Control and Automation (WCICA) June 12-15, 2016, Guilin, China*

[2] Palfinger Marine: *3D Compensated Cranes* [Internet] Available at <https://www.palfinger.com/en/marine/products/cranes/wind-cranes/3d-compensated-cranes> [Read 10 May 2017].

[3] Murakami, H., 2013, "A moving frame method for multibody dynamics," *Proceedings of the ASME 2013 International Mechanical Engineering Congress & Exposition*, paper IMECE2013-62833.

[4] Murakami, H., 2015, "A moving frame method for multi- body dynamics using SE(3)," *Proceedings of the ASME 2015 International Mechanical Engineering Congress & Exposition*, paper IMECE2015-51192.

[5] Palfinger Marine: *Stiff Boom Cranes* [Internet] Available at <https://www.palfinger.com/en/marine/products/cranes/offshore-cranes/stiff-boom-cranes> [Read 10 May 2017]

[6] Impelluso, T., 2016, "Rigid body dynamics: A new philosophy, math and pedagogy," *Proceedings of the ASME 2016 International Mechanical Engineering Congress & Exposition,* paper IMECE2016-65970.

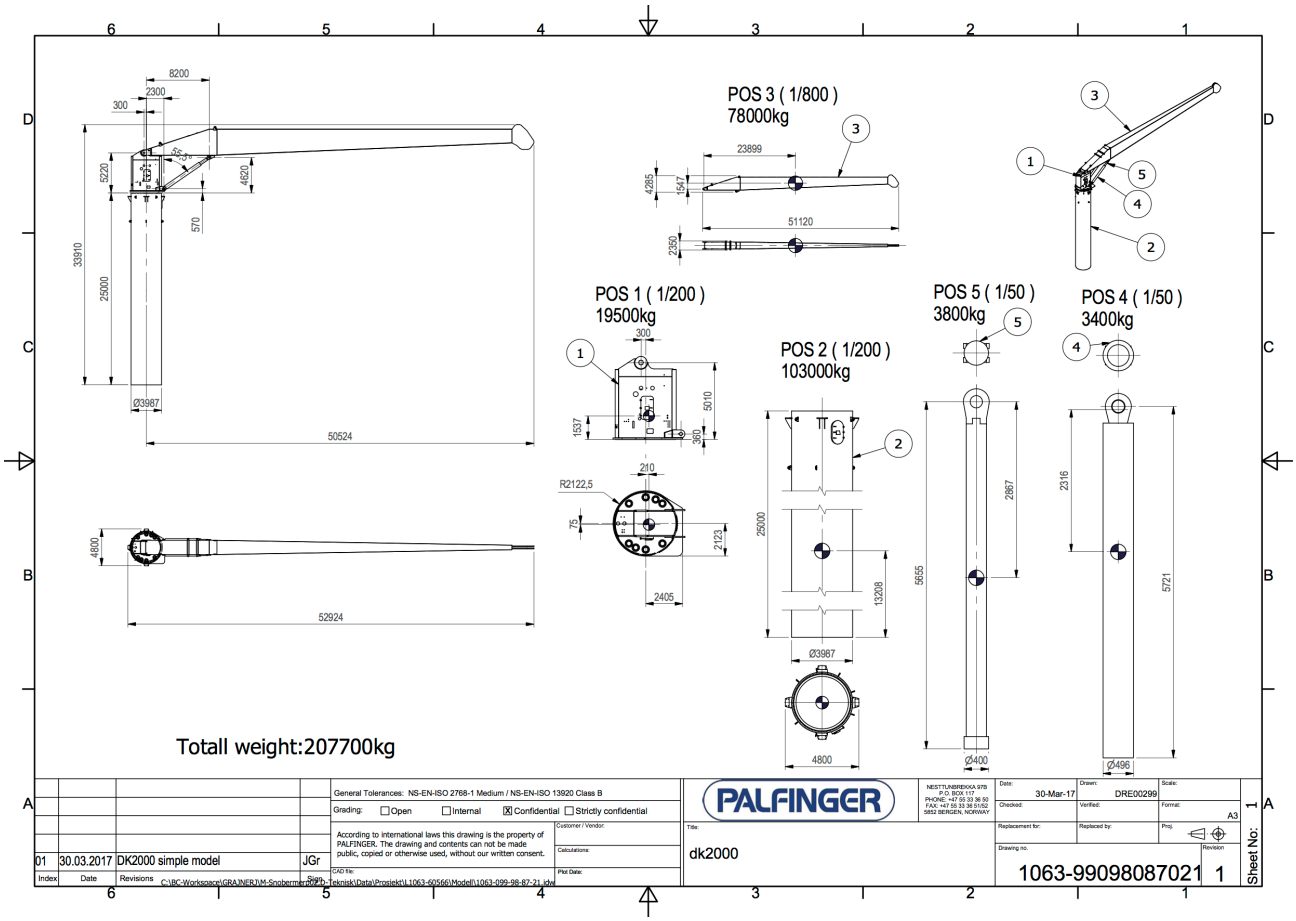
[7] Frankel, T., 2012, *The Geometry of Physics, an Introduction*, third edition, Cambridge University Press, New York; First edition published in 1997.

[8] Denavit, J. and Hartenberg, R., 1955, "A kinematic notation for lower-pair mechanisms based on matrices," *ASME Journal of Applied Mechanics, 22, pp.215-221.*

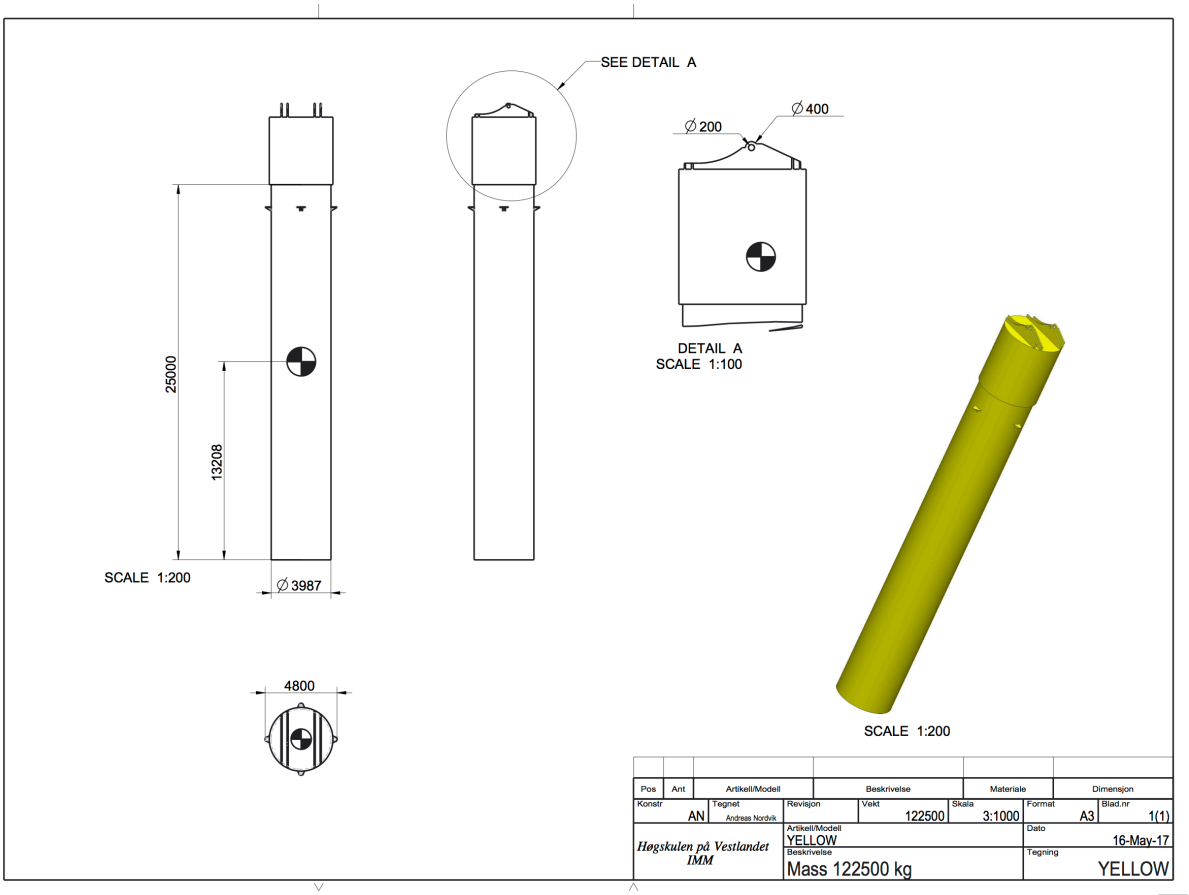
[9] Murakami, H., Rios, O., Amini, A., 2015, "A mathematical model with preliminary experiments of a gyroscopic ocean wave energy converter," *Proceedings of the ASME 2015 International Mechanical Engineering Congress & Exposition*, paper IMECE2015-51163.

APPENDIX A

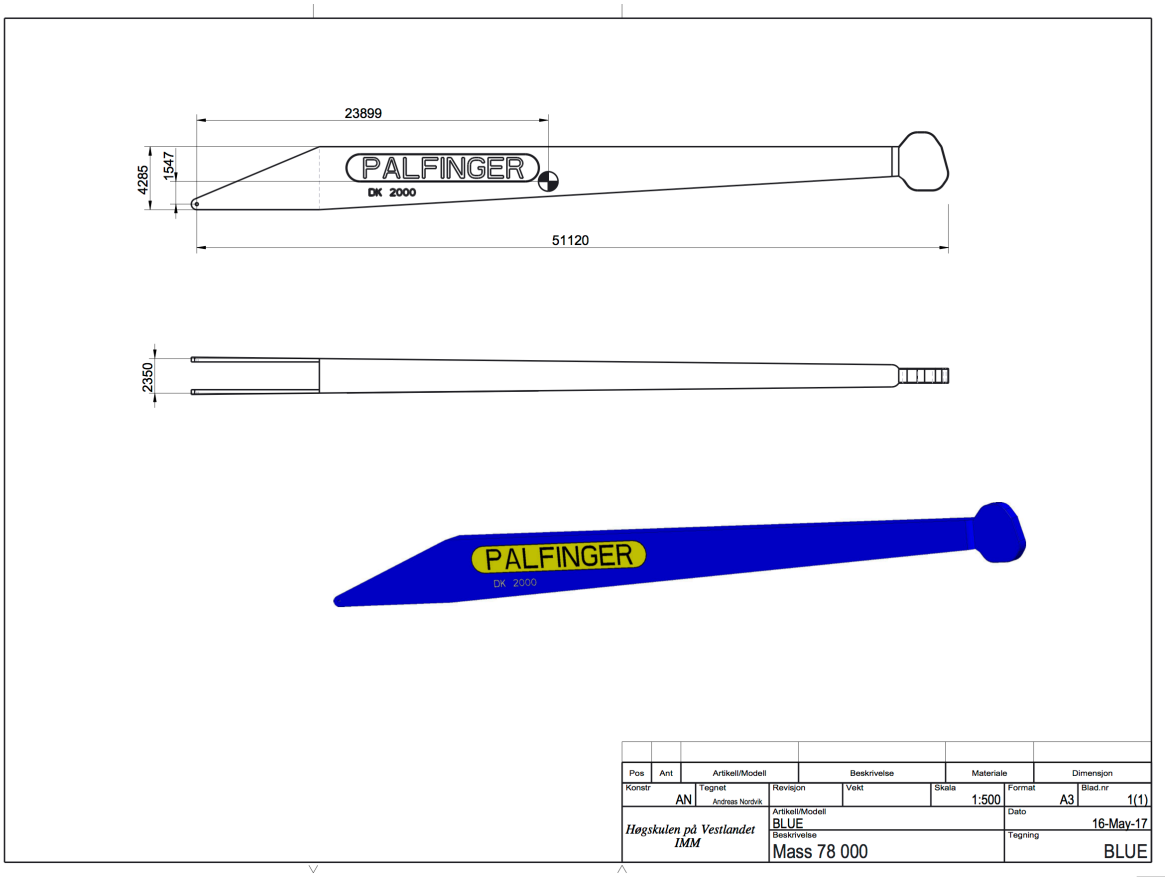
Stiff Boom Crane – Palfinger



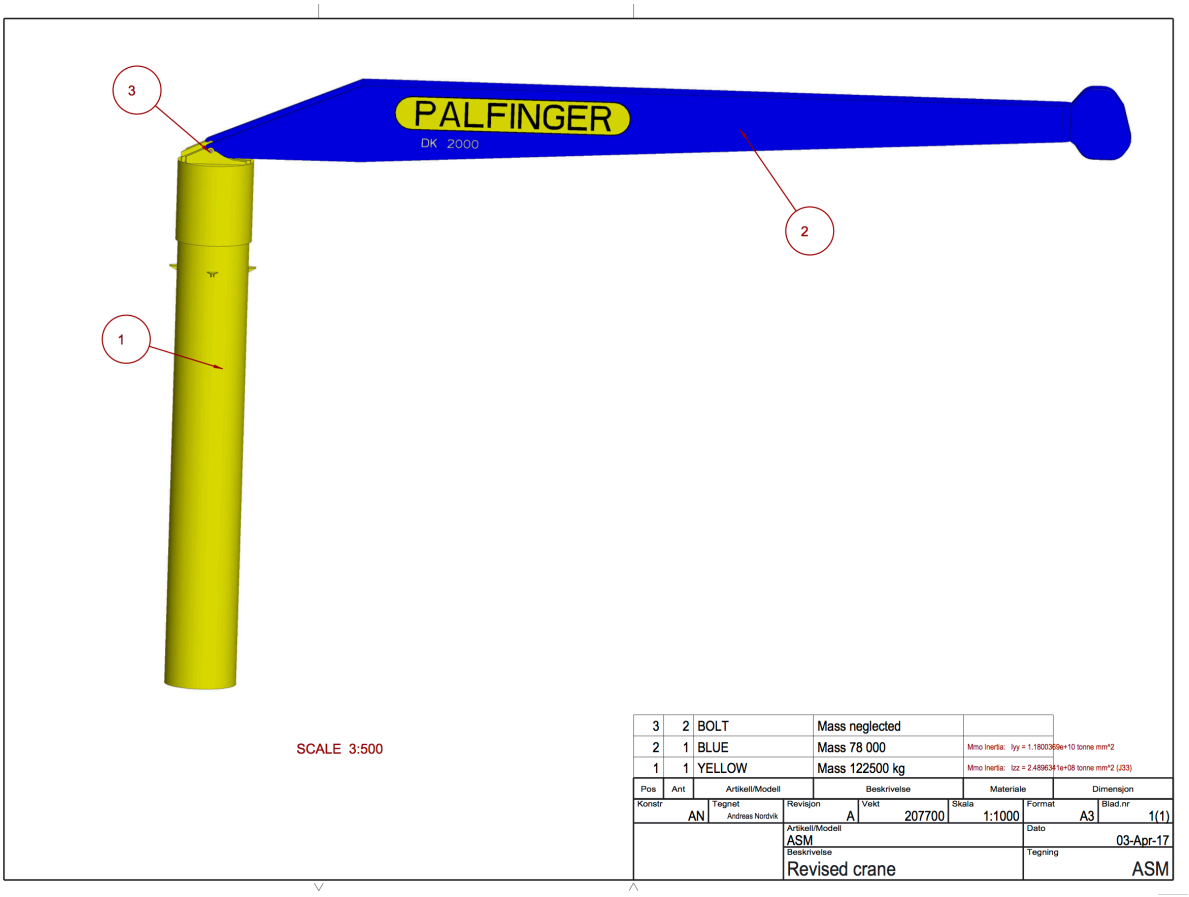
The crane drawn in CreoParametric 3.0:



The boom drawn in CreoParametric 3.0:



Assembly: The stiff boom crane drawn in CreoParametric 3.0:



APPENDIX B

$$\boldsymbol{B_{52}} = \boldsymbol{R^{(1)}(t)R^{(3/1)}(t)}\left(\boldsymbol{\overleftrightarrow{s_C^{(3/2)}}^T}\right)\left(\boldsymbol{R^{(3/1)}(t)}\right)^{\boldsymbol{T}} +$$

$$\boldsymbol{R^{(1)}(t)R^{(2/1)}(t)}\left(\boldsymbol{\overleftrightarrow{s_C^{(2/1)}}^T}\right)\left(\boldsymbol{R^{(2/1)}(t)}\right)^{\boldsymbol{T}} + \boldsymbol{R^{(1)}(t)}\left(\boldsymbol{\overleftrightarrow{s_C^{(2/1)}}^T}\right).$$

$$\boldsymbol{B_{53}} = R^{(1)}(t)R^{(3/1)}(t)\left(\overleftrightarrow{s_C^{(3/2)}}^T\right)\left(R^{(3/2)}(t)\right)^T e_3 + R^{(1)}(t)R^{(2/1)}(t)\left(\overleftrightarrow{s_C^{(2/1)}}^T\right)e_3$$

$$\boldsymbol{B_{54}} = R^{(1)}(t)R^{\left(\frac{3}{1}\right)}(t)\left(\overleftrightarrow{s_C^{\left(\frac{3}{2}\right)}}^T\right)e_2$$

APPENDIX C

The following equations provides angular velocity about the third, first and second axis, respectively. Admittedly, this was an overwrought attempt—to obtain expressions for the three angular velocities in this form using a Matlab symbolic Manipulator. In future work, it would be best to conduct a numerical integration directly with Eqn. (62).

$\dot{\omega}_1^{(1)}$ = (0.5*(2*J12^2*J31*ω2*ω3 - 2*J12*J13*J31*ω2*ω3 +
2*J12*J31*J33*ω2*ω3 - 2*J13*J31*J33*ω2*ω3 + J12*J31^2*ω1*ω3*sin(2*φ(t)) -
2*J12*J31^2*ω2*cos(φ(t))^2*dφ(t) - 2*J12*J31^2*cos(φ(t))*dφ(t)*dζ(t) +
J12*J31^2*ω1*dφ(t)*sin(2*φ(t)) - 2*J12*J31^2*ω2*ω3*cos(φ(t))^2 -
2*J12*J31^2*ω3*cos(φ(t))*dζ(t) - 2*J12^2*J31*ω2*ω3*cos(ζ(t))^2 +
2*J12^2*J33*ω2*ω3*cos(ζ(t))^2 + J11*J31*J33*ω1*ω3*sin(2*φ(t)) -
J12*J31*J32*ω1*ω3*sin(2*φ(t)) -J13*J31*J33*ω1*ω3*sin(2*φ(t)) +
2*J12*J31*J32*ω2*cos(φ(t))^2*dφ(t) - 2*J12*J31*J33*ω2*cos(φ(t))^2*dφ(t) +
2*J12*J31*J32*cos(φ(t))*dφ(t)*dζ(t) + 2*J12*J31*J33*cos(φ(t))*dφ(t)*dζ(t) -
J12*J31*J32*ω1*dφ(t)*sin(2*φ(t)) + J12*J31*J33*ω1*dφ(t)*sin(2*φ(t)) +
2*J12*J31^2*ω2*ω3*cos(φ(t))^2*cos(ζ(t))^2 -
2*J12*J33^2*ω2*ω3*cos(φ(t))^2*cos(ζ(t))^2 +
2*J12*J31^2*ω3*cos(φ(t))*cos(ζ(t))^2*dζ(t) -
2*J12*J33^2*ω3*cos(φ(t))*cos(ζ(t))^2*dζ(t) +
2*J12*J31*J32*ω2*ω3*cos(φ(t))^2 - 2*J12*J31*J33*ω2*ω3*cos(φ(t))^2 +
2*J13*J31*J33*ω2*ω3*cos(φ(t))^2 + 2*J12*J31*J32*ω3*cos(φ(t))*dζ(t) +
2*J12*J31*J33*ω3*cos(φ(t))*dζ(t) + 2*J12*J13*J31*ω2*ω3*cos(ζ(t))^2 -
2*J12*J13*J33*ω2*ω3*cos(ζ(t))^2 +
2*J12*J31^2*ω2*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) -
2*J12*J33^2*ω2*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) +
2*J12*J31^2*cos(φ(t))*cos(ζ(t))^2*dφ(t)*dζ(t) -
2*J12*J33^2*cos(φ(t))*cos(ζ(t))^2*dφ(t)*dζ(t) -
2*J12*J31^2*ω1*ω2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) +
2*J12*J33^2*ω1*ω2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) -
2*J12*J31*J32*ω2*ω3*cos(φ(t))^2*cos(ζ(t))^2 +
2*J12*J32*J33*ω2*ω3*cos(φ(t))^2*cos(ζ(t))^2 -
2*J12*J31*J32*ω3*cos(φ(t))*cos(ζ(t))^2*dζ(t) +
2*J12*J32*J33*ω3*cos(φ(t))*cos(ζ(t))^2*dζ(t) -
2*J12*J31^2*ω1*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) +
2*J12*J33^2*ω1*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) +
4*J12*J31^2*ω1*ω2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
4*J12*J33^2*ω1*ω2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
2*J12*J31^2*ω1^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
2*J12*J31^2*ω2^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
2*J12*J33^2*ω1^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J12*J33^2*ω2^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J12*J31*J32*ω2*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) +
2*J12*J32*J33*ω2*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) -
2*J12*J31*J32*cos(φ(t))*cos(ζ(t))^2*dφ(t)*dζ(t) +
2*J12*J32*J33*cos(φ(t))*cos(ζ(t))^2*dφ(t)*dζ(t) -
2*J12*J31^2*ω1*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) +

2*J12*J33^2*ω1*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) +
2*J12*J31^2*ω1*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) -
2*J12*J33^2*ω1*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) +
2*J12*J31*J32*ω1*ω2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) -
2*J12*J32*J33*ω1*ω2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) +
2*J12*J31*J32*ω1*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) -
2*J12*J32*J33*ω1*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) -
4*J12*J31*J32*ω1*ω2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) +
4*J12*J32*J33*ω1*ω2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) +
2*J12*J31*J32*ω1^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J12*J31*J32*ω2^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J12*J32*J33*ω1^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
2*J12*J32*J33*ω2^2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
2*J12*J31*J32*ω1*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) -
2*J12*J32*J33*ω1*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) -
2*J12*J31*J32*ω1*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) +
2*J12*J32*J33*ω1*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) +
2*J12*J31^2*ω2*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t) -
2*J12*J33^2*ω2*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t) -
2*J12*J31*J32*ω2*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t) +
2*J12*J32*J33*ω2*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t)))/(J11*J12*J31 +
J11*J31*J33 -J11*J31*J33*cos(φ(t))^2 + J12*J31*J33*cos(φ(t))^2 -
J11*J12*J31*cos(ζ(t))^2 + J11*J12*J33*cos(ζ(t))^2)

$\dot{\omega}_2^{(1)}$ = (0.5*(2*J11*J31^2*ω1*ω3 - 2*J11^2*J31*ω1*ω3 + 2*J11*J31^2*ω1*dφ(t)
+ 2*J11*J13*J31*ω1*ω3 - 2*J11*J31*J32*ω1*ω3 -J11*J31^2*ω2*ω3*sin(2*φ(t))
- 2*J11*J31^2*ω1*cos(φ(t))^2*dφ(t) - 2*J11*J31^2*ω1*cos(ζ(t))^2*dφ(t) +
2*J11*J33^2*ω1*cos(ζ(t))^2*dφ(t) - 2*J11*J31^2*sin(φ(t))*dφ(t)*dζ(t) -
2*J11*J31*J32*ω1*dφ(t) + 2*J11*J31*J33*ω1*dφ(t) -
J11*J31^2*ω2*dφ(t)*sin(2*φ(t)) + J11*J31^2*ω2*sin(2*ζ(t))*dζ(t) -
J11*J33^2*ω2*sin(2*ζ(t))*dζ(t) - 2*J11*J31^2*ω1*ω3*cos(φ(t))^2 -
2*J11*J31^2*ω1*ω3*cos(ζ(t))^2 + 2*J11^2*J31*ω1*ω3*cos(ζ(t))^2 +
2*J11*J33^2*ω1*ω3*cos(ζ(t))^2 - 2*J11^2*J33*ω1*ω3*cos(ζ(t))^2 -
2*J11*J31^2*ω3*sin(φ(t))*dζ(t) +
2*J11*J31^2*cos(ζ(t))^2*sin(φ(t))*dφ(t)*dζ(t) -
2*J11*J33^2*cos(ζ(t))^2*sin(φ(t))*dφ(t)*dζ(t) +
J11*J31*J32*ω2*ω3*sin(2*φ(t)) -J12*J31*J33*ω2*ω3*sin(2*φ(t)) +
J13*J31*J33*ω2*ω3*sin(2*φ(t)) -
2*J11*J31^2*ω1^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) +
2*J11*J31^2*ω2^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) +
2*J11*J33^2*ω1^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) -
2*J11*J33^2*ω2^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) +
2*J11*J31*J32*ω1*cos(φ(t))^2*dφ(t) - 2*J11*J31*J33*ω1*cos(φ(t))^2*dφ(t) +
2*J11*J31*J32*ω1*cos(ζ(t))^2*dφ(t) - 2*J11*J32*J33*ω1*cos(ζ(t))^2*dφ(t) +
2*J11*J31*J32*sin(φ(t))*dφ(t)*dζ(t) + 2*J11*J31*J33*sin(φ(t))*dφ(t)*dζ(t) +
J11*J31*J32*ω2*dφ(t)*sin(2*φ(t)) -J11*J31*J33*ω2*dφ(t)*sin(2*φ(t)) +
2*J11*J31^2*ω1^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
2*J11*J31^2*ω2^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
2*J11*J33^2*ω1^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) +

2*J11*J33^2*ω2^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
J11*J31*J32*ω2*sin(2*ζ(t))*dζ(t) + J11*J32*J33*ω2*sin(2*ζ(t))*dζ(t) +
2*J11*J31^2*ω1*ω3*cos(φ(t))^2*cos(ζ(t))^2 -
2*J11*J33^2*ω1*ω3*cos(φ(t))^2*cos(ζ(t))^2 +
2*J11*J31^2*ω3*cos(ζ(t))^2*sin(φ(t))*dζ(t) -
2*J11*J33^2*ω3*cos(ζ(t))^2*sin(φ(t))*dζ(t) +
2*J11*J31*J32*ω1*ω3*cos(φ(t))^2 - 2*J11*J31*J33*ω1*ω3*cos(φ(t))^2 +
2*J13*J31*J33*ω1*ω3*cos(φ(t))^2 - 2*J11*J13*J31*ω1*ω3*cos(ζ(t))^2 +
2*J11*J13*J33*ω1*ω3*cos(ζ(t))^2 + 2*J11*J31*J32*ω1*ω3*cos(ζ(t))^2 -
2*J11*J32*J33*ω1*ω3*cos(ζ(t))^2 + 2*J11*J31*J32*ω3*sin(φ(t))*dζ(t) +
2*J11*J31*J33*ω3*sin(φ(t))*dζ(t) +
2*J11*J31^2*ω1*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) -
2*J11*J33^2*ω1*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) +
2*J11*J31*J32*ω1^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) -
2*J11*J31*J32*ω2^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) -
2*J11*J32*J33*ω1^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) +
2*J11*J32*J33*ω2^2*cos(φ(t))*cos(ζ(t))*sin(ζ(t)) -
2*J11*J31^2*ω1*ω2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
2*J11*J33^2*ω1*ω2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J11*J31*J32*ω1^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) +
2*J11*J31*J32*ω2^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) +
2*J11*J32*J33*ω1^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
2*J11*J32*J33*ω2^2*cos(φ(t))^3*cos(ζ(t))*sin(ζ(t)) -
2*J11*J31*J32*ω1*ω3*cos(φ(t))^2*cos(ζ(t))^2 +
2*J11*J32*J33*ω1*ω3*cos(φ(t))^2*cos(ζ(t))^2 +
2*J11*J31^2*ω2*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) -
2*J11*J33^2*ω2*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) -
2*J11*J31*J32*ω3*cos(ζ(t))^2*sin(φ(t))*dζ(t) +
2*J11*J32*J33*ω3*cos(ζ(t))^2*sin(φ(t))*dζ(t) -
2*J11*J31*J32*ω1*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) +
2*J11*J32*J33*ω1*cos(φ(t))^2*cos(ζ(t))^2*dφ(t) +
2*J11*J31^2*ω2*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) -
2*J11*J33^2*ω2*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) -
2*J11*J31*J32*cos(ζ(t))^2*sin(φ(t))*dφ(t)*dζ(t) +
2*J11*J32*J33*cos(ζ(t))^2*sin(φ(t))*dφ(t)*dζ(t) -
2*J11*J31^2*ω2*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) +
2*J11*J33^2*ω2*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) +
2*J11*J31*J32*ω1*ω2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J11*J32*J33*ω1*ω2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J11*J31*J32*ω2*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) +
2*J11*J32*J33*ω2*ω3*cos(φ(t))*cos(ζ(t))^2*sin(φ(t)) -
2*J11*J31*J32*ω2*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) +
2*J11*J32*J33*ω2*cos(φ(t))*cos(ζ(t))^2*sin(φ(t))*dφ(t) +
2*J11*J31*J32*ω2*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) -
2*J11*J32*J33*ω2*cos(φ(t))^2*cos(ζ(t))*sin(ζ(t))*dζ(t) +
4*J11*J31^2*ω1*ω2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
4*J11*J33^2*ω1*ω2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
2*J11*J31^2*ω1*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t) -
2*J11*J33^2*ω1*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t) -

4*J11*J31*J32*ω1*ω2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) +
4*J11*J32*J33*ω1*ω2*cos(φ(t))^2*cos(ζ(t))*sin(φ(t))*sin(ζ(t)) -
2*J11*J31*J32*ω1*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t) +
2*J11*J32*J33*ω1*cos(φ(t))*cos(ζ(t))*sin(φ(t))*sin(ζ(t))*dζ(t)))/(J11*J12*J31 +
J11*J31*J33 - J11*J31*J33*cos(φ(t))^2 + J12*J31*J33*cos(φ(t))^2 -
J11*J12*J31*cos(ζ(t))^2 + J11*J12*J33*cos(ζ(t))^2)

$\dot{\omega}_3^{(1)}$ = (ω1*ω2*(J11 - J12))/J13

APPENDIX D

Matlab code – CraneOnShip

```
function craneOnShip()
tic % tic starts the timer

%{
BODIES:
Ship: 1
Crane: 2
Boom: 3

Declaration of symbolic variables.
All components of matrices and vectors has to be
defined as symbolic
variables.

Symbolic variables can be defined as dependent
variable i.e. time dependent
variables. This can be used for differentiation.

Declaration of symbolic variable:       syms
<varName>
Declaration of dependent sym variable:  syms
<varName(var)>
%}
syms t
syms phi3(t)  dphi3(t) ddphi3(t)
syms zeta2(t) dzeta2(t) ddzeta2(t)

syms w1 w2 w3 dw1 dw2 dw3

syms dx1 dx2 dx3 ddx1 ddx2 ddx3

syms R1_11(t) R1_12(t) R1_13(t)
syms R1_21(t) R1_22(t) R1_23(t)
syms R1_31(t) R1_32(t) R1_33(t)

syms h21 h22 h23
syms d31 d32 d33

% Vectors and matrices containing non-symbolic
elements should be defined
% by using the sym function.
I3 = sym(eye(3));
```

```
% e2 and e3 are unit vectors for the two and three
axis. They are both used
% to simplify notation.
e2 = sym(zeros(3,1));
e2(2) = 1;
e3 = sym(zeros(3,1));
e3(3) = 1;

% Pescribed rates:

%Turret
% Angular acceleration is equal to zero
ddphi3(t) = 0;
% Angular velocity:
dphi3(t) = 0.15; %rad/s --> v=v0+at where a=0--> v=v0
aka pescribed rate.

%Position: x=x0+v0t+1/2at^2 --> x=v0t (and if
nessesary, an initial
%condition is set x=x0+v0t) This is ploted in the
Rotation matrices above:
phi3(t) = dphi3(t)*t;

%Boom
% Angular acceleration is equal to zero
ddzeta2(t) = 0;

% Angular velocity:
A = 1.414; %81*3.1416/180; %Amplitude deg*pi/180
f = 0.05; %frequency
w = 2*3.1416*f; %Angular velocity 2*pi*f
dzeta2(t) = A*sin(w*t); %rad/s --> v=v0+at where a=0-
-> v=v0 aka pescribed rate.

%Position: x=x0+v0t+1/2at^2 --> x=v0t (and if
nessesary, an initial
%condition is set x=x0+v0t) This is ploted in the
Rotation matrices above:
zeta2(t) = -A*cos(w*t)/t;

% Rotation matrices are defined below
% Ship
R1 = [ R1_11 R1_12 R1_13
    R1_21 R1_22 R1_23
    R1_31 R1_32 R1_33 ];

% Crane from ship (z-axis)
R21 = sym(zeros(3));
```

```
R21(1,1) =  cos(phi3(t));
R21(1,2) =  -sin(phi3(t));
R21(2,1) =  sin(phi3(t));
R21(2,2) =  cos(phi3(t));
R21(3,3) =  1;

% Boom from crane (x-axis)
R32 = sym(zeros(3));
R32(1,1) =  cos(zeta2(t));
R32(1,3) = sin(zeta2(t));
R32(2,2) =  1;
R32(3,1) =  -sin(zeta2(t));
R32(3,3) =  cos(zeta2(t));

% Boom from ship
R31 = R21*R32;

% Distances from the ship CM to the crane CM. These
distances are assumed
% to be zero.
h21 = 0;
h22 = 0;
h23 = 0; % heigth to the crane's CM is 13208
millimeters

h2_1 = sym(zeros(3,1));
h2_1(1) = h21;
h2_1(2) = h22;
h2_1(3) = h23;

%Distances from the boom CM to the crane CM.
d31 = 0;
d32 = 0; % outreach of 23.899 meters
d33 = 0; % heigth to top of crane is 17.012 meters

d3_2 = sym(zeros(3,1));
d3_2(1) = d31;
d3_2(2) = d32;
d3_2(3) = d33;

% The distance vectors are skewed using symSkew
(defined below). These
% matrices are used in the B matrix.
s21Skew = symSkew(h2_1);
s32Skew = symSkew(d3_2);

% B matrix construction
B = sym(zeros(18, 8));
```

```
B(1:3,   1:3) = I3;
B(4:6,   4:6) = I3;
B(7:9,   1:3) = I3;
B(7:9,   4:6) = R1*s21Skew.';
B(10:12, 4:6) = R21.';
B(10:12, 7)   = e3;
B(13:15, 1:3) = I3;
B(13:15, 4:6) = R1*R31*s32Skew.'*R31.' +
R1*R21*s21Skew.'*R21.' + R1*s21Skew.';
B(13:15, 7)   = R1*R31*s32Skew.'*R32.'*e3 +
R1*R21*s21Skew.'*e3;
B(13:15, 8)   = R1*R31*s32Skew.'*e2;
B(16:18, 4:6) = R31.';
B(16:18, 7)   = R32.'*e3;
B(16:18, 8)   = e2;
Bn = simplify(B); %We simplify to help MatLab solve
it faster.

% The operator .' transposes symbolic matrices and
vectors. Using the '
% operator computes complex conjugate transposes.
Bt = Bn.';

% Differentiating of the B matrix with respect to
time using diff
% and simplification using simplify.
dB = diff(Bn,t);

% Using the subs function one can replace symbolic
expressions.
% Below 'diff(phi2(t),t)' are replaced with
'dphi2(t)' and so on.
dB = subs(dB,diff(phi3(t), t), dphi3(t));
dB = subs(dB,diff(zeta2(t), t),dzeta2(t));

% Defining the omegas
w1b = sym(zeros(3,1));
w1b(1) = w1;
w1b(2) = w2;
w1b(3) = w3;
Omega1 = symSkew(w1b);

O2 = sym(zeros(3,1));
O2 = R21.'*w1b+dphi3(t)*e3;
Omega2 = symSkew(O2);

O3 = sym(zeros(3,1));
O3 = R31.'*w1b+R32.'*dphi3(t)*e3+dzeta2(t)*e2;
```

```
Omega3 = symSkew(O3);

% D matrix construction
D = sym(zeros(18));
D(4:6,4:6)     = Omega1(1:3,1:3);
D(10:12,10:12) = Omega2(1:3,1:3);
D(16:18,16:18) = Omega3(1:3,1:3);

% dR = R*omegaskew
dR1 = R1*Omega1;

dB = subs(dB,diff(R1, t),dR1);

% M matrix construction
M = sym(zeros(18));

%Ship J11 ~= J12 -- The mass of the ship is assumed,
and the inertia is
%calculated like a cuboid.
%The inertia of the ship has to be different numbers
M(1,1) = 4100; %tonne
M(2,2) = 4100; %tonne
M(3,3) = 4100; %tonne

wi = 22; %m
h = 10;  %m
d = 85; %m

M(4,4) = 1/12*M(1,1)*(h^2+d^2);  %tonne m^2
M(5,5) = 1/12*M(2,2)*(wi^2+d^2); %tonne m^2
M(6,6) = 1/12*M(3,3)*(wi^2+h^2); %tonne m^2

% Turret all J's are zero
M(7,7) = 0;
M(8,8) = 0;
M(9,9) = 0;

M(10,10) = 0;
M(11,11) = 0;
M(12,12) = 0;

% Boom J31 ~= J32 ~= J33
M(13,13) = 78+3.8+3.4; %tonne
M(14,14) = 78+3.8+3.4; %tonne
M(15,15) = 78+3.8+3.4; %tonne

%The inertis is calculated using Creo
M(16,16) = 1.1763498e+5; %tonne m^2
```

```
M(17,17) = 1.1800369e+7; %tonne m^2
M(18,18) = 1.1739636e+7; %tonne m^2

%Assumptions:
%text.

% Vector for the generalized velocities, to be
multiplied by N*.
dq = sym(zeros(8,1));
dq(1)  = dx1;
dq(2)  = dx2;
dq(3)  = dx3;
dq(4)  = w1;
dq(5)  = w2;
dq(6)  = w3;
dq(7)  = dphi3(t);
dq(8)  = dzeta2(t);

% Vector for the generalized acceleration, to be
multiplied by M*.
ddq = sym(zeros(8,1));
ddq(1)  = ddx1;
ddq(2)  = ddx2;
ddq(3)  = ddx3;
ddq(4)  = dw1;
ddq(5)  = dw2;
ddq(6)  = dw3;
ddq(7)  = ddphi3(t);
ddq(8)  = ddzeta2(t);

%{
Building M*, N* -> See Chapter 13 of Murakami's and
Impelluso's
textbook.
Equation to be solved for generalized accelerations
(ddq):
(M*)(ddq) + (N*)(dq) = 0 --> ddq = inv(M*)[-(N*)(dX)]
%}
Mstar  = simplify((Bt*M*Bn)); %Mstar
Nstar  = simplify((Bt*(D*M*Bn + M*dB))); %Nstar
%{
There are two ways of solving the equations by using
the inverse in MATLAB.
The following operations produces the same result:
1: Ax = b --> x = inv(A)b
2: Ax = b --> x = A\b.
```

```
MATLAB advices to use the second method, however in
this case and on this machine
the second method seemed to be the fastest.
%}
fprintf('%4.2f seconds. Inverse of Mstar has
started.\n',toc);
invMstar = inv(Mstar);
fprintf('%4.2f seconds. Inverse of Mstar has
ended.\n',toc);
%{
Solving the equation is done below. The inbuilt
simplify function is used
to simplify the input argument and make it more
readable.

NOTE:
Using the simplify function is time consuming and
should be avoided unless
one wants to print something to the command window or
a file.
In this script, we simplify 1000 times.
%}
ddq = simplify(invMstar*(-Nstar*dq));

fprintf('%4.2f seconds. ddq has been
calculated.\n',toc);

parfor i = 4:6;
    ddq(i) = simplify(ddq(i),'Steps',1000);
end
%{
ddq(1) = ddx1
ddq(2) = ddx2
ddq(3) = ddx3
ddq(4) = dw1  !
ddq(5) = dw2  !
ddq(6) = dw3  !
ddq(7) = ddphi3
ddq(8) = ddzeta2

In this problem the equations of interest are the
ones for the angular
accelerations of the ship (ddw1, ddw2, ddw3).
Vpa, gives the option of choosing how many
significant numbers to print.
%}

disp(vpa(ddq(4),3));
```

```
disp(vpa(ddq(5),3));
disp(vpa(ddq(6),3));
toc % Stop timer
end

function [Mat] = symSkew(vec)
% symSkew skews a symbolic 3-element vector (input)
to  a 3x3 skew
% symmetric symbolic matrix (output).
    Mat = sym(zeros(3));

    Mat(1,2) = -vec(3);
    Mat(1,3) =  vec(2);
    Mat(2,3) = -vec(1);

    Mat(2,1) = -Mat(1,2);
    Mat(3,1) = -Mat(1,3);
    Mat(3,2) = -Mat(2,3);
End
```

Matlab code – GetOmegaRK4

```
function getOmegaRK4()

%{
To accommodate for changing angular velocities, the
reconstruction can be
performed at every time step of the Runge-Kutta
integration, assuming the
assuming the angular velocity stays constant from t
to t + delta-t.
%}
    dt = 0.05; % Small timestep
```

```
    t = 0.01:dt:150; % Timevector to 150 secounds.

    % Declaration of omegas as a vector of zeros.
    w1 = zeros(1,length(t));
    w2 = zeros(1,length(t));
    w3 = zeros(1,length(t));

for i = 2:(length(t)) % i counter starts at 2 and has
the same numbers of elements as timevector

     K1 = f1( t(i-1), w1(i-1), w2(i-1), w3(i-1))*dt;
     L1 = f2( t(i-1), w1(i-1), w2(i-1), w3(i-1))*dt;
     M1 = f3( t(i-1), w1(i-1), w2(i-1), w3(i-1))*dt;

     K2 = f1( t(i-1), (w1(i-1) + K1/2) ,(w2(i-1)+
L1/2), (w3(i-1)+ M1/2))*dt;
     L2 = f2( t(i-1), (w1(i-1) + K1/2) ,(w2(i-1)+
L1/2), (w3(i-1)+ M1/2))*dt;
     M2 = f3( t(i-1), (w1(i-1) + K1/2) ,(w2(i-1)+
L1/2), (w3(i-1)+ M1/2))*dt;

     K3 = f1( t(i-1), (w1(i-1) + K2/2) ,(w2(i-1)+
L2/2), (w3(i-1)+ M2/2))*dt;
     L3 = f2( t(i-1), (w1(i-1) + K2/2) ,(w2(i-1)+
L2/2), (w3(i-1)+ M2/2))*dt;
     M3 = f3( t(i-1), (w1(i-1) + K2/2) ,(w2(i-1)+
L2/2), (w3(i-1)+ M2/2))*dt;

     K4 = f1( t(i-1), (w1(i-1) + K3) ,(w2(i-1)+ L3),
(w3(i-1)+ M3))*dt;
     L4 = f2( t(i-1), (w1(i-1) + K3) ,(w2(i-1)+ L3),
(w3(i-1)+ M3))*dt;
     M4 = f3( t(i-1), (w1(i-1) + K3) ,(w2(i-1)+ L3),
(w3(i-1)+ M3))*dt;

     w1(i) = w1(i-1) + (K1 + 2*K2 + 2*K3 + K4)/6;
     w2(i) = w2(i-1) + (L1 + 2*L2 + 2*L3 + L4)/6;
     w3(i) = w3(i-1) + (M1 + 2*M2 + 2*M3 + M4)/6;

    % This is the nature of runge kutta fourth order
end
% Boom motion
A = 1.414; %81*3.1416/180; %Amplitude deg*pi/180
f = 0.05; %frequency
w = 2*3.1416*f; %Angular velocity 2*pi*f
dzeta2 = A*sin(w*t); %The exitation of the Boom

    figure(1);                  % Figure number
```

```
    hold on                         % Holds on to the
plot to plot the next one
    plot(t, w1,'k');                % Plots omega1 with
black solid line
    plot(t,dzeta2/20,'-.k');        % Plots the exitation
of the Boom
    ylabel('Omega 1 [rad/s]');      % Makes Label on y-
axis
    xlabel('time [s]');             % Makes Label on x-
axis
    title(['\fontsize{22}Omega 1, Roll']); %Title of
plot
    hold off

    figure(2);
    hold on
    plot(t, w2,'k');                %Plots omega2
    plot(t,dzeta2/20,'-.k');
    ylabel('Omega 2 [rad/s]');
    xlabel('time [s]');
    title(['\fontsize{22}Omega 2, Pitch']);
    hold off

    figure(3);
    hold on
    plot(t, w3,'k');
    plot(t,dzeta2/20,'-.k');
    ylim([-0.15001 0.15001])
    ylabel('Omega 3 [rad/s]');
    xlabel('time [s]');
    title(['\fontsize{22}Omega 3, Yaw']);
    hold off
end

function f1 = f1(t,w1,w2,w3)

    % This is the function for omega1 which comes
from craneOnShip.m sctipt

    f1 = (1.16e-16*(1.71e42*w2*w3 ...
        + 6.38e41*cos(0.15*t)*sin(0.314*t) ...
        - 1.1e39*w2*cos(0.15*t)^2 ...
        + 3.09e43*w2*w3*cos((1.41*cos(0.314*t))/t)^2
...
        + 1.07e41*w2*w3*cos(0.15*t)^2 ...
        -
1.53e41*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t)*sin(
0.314*t) ...
```

```
        -
1.08e41*w2*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t)^2
...
        + 1.1e39*w1*cos(0.15*t)*sin(0.15*t) ...
        + 4.25e42*w3*cos(0.15*t)*sin(0.314*t) ...
        -
7.22e41*w2*w3*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t
)^2 ...
        - 1.82e41*w1*w3*cos(0.15*t)*sin(0.15*t) ...
        +
1.08e41*w1*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t)*s
in(0.15*t) ...
        -
1.02e42*w3*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t)*s
in(0.314*t) ...
        +
1.44e42*w1*w2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^
3*sin((1.41*cos(0.314*t))/t) ...
        +
7.22e41*w1*w3*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t
)*sin(0.15*t) ...
        +
1.02e42*w1*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^2*s
in((1.41*cos(0.314*t))/t)*sin(0.314*t) ...
        -
7.22e41*w1^2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^2
*sin((1.41*cos(0.314*t))/t)*sin(0.15*t) ...
        +
7.22e41*w2^2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^2
*sin((1.41*cos(0.314*t))/t)*sin(0.15*t) ...
        -
7.22e41*w1*w2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)*
sin((1.41*cos(0.314*t))/t) ...
        +
1.02e42*w2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)*sin
((1.41*cos(0.314*t))/t)*sin(0.15*t)*sin(0.314*t)))/(3
.7e27*cos((1.41*cos(0.314*t))/t)^2 ...
        + 8.75e24*cos(0.15*t)^2 ...
        + 2.04e26);
 end

function f2 = f2(t,w1,w2,w3)

    % This is the function for omega2 which comes
from craneOnShip.m sctipt

     f2 = -(2.91e-15*(6.83e40*w1*w3 ...
         - 4.17e37*w1 ...
```

```
        - 2.43e40*sin(0.15*t)*sin(0.314*t) ...
        - 4.12e39*w1*cos((1.41*cos(0.314*t))/t)^2
...
        + 4.17e37*w1*cos(0.15*t)^2 ...
        - 1.62e41*w3*sin(0.15*t)*sin(0.314*t) ...
        + 1.08e42*w1*w3*cos((1.41*cos(0.314*t))/t)^2
...
        - 4.29e39*w1*w3*cos(0.15*t)^2 ...
        +
5.82e39*cos((1.41*cos(0.314*t))/t)^2*sin(0.15*t)*sin(
0.314*t) ...
        +
4.12e39*w1*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t)^2
...
        + 4.17e37*w2*cos(0.15*t)*sin(0.15*t) ...
        +
2.75e40*w1*w3*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t
)^2 ...
        -
2.75e40*w1^2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^3
*sin((1.41*cos(0.314*t))/t) ...
        +
2.75e40*w2^2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^3
*sin((1.41*cos(0.314*t))/t) ...
        - 1.29e39*w2*w3*cos(0.15*t)*sin(0.15*t) ...
        -
3.88e40*w2*cos((1.41*cos(0.314*t))/t)*sin((1.41*cos(0
.314*t))/t)*sin(0.314*t) ...
        +
2.75e40*w1^2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)*s
in((1.41*cos(0.314*t))/t) ...
        -
2.75e40*w2^2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)*s
in((1.41*cos(0.314*t))/t) ...
        +
4.12e39*w2*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t)*s
in(0.15*t) ...
        +
3.88e40*w3*cos((1.41*cos(0.314*t))/t)^2*sin(0.15*t)*s
in(0.314*t) ...
        +
2.75e40*w2*w3*cos((1.41*cos(0.314*t))/t)^2*cos(0.15*t
)*sin(0.15*t) ...
        +
3.88e40*w2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^2*s
in((1.41*cos(0.314*t))/t)*sin(0.314*t) ...
        +
2.75e40*w1*w2*cos((1.41*cos(0.314*t))/t)*sin((1.41*co
```

```
s(0.314*t))/t)*sin(0.15*t) ...
         -
3.88e40*w1*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)*sin
((1.41*cos(0.314*t))/t)*sin(0.15*t)*sin(0.314*t) ...
         -
5.49e40*w1*w2*cos((1.41*cos(0.314*t))/t)*cos(0.15*t)^
2*sin((1.41*cos(0.314*t))/t)*sin(0.15*t)))/(3.7e27*co
s((1.41*cos(0.314*t))/t)^2 ...
         + 8.75e24*cos(0.15*t)^2 ...
         + 2.04e26);
 end

function f3 = f3(t,w1,w2,w3)

    % This is the function for omega1 which comes
from craneOnShip.m sctipt

    f3 = -0.658*w1*w2;
end
```

APPENDIX E

JavaScript code – Starter.js

```
function loadObjects(scene) {

    var loader = new THREE.OBJMTLLoader();
    o1 = loadObject(loader,1);
    o2 = loadObject(loader,2);
    o3 = loadObject(loader,3);

    var linelength = 6000;
    var material = new THREE.LineBasicMaterial({
        color: 0xff0000
    });
    var linegeo = new THREE.Geometry();
    linegeo.vertices.push(
        new THREE.Vector3(linelength, 0, 0),
        new THREE.Vector3(0, 0, 0)
    );
    var line = new THREE.Line(linegeo, material);

    scene.add(line);

    var material1 = new THREE.LineBasicMaterial({
        color: 0x00ff00
    });
    var linegeo1 = new THREE.Geometry();
    linegeo1.vertices.push(
        new THREE.Vector3(0, linelength, 0),
        new THREE.Vector3(0, 0, 0)
    );
    var line1 = new THREE.Line(linegeo1, material1);

    scene.add(line1);

    var material2 = new THREE.LineBasicMaterial({
        color: 0x0000ff
    });
    var linegeo2 = new THREE.Geometry();
    linegeo2.vertices.push(
        new THREE.Vector3(0, 0, linelength),
        new THREE.Vector3(0, 0, 0)
    );
    var line2 = new THREE.Line(linegeo2, material2);

    scene.add(line2);
```

```
    var frames = [];

    frames[0] = new THREE.Object3D(0, 0, 0);
    frames[1] = new THREE.Object3D(0, 0, 0);
    frames[1].position.y = 0;
    frames[1].position.z = 0;
    frames[2] = new THREE.Object3D(0, 0, 0);
    frames[2].position.z = 5500;
    frames[3] = new THREE.Object3D(0, 0, 0);
    frames[3].position.y = 0;

    scene.add(frames[0]);
    frames[0].add(o1);
    frames[0].add(frames[1]);
    frames[1].add(o2);
    frames[1].add(frames[2]);
    frames[2].add(o3);
    return frames;
}

function loadObject(loader,num) {

    switch (num) {
       case 1:
            var o1 = new THREE.Object3D();
            loader.load('models/baat.obj',
'models/baat.mtl', function (object) {
o1.add(object); });
            o1.position.y = 0;
            o1.position.z = -650;
            o1.position.x = 0;
            o1.rotation.y = 0;
            o1.rotation.z = Math.PI / 2;
            o1.scale.x = 8000;
            o1.scale.y = 8000;
            o1.scale.z = 8000;
            return o1;

       case 2:
            var o2 = new THREE.Object3D();
            loader.load('models/crane.obj',
'models/crane.mtl', function (object) {
o2.add(object); });
            o2.rotation.z = Math.PI/2;
            o2.position.y = 0;
            o2.position.z = 800;
            o2.position.x = 0;
```

```
            o2.scale.x = 0.15;
            o2.scale.y = 0.15;
            o2.scale.z = 0.15;
            return o2;
        case 3:


            var o3 = new THREE.Object3D();
            loader.load('models/boom.obj',
'models/boom.mtl', function (object) {
o3.add(object); });

            o3.rotation.z = Math.PI / 2;

            o3.position.x = 0;
            o3.position.y = 800;
            o3.position.z = 0;

            o3.scale.x = 0.12;
            o3.scale.y = 0.12;
            o3.scale.z = 0.12;
            return o3;
    }
}

function generateHeight(width, height) {
        var size = width * height, data = new
Uint8Array(size),
        perlin = new ImprovedNoise(), quality = 1, z
= Math.random() * 5;
        for (var j = 0; j < 4; j++) {
            for (var i = 0; i < size; i++) {
                var x = i % width, y = ~~(i / width);
                data[i] += Math.abs(perlin.noise(x /
quality, y / quality, z) * quality * 1.75);
            }
            quality *= 1.2;
        }
        return data;
    }

    function generateTexture(data, width, height) {
        var canvas, canvasScaled, context, image,
imageData,
        level, diff, vector3, sun, shade;
        vector3 = new THREE.Vector3(0, 0, 0);
        sun = new THREE.Vector3(1, 1, 1);
```

```
        sun.normalize();
        canvas = document.createElement('canvas');
        canvas.width = width;
        canvas.height = height;
        context = canvas.getContext('2d');
        context.fillStyle = '#000';
        context.fillRect(0, 0, width, height);
        image = context.getImageData(0, 0,
canvas.width, canvas.height);
        imageData = image.data;
        for (var i = 0, j = 0, l = imageData.length;
i < l; i += 4, j++) {
            vector3.x = data[j - 2] - data[j + 2];
            vector3.y = 2;
            vector3.z = data[j - width * 2] - data[j
+ width * 2];
            vector3.normalize();
            shade = vector3.dot(sun);
            imageData[i] = (shade * 128) * (0.5 +
data[j] * 0.007);
            imageData[i + 1] = (32 + shade * 96) *
(0.5 + data[j] * 0.007);
            imageData[i + 2] = (shade * 96) * (0.5 +
data[j] * 0.007);
        }
        context.putImageData(image, 0, 0);
        // Scaled 4x
        canvasScaled =
document.createElement('canvas');
        canvasScaled.width = width * 4;
        canvasScaled.height = height * 4;
        context = canvasScaled.getContext('2d');
        context.scale(4, 4);
        context.drawImage(canvas, 0, 0);
        image = context.getImageData(0, 0,
canvasScaled.width, canvasScaled.height);
        imageData = image.data;
        for (var i = 0, l = imageData.length; i < l;
i += 4) {

            var v = ~~(Math.random() * 5);
            imageData[i] += v;
            imageData[i + 1] += v;
            imageData[i + 2] += v;
        }

        context.putImageData(image, 0, 0);
        return canvasScaled;
```

```
    }



    function gui() {
        var gui = new dat.GUI({
            height: 5 * 32 - 1
        });
        guiParams = {
            Crane: 0.3,
            Boom: 81/ 2 * Math.PI / 180,
            Boom2: 20

        };
        gui.add(guiParams, 'Crane',
0.1,2*Math.PI).name('Velocity crane');
        gui.add(guiParams, 'Boom',  0.1,
(81/2*Math.PI/180)).name('Amplitude');
        gui.add(guiParams, 'Boom2',
5,30).name('Period');
        return guiParams;
    }

    function period(seconds) {
        return (Math.PI * 2) / seconds;
    }
```

JavaScript code – Main.js

```
$(document).ready(function () {

    var camera, renderer, controls;
    var scene = new THREE.Scene();
    var frames = [];
    var dq = [0, 0, 0];
    var q = [0, 0, 0];
    var guiParams = gui();
    var t = 0.01;
    const dt = 0.016;
```

```
    function init() {
        setRenderer();
        frames = loadObjects(scene).slice();
        setCamera();
        setLights();
        OceanSurface();
        setControls();
        window.addEventListener('resize',
onWindowResize, false);
        renderer.render(scene, camera);
        animate();
    }

    function OceanSurface() {

    var geometry, points = [];
    var width  = 30000;
    var height = 30000;
    var widthSegments = 500;
    var heightSegments = 500;

    geometry = new THREE.PlaneGeometry( width,
height, widthSegments, heightSegments );
    var oceanmat = new THREE.MeshLambertMaterial( {
        color: 0x00ccff,
        opacity: 0.5,
        transparent: true,
        side: THREE.DoubleSide
    } );
    var ocean = new THREE.Mesh( geometry, oceanmat );
    scene.add( ocean );
    }

    function animate() {
        requestAnimationFrame(animate);
        var SHIP_data = updateSHIP(dq, dt, t,q);
        dq = SHIP_data.ddq.slice();
        for (var i = 0 ; i < 3 ; dq[i] =
SHIP_data.ddq[i] * 0.99, i++);
        for (var i = 0 ; i < 3 ; q[i] =
SHIP_data.angles[i], i++);
        //q = SHIP_data.angles.slice();
        rotate(SHIP_data, getShipAngles(t));
        controls.update();
        renderer.render(scene, camera);
        t += dt;
        console.log(t);
    }
```

```
    function rotate(SHIP_data, angles) {
        frames[0].rotation.x = SHIP_data.angles[0];
        frames[0].rotation.y = SHIP_data.angles[1];
        frames[0].rotation.z = SHIP_data.angles[2];
        frames[1].rotation.z = angles.phit;
        frames[2].rotation.x = angles.zetat;
    }

    function setCamera() {
        camera = new THREE.PerspectiveCamera(60,
window.innerWidth / window.innerHeight, 100, 50000);
        camera.position.set(10000, 12500, 4000);
        camera.lookAt(0, 0, 0);
        camera.up.set(0, 0, 1);
    }

    function setRenderer() {
        renderer = new THREE.WebGLRenderer({
antialias: true });
        renderer.setSize(window.innerWidth,
window.innerHeight);
        renderer.setClearColor(0xFFFFFF, 1.0);
        var container =
document.getElementById("myCanvas");
        container.appendChild(renderer.domElement);
    }

    function setLights() {
        var light = new
THREE.HemisphereLight(0xffffbb, 0x080820, 1);
        light.castShadow = true;
        scene.add(light);

        var amblight = new
THREE.AmbientLight(0x555555);
        scene.add(amblight);

    }
    function setControls() {
        controls = new THREE.OrbitControls(camera,
renderer.domElement);
        controls.rotateSpeed = 0.5;
        controls.zoomSpeed = 2.0;
        controls.panSpeed = 5.;
        controls.minDistance = 5.0;
        controls.maxDistance = 100000;
    }
```

```
    function onWindowResize() {
        camera.aspect = window.innerWidth /
window.innerHeight;
        camera.updateProjectionMatrix();
        renderer.setSize(window.innerWidth,
window.innerHeight);
    }
    init();
});
```

JavaScript code – Dynamics.js

```
// JavaScript source code


const wi = 22; //m
const h = 10; //m
const d = 85; //m
const A = 81*Math.PI/180; //Amplitude deg*pi/180
const f = 0.05;     //frequency
const w = 2*Math.PI*f; //Angular velocity 2*pi*f

var Boom = 0;
var Boom2 = 0;
var Crane = period(5);
var Crane2 = period(7);
var x = 0;
var y = 0;
const phase = Math.PI/2;

var rotationMatrix = eyeMat(3, 3);
var identityMat = eyeMat(3, 3);
function getShipAngles(t) {

    return {
        phit: guiParams.Crane*t,
        dphit: 0.3,
        ddphit: 0,
        zetat: - guiParams.Boom*Math.cos(t*(2*Math.PI
/ guiParams.Boom2)) + 1.414 / 2,
        dzetat: (2*Math.PI / guiParams.Boom2)*
guiParams.Boom*Math.sin(t*(2*Math.PI /
guiParams.Boom2)),
        ddzetat: 0
    }
}
```

```
function matlabFunction(om_flag, prev_dq, t,q) {

    const m1 = 4100; // Ship-weight
    var J11 = 1/12*m1*(h**2+d**2);
    var J12 = 1/12*m1*(wi**2+d**2);
    var J13 = 1/12*m1*(wi**2+h**2);

    var m3 = 78+3.4+3.8; //tons
    var J31 = 1.1763489*Math.pow(10,5); //tons*m**2
    var J32 = 1.1800369*Math.pow(10,7);
    var J33 = 1.1739636*Math.pow(10,7);



    var w1 = prev_dq[0];
    var w2 = prev_dq[1];
    var w3 = prev_dq[2];
    var angles = getShipAngles(t);

    var phi = angles.phit;
    var dphi = angles.dphit;
    var ddphi = angles.ddphit;
    var zeta = angles.zetat;
    var dzeta = angles.dzetat;
    var ddzeta = angles.ddzetat;

    var M = getMoments(q,m1,prev_dq);
    const M1x = M.M1x;
    const M1y = M.M1y;

    switch (om_flag) {
        case 0:
            return (0.5*(2*J12**2*J31*w2*w3 -
2*J12*J13*J31*w2*w3 + 2*J12*J31*J33*w2*w3 -
2*J13*J31*J33*w2*w3 +
J12*J31**2*w1*w3*Math.sin(2*phi) -
2*J12*J31**2*w2*Math.cos(phi)**2*dphi -
2*J12*J31**2*Math.cos(phi)*dphi*dzeta +
J12*J31**2*w1*dphi*Math.sin(2*phi) -
2*J12*J31**2*w2*w3*Math.cos(phi)**2 -
2*J12*J31**2*w3*Math.cos(phi)*dzeta -
2*J12**2*J31*w2*w3*Math.cos(zeta)**2 +
2*J12**2*J33*w2*w3*Math.cos(zeta)**2 +
J11*J31*J33*w1*w3*Math.sin(2*phi) -
1*J12*J31*J32*w1*w3*Math.sin(2*phi) -
1*J13*J31*J33*w1*w3*Math.sin(2*phi) +
2*J12*J31*J32*w2*Math.cos(phi)**2*dphi -
```

```
2*J12*J31*J33*w2*Math.cos(phi)**2*dphi +
2*J12*J31*J32*Math.cos(phi)*dphi*dzeta +
2*J12*J31*J33*Math.cos(phi)*dphi*dzeta -
1*J12*J31*J32*w1*dphi*Math.sin(2*phi) +
J12*J31*J33*w1*dphi*Math.sin(2*phi) +
2*J12*J31**2*w2*w3*Math.cos(phi)**2*Math.cos(zeta)**2
-
2*J12*J33**2*w2*w3*Math.cos(phi)**2*Math.cos(zeta)**2
+
2*J12*J31**2*w3*Math.cos(phi)*Math.cos(zeta)**2*dzeta
-
2*J12*J33**2*w3*Math.cos(phi)*Math.cos(zeta)**2*dzeta
+ 2*J12*J31*J32*w2*w3*Math.cos(phi)**2 -
2*J12*J31*J33*w2*w3*Math.cos(phi)**2 +
2*J13*J31*J33*w2*w3*Math.cos(phi)**2 +
2*J12*J31*J32*w3*Math.cos(phi)*dzeta +
2*J12*J31*J33*w3*Math.cos(phi)*dzeta +
2*J12*J13*J31*w2*w3*Math.cos(zeta)**2 -
2*J12*J13*J33*w2*w3*Math.cos(zeta)**2 +
2*J12*J31**2*w2*Math.cos(phi)**2*Math.cos(zeta)**2*dp
hi -
2*J12*J33**2*w2*Math.cos(phi)**2*Math.cos(zeta)**2*dp
hi +
2*J12*J31**2*Math.cos(phi)*Math.cos(zeta)**2*dphi*dze
ta -
2*J12*J33**2*Math.cos(phi)*Math.cos(zeta)**2*dphi*dze
ta -
2*J12*J31**2*w1*w2*Math.cos(phi)*Math.cos(zeta)*Math.
sin(zeta) +
2*J12*J33**2*w1*w2*Math.cos(phi)*Math.cos(zeta)*Math.
sin(zeta) -
2*J12*J31*J32*w2*w3*Math.cos(phi)**2*Math.cos(zeta)**
2 +
2*J12*J32*J33*w2*w3*Math.cos(phi)**2*Math.cos(zeta)**
2 -
2*J12*J31*J32*w3*Math.cos(phi)*Math.cos(zeta)**2*dzet
a +
2*J12*J32*J33*w3*Math.cos(phi)*Math.cos(zeta)**2*dzet
a -
2*J12*J31**2*w1*w3*Math.cos(phi)*Math.cos(zeta)**2*Ma
th.sin(phi) +
2*J12*J33**2*w1*w3*Math.cos(phi)*Math.cos(zeta)**2*Ma
th.sin(phi) +
4*J12*J31**2*w1*w2*Math.cos(phi)**3*Math.cos(zeta)*Ma
th.sin(zeta) -
4*J12*J33**2*w1*w2*Math.cos(phi)**3*Math.cos(zeta)*Ma
th.sin(zeta) -
2*J12*J31**2*w1**2*Math.cos(phi)**2*Math.cos(zeta)*Ma
```

```
th.sin(phi)*Math.sin(zeta) +
2*J12*J31**2*w2**2*Math.cos(phi)**2*Math.cos(zeta)*Ma
th.sin(phi)*Math.sin(zeta) +
2*J12*J33**2*w1**2*Math.cos(phi)**2*Math.cos(zeta)*Ma
th.sin(phi)*Math.sin(zeta) -
2*J12*J33**2*w2**2*Math.cos(phi)**2*Math.cos(zeta)*Ma
th.sin(phi)*Math.sin(zeta) -
2*J12*J31*J32*w2*Math.cos(phi)**2*Math.cos(zeta)**2*d
phi +
2*J12*J32*J33*w2*Math.cos(phi)**2*Math.cos(zeta)**2*d
phi -
2*J12*J31*J32*Math.cos(phi)*Math.cos(zeta)**2*dphi*dz
eta +
2*J12*J32*J33*Math.cos(phi)*Math.cos(zeta)**2*dphi*dz
eta -
2*J12*J31**2*w1*Math.cos(phi)*Math.cos(zeta)**2*Math.
sin(phi)*dphi +
2*J12*J33**2*w1*Math.cos(phi)*Math.cos(zeta)**2*Math.
sin(phi)*dphi +
2*J12*J31**2*w1*Math.cos(phi)**2*Math.cos(zeta)*Math.
sin(zeta)*dzeta -
2*J12*J33**2*w1*Math.cos(phi)**2*Math.cos(zeta)*Math.
sin(zeta)*dzeta +
2*J12*J31*J32*w1*w2*Math.cos(phi)*Math.cos(zeta)*Math
.sin(zeta) -
2*J12*J32*J33*w1*w2*Math.cos(phi)*Math.cos(zeta)*Math
.sin(zeta) +
2*J12*J31*J32*w1*w3*Math.cos(phi)*Math.cos(zeta)**2*M
ath.sin(phi) -
2*J12*J32*J33*w1*w3*Math.cos(phi)*Math.cos(zeta)**2*M
ath.sin(phi) -
4*J12*J31*J32*w1*w2*Math.cos(phi)**3*Math.cos(zeta)*M
ath.sin(zeta) +
4*J12*J32*J33*w1*w2*Math.cos(phi)**3*Math.cos(zeta)*M
ath.sin(zeta) +
2*J12*J31*J32*w1**2*Math.cos(phi)**2*Math.cos(zeta)*M
ath.sin(phi)*Math.sin(zeta) -
2*J12*J31*J32*w2**2*Math.cos(phi)**2*Math.cos(zeta)*M
ath.sin(phi)*Math.sin(zeta) -
2*J12*J32*J33*w1**2*Math.cos(phi)**2*Math.cos(zeta)*M
ath.sin(phi)*Math.sin(zeta) +
2*J12*J32*J33*w2**2*Math.cos(phi)**2*Math.cos(zeta)*M
ath.sin(phi)*Math.sin(zeta) +
2*J12*J31*J32*w1*Math.cos(phi)*Math.cos(zeta)**2*Math
.sin(phi)*dphi -
2*J12*J32*J33*w1*Math.cos(phi)*Math.cos(zeta)**2*Math
.sin(phi)*dphi -
2*J12*J31*J32*w1*Math.cos(phi)**2*Math.cos(zeta)*Math
```

```
.sin(zeta)*dzeta +
2*J12*J32*J33*w1*Math.cos(phi)**2*Math.cos(zeta)*Math
.sin(zeta)*dzeta +
2*J12*J31**2*w2*Math.cos(phi)*Math.cos(zeta)*Math.sin
(phi)*Math.sin(zeta)*dzeta -
2*J12*J33**2*w2*Math.cos(phi)*Math.cos(zeta)*Math.sin
(phi)*Math.sin(zeta)*dzeta -
2*J12*J31*J32*w2*Math.cos(phi)*Math.cos(zeta)*Math.si
n(phi)*Math.sin(zeta)*dzeta +
2*J12*J32*J33*w2*Math.cos(phi)*Math.cos(zeta)*Math.si
n(phi)*Math.sin(zeta)*dzeta))/(J11*J12*J31 +
J11*J31*J33 - 1*J11*J31*J33*Math.cos(phi)**2 +
J12*J31*J33*Math.cos(phi)**2 -
1*J11*J12*J31*Math.cos(zeta)**2 +
J11*J12*J33*Math.cos(zeta)**2);
        case 1:
            return  (0.5*(2*J11*J31**2*w1*w3 -
2*J11**2*J31*w1*w3 + 2*J11*J31**2*w1*dphi +
2*J11*J13*J31*w1*w3 - 2*J11*J31*J32*w1*w3 -
J11*J31**2*w2*w3*Math.sin(2*phi) -
2*J11*J31**2*w1*Math.cos(phi)**2*dphi -
2*J11*J31**2*w1*Math.cos(zeta)**2*dphi +
2*J11*J33**2*w1*Math.cos(zeta)**2*dphi -
2*J11*J31**2*Math.sin(phi)*dphi*dzeta -
2*J11*J31*J32*w1*dphi + 2*J11*J31*J33*w1*dphi -
1*J11*J31**2*w2*dphi*Math.sin(2*phi) +
J11*J31**2*w2*Math.sin(2*zeta)*dzeta -
1*J11*J33**2*w2*Math.sin(2*zeta)*dzeta -
2*J11*J31**2*w1*w3*Math.cos(phi)**2 -
2*J11*J31**2*w1*w3*Math.cos(zeta)**2 +
2*J11**2*J31*w1*w3*Math.cos(zeta)**2 +
2*J11*J33**2*w1*w3*Math.cos(zeta)**2 -
2*J11**2*J33*w1*w3*Math.cos(zeta)**2 -
2*J11*J31**2*w3*Math.sin(phi)*dzeta +
2*J11*J31**2*Math.cos(zeta)**2*Math.sin(phi)*dphi*dze
ta -
2*J11*J33**2*Math.cos(zeta)**2*Math.sin(phi)*dphi*dze
ta + J11*J31*J32*w2*w3*Math.sin(2*phi) -
1*J12*J31*J33*w2*w3*Math.sin(2*phi) +
J13*J31*J33*w2*w3*Math.sin(2*phi) -
2*J11*J31**2*w1**2*Math.cos(phi)*Math.cos(zeta)*Math.
sin(zeta) +
2*J11*J31**2*w2**2*Math.cos(phi)*Math.cos(zeta)*Math.
sin(zeta) +
2*J11*J33**2*w1**2*Math.cos(phi)*Math.cos(zeta)*Math.
sin(zeta) -
2*J11*J33**2*w2**2*Math.cos(phi)*Math.cos(zeta)*Math.
sin(zeta) + 2*J11*J31*J32*w1*Math.cos(phi)**2*dphi -
```

```
2*J11*J31*J33*w1*Math.cos(phi)**2*dphi +
2*J11*J31*J32*w1*Math.cos(zeta)**2*dphi -
2*J11*J32*J33*w1*Math.cos(zeta)**2*dphi +
2*J11*J31*J32*Math.sin(phi)*dphi*dzeta +
2*J11*J31*J33*Math.sin(phi)*dphi*dzeta +
J11*J31*J32*w2*dphi*Math.sin(2*phi) -
1*J11*J31*J33*w2*dphi*Math.sin(2*phi) +
2*J11*J31**2*w1**2*Math.cos(phi)**3*Math.cos(zeta)*Ma
th.sin(zeta) -
2*J11*J31**2*w2**2*Math.cos(phi)**3*Math.cos(zeta)*Ma
th.sin(zeta) -
2*J11*J33**2*w1**2*Math.cos(phi)**3*Math.cos(zeta)*Ma
th.sin(zeta) +
2*J11*J33**2*w2**2*Math.cos(phi)**3*Math.cos(zeta)*Ma
th.sin(zeta) -
1*J11*J31*J32*w2*Math.sin(2*zeta)*dzeta +
J11*J32*J33*w2*Math.sin(2*zeta)*dzeta +
2*J11*J31**2*w1*w3*Math.cos(phi)**2*Math.cos(zeta)**2
-
2*J11*J33**2*w1*w3*Math.cos(phi)**2*Math.cos(zeta)**2
+
2*J11*J31**2*w3*Math.cos(zeta)**2*Math.sin(phi)*dzeta
-
2*J11*J33**2*w3*Math.cos(zeta)**2*Math.sin(phi)*dzeta
+ 2*J11*J31*J32*w1*w3*Math.cos(phi)**2 -
2*J11*J31*J33*w1*w3*Math.cos(phi)**2 +
2*J13*J31*J33*w1*w3*Math.cos(phi)**2 -
2*J11*J13*J31*w1*w3*Math.cos(zeta)**2 +
2*J11*J13*J33*w1*w3*Math.cos(zeta)**2 +
2*J11*J31*J32*w1*w3*Math.cos(zeta)**2 -
2*J11*J32*J33*w1*w3*Math.cos(zeta)**2 +
2*J11*J31*J32*w3*Math.sin(phi)*dzeta +
2*J11*J31*J33*w3*Math.sin(phi)*dzeta +
2*J11*J31**2*w1*Math.cos(phi)**2*Math.cos(zeta)**2*dp
hi -
2*J11*J33**2*w1*Math.cos(phi)**2*Math.cos(zeta)**2*dp
hi +
2*J11*J31*J32*w1**2*Math.cos(phi)*Math.cos(zeta)*Math
.sin(zeta) -
2*J11*J31*J32*w2**2*Math.cos(phi)*Math.cos(zeta)*Math
.sin(zeta) -
2*J11*J32*J33*w1**2*Math.cos(phi)*Math.cos(zeta)*Math
.sin(zeta) +
2*J11*J32*J33*w2**2*Math.cos(phi)*Math.cos(zeta)*Math
.sin(zeta) -
2*J11*J31**2*w1*w2*Math.cos(zeta)*Math.sin(phi)*Math.
sin(zeta) +
2*J11*J33**2*w1*w2*Math.cos(zeta)*Math.sin(phi)*Math.
```

```
sin(zeta) -
2*J11*J31*J32*w1**2*Math.cos(phi)**3*Math.cos(zeta)*M
ath.sin(zeta) +
2*J11*J31*J32*w2**2*Math.cos(phi)**3*Math.cos(zeta)*M
ath.sin(zeta) +
2*J11*J32*J33*w1**2*Math.cos(phi)**3*Math.cos(zeta)*M
ath.sin(zeta) -
2*J11*J32*J33*w2**2*Math.cos(phi)**3*Math.cos(zeta)*M
ath.sin(zeta) -
2*J11*J31*J32*w1*w3*Math.cos(phi)**2*Math.cos(zeta)**
2 +
2*J11*J32*J33*w1*w3*Math.cos(phi)**2*Math.cos(zeta)**
2 +
2*J11*J31**2*w2*w3*Math.cos(phi)*Math.cos(zeta)**2*Ma
th.sin(phi) -
2*J11*J33**2*w2*w3*Math.cos(phi)*Math.cos(zeta)**2*Ma
th.sin(phi) -
2*J11*J31*J32*w3*Math.cos(zeta)**2*Math.sin(phi)*dzet
a +
2*J11*J32*J33*w3*Math.cos(zeta)**2*Math.sin(phi)*dzet
a -
2*J11*J31*J32*w1*Math.cos(phi)**2*Math.cos(zeta)**2*d
phi +
2*J11*J32*J33*w1*Math.cos(phi)**2*Math.cos(zeta)**2*d
phi +
2*J11*J31**2*w2*Math.cos(phi)*Math.cos(zeta)**2*Math.
sin(phi)*dphi -
2*J11*J33**2*w2*Math.cos(phi)*Math.cos(zeta)**2*Math.
sin(phi)*dphi -
2*J11*J31*J32*Math.cos(zeta)**2*Math.sin(phi)*dphi*dz
eta +
2*J11*J32*J33*Math.cos(zeta)**2*Math.sin(phi)*dphi*dz
eta -
2*J11*J31**2*w2*Math.cos(phi)**2*Math.cos(zeta)*Math.
sin(zeta)*dzeta +
2*J11*J33**2*w2*Math.cos(phi)**2*Math.cos(zeta)*Math.
sin(zeta)*dzeta +
2*J11*J31*J32*w1*w2*Math.cos(zeta)*Math.sin(phi)*Math
.sin(zeta) -
2*J11*J32*J33*w1*w2*Math.cos(zeta)*Math.sin(phi)*Math
.sin(zeta) -
2*J11*J31*J32*w2*w3*Math.cos(phi)*Math.cos(zeta)**2*M
ath.sin(phi) +
2*J11*J32*J33*w2*w3*Math.cos(phi)*Math.cos(zeta)**2*M
ath.sin(phi) -
2*J11*J31*J32*w2*Math.cos(phi)*Math.cos(zeta)**2*Math
.sin(phi)*dphi +
2*J11*J32*J33*w2*Math.cos(phi)*Math.cos(zeta)**2*Math
```

```
.sin(phi)*dphi +
2*J11*J31*J32*w2*Math.cos(phi)**2*Math.cos(zeta)*Math
.sin(zeta)*dzeta -
2*J11*J32*J33*w2*Math.cos(phi)**2*Math.cos(zeta)*Math
.sin(zeta)*dzeta +
4*J11*J31**2*w1*w2*Math.cos(phi)**2*Math.cos(zeta)*Ma
th.sin(phi)*Math.sin(zeta) -
4*J11*J33**2*w1*w2*Math.cos(phi)**2*Math.cos(zeta)*Ma
th.sin(phi)*Math.sin(zeta) +
2*J11*J31**2*w1*Math.cos(phi)*Math.cos(zeta)*Math.sin
(phi)*Math.sin(zeta)*dzeta -
2*J11*J33**2*w1*Math.cos(phi)*Math.cos(zeta)*Math.sin
(phi)*Math.sin(zeta)*dzeta -
4*J11*J31*J32*w1*w2*Math.cos(phi)**2*Math.cos(zeta)*M
ath.sin(phi)*Math.sin(zeta) +
4*J11*J32*J33*w1*w2*Math.cos(phi)**2*Math.cos(zeta)*M
ath.sin(phi)*Math.sin(zeta) -
2*J11*J31*J32*w1*Math.cos(phi)*Math.cos(zeta)*Math.si
n(phi)*Math.sin(zeta)*dzeta +
2*J11*J32*J33*w1*Math.cos(phi)*Math.cos(zeta)*Math.si
n(phi)*Math.sin(zeta)*dzeta))/(J11*J12*J31 +
J11*J31*J33 - 1*J11*J31*J33*Math.cos(phi)**2 +
J12*J31*J33*Math.cos(phi)**2 -
1*J11*J12*J31*Math.cos(zeta)**2 +
J11*J12*J33*Math.cos(zeta)**2);
        case 2:
            return (w1*w2*(J11 - J12))/J13;
        }
    console.log("ERROR in Matlab-Function, swtich");
    return 0;
}

function getMoments(q,m1,dq){

    const length = 85;
    const gravity = 9.81;
    const mass = m1;
    return{
        M1x :   - m1 * length * gravity *
Math.sin(q[0]),
        M1y :    m1 * length * gravity *
Math.sin(q[1])
    }
}
function updateSHIP(dq, dt, t,q) {
    /*if(period(guiParams.Crane)!=Crane){
        x = Math.asin(getShipAngles(t).phit/Boom)-
t*period(guiParams.Crane)-phase;
```

```
        while(x > 2 * Math.PI){
            x -= (2 * Math.PI);
        }
        while(x < 0){
            x += 2 * Math.PI;
        }
        Crane = period(guiParams.Crane);
    }
    if(period(guiParams.Crane2)!=Crane2){
        y = Math.asin(getShipAngles(t).zetat/Boom2)-
t*period(guiParams.Crane2)-phase;
        while(y > 2 * Math.PI){
            y -= (2 * Math.PI);
        }
        while(y < 0){
            y += 2 * Math.PI;
        }
        Crane2 = period(guiParams.Crane2);
    }*/
    //Boom = Boom > guiParams.Boom + 0.05 ? Boom -
0.005 : Boom < guiParams.Boom - 0.05 ? Boom + 0.005 :
Boom;
   // Boom2 = Boom2 > guiParams.Boom2 + 0.05 ? Boom2
- 0.005 : Boom2 < guiParams.Boom2 - 0.05 ? Boom2 +
0.005 : Boom2;
    var ddq = RK4(dq, dt, t,q);
    rotationMatrix = getRotMat(rotationMatrix, ddq,
dt);
    var angles = rotMatToAngle(rotationMatrix);
    return {
        angles: angles,
        ddq: ddq
    }

}


function RK4(last_dq, dt, t,q) {
    var k1 = [], k2 = [], k3 = [], k4 = [];
    var arr = [];
    var dq = last_dq.slice();
    for (var a = 0 ; a < dq.length ; a++) {
        k1[a] = dt * matlabFunction(a, dq, t,q);
    }
    for (var a = 0 ; a < dq.length ; a++) {
        for (var i = 0 ; i < k1.length ; arr[i] =
dq[i] + k1[i] / 2, i++);
        k2[a] = dt * matlabFunction(a, arr, t + dt /
```

```
2,q)
    }
    for (var a = 0 ; a < dq.length ; a++) {
        for (var i = 0 ; i < k2.length ; arr[i] =
dq[i] + k2[i] / 2, i++);
        k3[a] = dt * matlabFunction(a, arr, t + dt /
2,q);
    }
    for (var a = 0 ; a < dq.length ; a++) {
        for (var i = 0 ; i < k3.length ; arr[i] =
dq[i] + k3[i], i++);
        k4[a] = dt * matlabFunction(a, arr, t +
dt,q);
    }
    for (var i = 0 ; i < dq.length ; dq[i] = dq[i] +
(k1[i] + 2 * k2[i] + 2 * k3[i] + k4[i]) / 6, i++);

    return dq;
}

function rotMatToAngle(rotMat) {
    var r00 = rotMat[0][0];
    var r01 = rotMat[0][1];
    var r10 = rotMat[1][0];
    var r11 = rotMat[1][1];
    var r02 = rotMat[0][2];
    var r20 = rotMat[2][0];
    var r12 = rotMat[1][2];
    var r21 = rotMat[2][1];
    var r22 = rotMat[2][2];

    var rot1 = Math.atan2(r12, r22);
    var s1 = Math.sin(rot1);
    var c1 = Math.cos(rot1);
    var c2 = Math.sqrt(r00 * r00 + r01 * r01);
    var rot2 = Math.atan2(r02, c2);
    var rot3 = Math.atan2(s1 * r20 - c1 * r10, c1 *
r11 - s1 * r21);
    return [rot1, rot2, rot3];
}

function eyeMat(rows, cols) {
    var Mat = zerosMat([rows, cols]);
    for (var i = 0; i < rows; i++) {
        Mat[i][i] = 1;
    }
    return Mat;
}
```

```
function getRotMat(rotMat, w, dt) {
    var w1 = w[0];
    var w2 = w[1];
    var w3 = w[2];
    var wNorm = Math.sqrt(w1 * w1 + w2 * w2 + w3 *
w3);
    if (w1 * w2 * w3 < 0) wNorm = -wNorm;

    var wSkew = [[0, -w3, w2], [w3, 0, -w1], [-w2,
w1, 0]];

    var c2 = Math.sin(dt * wNorm) / wNorm;
    var c3 = (1 - Math.cos(dt * wNorm)) / (wNorm *
wNorm);

    var term1 = eyeMat(3, 3);
    var term2 = constMatMult(c2, wSkew);
    var term3 = constMatMult(c3, MatMatMult(wSkew,
wSkew));

    var expwt = MatMatAdd(term1, MatMatAdd(term2,
term3));
    rotMat = MatMatMult(rotMat, expwt);

    return rotMat;
}

function constMatMult(c, Mat) {
    var rows = Mat.length;
    var cols = Mat[0].length;
    for (var i = 0; i < rows; i++) {
        for (var j = 0; j < cols; j++) {
            Mat[i][j] = c * Mat[i][j];
        }
    }
    return Mat;
}

function MatMatAdd(Mat1, Mat2) {
    var rows = Mat1.length;
    var cols = Mat1[0].length;
    for (var i = 0; i < rows; i++) {
        for (var j = 0; j < cols; j++) {
            Mat1[i][j] += Mat2[i][j];
        }
    }
    return Mat1;
```

```
}

function MatMatMult(Mat1, Mat2) {
    var rows1 = Mat1.length;
    var cols1 = Mat1[0].length;
    var rows2 = Mat2.length;
    var cols2 = Mat2[0].length;
    var Mat = zerosMat([rows1, cols2]);

    for (var i = 0; i < rows1; i++) {
        for (var j = 0; j < cols2; j++) {
            for (var k = 0; k < cols1; k++) {
                Mat[i][j] += Mat1[i][k] * Mat2[k][j];
            }
        }
    }
    return Mat;
}

function zerosMat(dimensions) {
    var mat = [];
    for (var i = 0; i < dimensions[0]; ++i) {
        mat.push(dimensions.length == 1 ? 0 :
zerosMat(dimensions.slice(1)));
    }
    return mat;
}

function period(seconds) {
    return (Math.PI * 2) / seconds;
}
```

APPENDIX F

Details on the moving frame method for a crane on a ship as worked out by our supervisor Prof. Thomas J. Impelluso.

We will place a frame at the center of mass of a boat.
We drop an inertial frame from it.
The 1-axis of the inertial points toward the bow
The 2-axis of the inertial points toward the port
The 3-axis of the inertial points up
Initially, the boat (1) frame is aligned with the inertial.
We construct the frame relationship FROM the inertial frame.
Basically, this is how to "get to" and "orient" the boat (1) frame from the inertial.

$$\left(E^{(1)}(t)\right) = \begin{bmatrix} R^{(1)}(t) & x_c^{(1)}(t) \\ 0 & 1 \end{bmatrix}$$

$$\left(\dot{E}^{(1)}(t)\right) = \begin{bmatrix} \dot{R}^{(1)}(t) & \dot{x}_c^{(1)}(t) \\ 0 & 1 \end{bmatrix}$$

$$\left(E^{(1)}(t)\right)^{-1} = \begin{bmatrix} R^{(1)}(t)^T & -\left(R^{(1)}(t)^T\right)x_c^{(1)}(t) \\ 0 & 1 \end{bmatrix}$$

$$\left(E^{(1)}(t)\right)^{-1} = \begin{bmatrix} R^{(1)}(t)^T & -\left(R^{(1)}(t)^T\right)x_c^{(1)}(t) \\ 0 & 1 \end{bmatrix}$$

$$\Omega^{(1)}(t) = \left(E^{(1)}(t)\right)^{-1}\dot{E}^{(1)}(t)$$

$$\Omega^{(1)}(t) = \begin{bmatrix} \widetilde{\omega}^{(1)} & R^{(1)}(t)^T\dot{x}_c^{(1)}(t) \\ 0 & 0 \end{bmatrix}$$

$$\left(\dot{\mathbf{e}}^{(1)} \quad \dot{\mathbf{r}}_c^{(1)}\right) = \left(\mathbf{e}^{(1)} \quad \mathbf{r}_c^{(1)}\right)\boldsymbol{\Omega}^{(1)}(t)$$

Split it:

$$\dot{\mathbf{e}}^{(1)} = \mathbf{e}^{(1)}\widetilde{\omega}^{(1)}(t)$$

$$\dot{\mathbf{r}}_c^{(1)} = \mathbf{e}^{(1)}\left(R^{(1)}(t)\right)^T\dot{x}_c^{(1)}(t)$$

Generalized velocities:

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix}$$

$\{\dot{X}(t)\}$ has 18 rows

The subscript (1) will be the boat frame
The subscript (2) will be the tower of the crane.
The subscript (3) will be the extended boom of the crane.

Coordinates from the inertial:

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix}$$

Constructing the components from the inertial frame:

$$\dot{\mathbf{r}}_c^{(1)} = \mathbf{e}^{(1)}\left(R^{(1)}(t)\right)^T \dot{x}_c^{(1)}(t)$$

$$\dot{\mathbf{r}}_c^{(1)} = \mathbf{e}^I \dot{x}_c^{(1)}(t)$$

These are parameters we want to observe:

$$\widetilde{\omega}^{(1)} \qquad\qquad \dot{x}_c^{(1)}(t)$$

We will want to find these two in order to get the position and orientation of a boat.

Before continuing, state the generalized velocities and the ESSENTIAL generalized velocities:

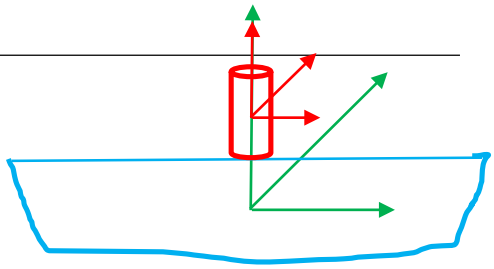
$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix} \qquad\qquad \{\dot{X}(t)\} = \{\dot{q}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(3)} \end{Bmatrix}$$

Above, on the left, are the GENERALIZED velocities. On the right are the ESSENTIAL GENERALIZE velocities.

In setting it up this way on the LEFT, we have decided on the format of how we will create B.
Our goal is now to construct B by marching through each link and obtain an expression for each term on the LEFT.
Then we will decide that the terms on the right are the unknowns when we apply the moments and forces.
We know that if we knew the terms on the right, we have everything: for those are the essential generalized coordinates. We will see this to be true.

$$\{\dot{X}(t)\} = [B(t)]\{\dot{q}(t)\}$$

Now we must get the omega and position for the rotation of the base of the crane.
Move to the crane that turns.



Crane Kinematics

$$\mathbf{e}^{(2)}(t) = \mathbf{e}^{(1)}R^{(2/1)}(t)$$

The crane will rotate about the vertical axis as can be seen by the following:

$$R^{(2/1)}(t) = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And now we note that due to the driving torque, the angular velocity of the turret is: $\dot{\phi}$
Here, naturally, it could be $\dot{\phi}(t)$.
The Center of Mass of the crane will be displaced upward from the deck of the ship with a distance *h*.

$$s_c^{(2/1)} = \begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix}$$

Return to a general form, in reality, it could be at the stern once we realize that $s_c^{(2/1)}$ could contain all three positions.

$$\mathbf{r}_c^{(2)}(t) = \mathbf{e}^{(1)}s_c^{(2/1)} + \mathbf{r}_c^{(1)}(t)$$

Thus

$$\left(\mathbf{e}^{(2)}(t) \quad \mathbf{r}_c^{(2)}(t)\right) = \left(\mathbf{e}^{(1)}(t) \quad \mathbf{r}_c^{(1)}(t)\right)\begin{bmatrix} R^{(2/1)}(t) & s_c^{(2/1)} \\ 0 & 1 \end{bmatrix} = \left(\mathbf{e}^{(1)}(t) \quad \mathbf{r}_c^{(1)}(t)\right)E^{(2/1)}(t)$$

Faster way to build the matrix-system:

$$\left(\mathbf{e}^{(2)}(t) \quad \mathbf{r}_c^{(2)}(t)\right) = (\mathbf{e}^{I} \quad 0)E^{(2)}(t)$$

$$\left(\mathbf{e}^{(2)}(t) \quad \mathbf{r}_c^{(2)}(t)\right) = (\mathbf{e}^{I} \quad 0)E^{(1)}(t)E^{(2/1)}(t)$$

$$E^{(2)}(t) = E^{(1)}(t)E^{(2/1)}(t)$$

$$E^{(2)}(t) = \begin{bmatrix} R^{(1)}(t) & x_c^{(1)}(t) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R^{(2/1)}(t) & s_c^{(2/1)}(t) \\ 0 & 1 \end{bmatrix}$$

$$E^{(2)}(t) = \begin{bmatrix} R^{(1)}(t)R^{(2/1)}(t) & R^{(1)}(t)s_c^{(2/1)}(t) + x_c^{(1)}(t) \\ 0 & 1 \end{bmatrix}$$

Now we define the OMEGA, and look at the RATE and INVERSE

$$\dot{E}^{(2)}(t) = \begin{bmatrix} \dot{R}^{(1)}(t)R^{(2/1)}(t) + R^{(1)}(t)\dot{R}^{(2/1)}(t) & \dot{R}^{(1)}(t)s_c^{(2/1)}(t) + \dot{x}_c^{(1)}(t) \\ 0 & 0 \end{bmatrix}$$

$$\left(E^{(2)}(t)\right)^{-1} = \begin{bmatrix} \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T & -\left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(R^{(1)}(t) s_c^{(2/1)}(t) + x_c^{(1)}(t)\right) \\ 0 & 1 \end{bmatrix}$$

$$\Omega^{(2)}(t) = \begin{bmatrix} \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T & -\left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(R^{(1)}(t) s_c^{(2/1)}(t) \right. \\ 0 & 1 \end{bmatrix} \left. + x_c^{(1)}(t)\right) \begin{bmatrix} \dot{R}^{(1)}(t) R^{(2/1)}(t) + R^{(1)}(t) \dot{R}^{(2/1)}(t) & \dot{R}^{(1)}(t) s_c^{(2/1)}(t) + \dot{x}_c^{(1)}(t) \\ 0 & 0 \end{bmatrix}$$

$$\Omega^{(2)}(t) = \begin{bmatrix} \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(\dot{R}^{(1)}(t) R^{(2/1)}(t) + R^{(1)}(t) \dot{R}^{(2/1)}(t)\right) & \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(\dot{R}^{(1)}(t) s_c^{(2/1)}(t) + \dot{x}_c^{(1)}(t)\right) \\ 0 & 0 \end{bmatrix}$$

Now collapse it. In the next one, there is a faster way.

$$\Omega^{(2)}(t) = \begin{bmatrix} \left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t) R^{(2/1)}(t) + \left(R^{(2/1)}(t)\right)^T \dot{R}^{(2/1)}(t)\right) & \left(R^{(2/1)}(t)\right)^T \left(\tilde{\omega}^{(1)}(t) s_c^{(2/1)} + \left(R^{(1)}(t)\right)^T \dot{x}_c^{(1)}(t)\right) \\ 0 & 0 \end{bmatrix}$$

$$\Omega^{(2)}(t) = \begin{bmatrix} \left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t) R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right) & \left(R^{(2/1)}(t)\right)^T \left(\tilde{\omega}^{(1)}(t) s_c^{(2/1)} + \left(R^{(1)}(t)\right)^T \dot{x}_c^{(1)}(t)\right) \\ 0 & 0 \end{bmatrix}$$

Thus we have

$$\left(\dot{\mathbf{e}}^{(2)}(t) \quad \dot{\mathbf{r}}_c^{(2)}(t)\right) = \left(\mathbf{e}^{(2)}(t) \quad \mathbf{r}_c^{(2)}(t)\right) \boldsymbol{\Omega}^{(2)}(t)$$

$$\left(\dot{\mathbf{e}}^{(2)}(t) \quad \dot{\mathbf{r}}_c^{(2)}(t)\right) = \left(\mathbf{e}^{(2)}(t) \quad \mathbf{r}_c^{(2)}(t)\right) \begin{bmatrix} \left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t) R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right) & \left(R^{(2/1)}(t)\right)^T \left(\tilde{\omega}^{(1)}(t) s_c^{(2/1)} + \left(R^{(1)}(t)\right)^T \dot{x}_c^{(1)}(t)\right) \\ 0 & 0 \end{bmatrix}$$

Now let me pull out the two I need.

$$\dot{\mathbf{e}}^{(2)}(t) = \mathbf{e}^{(2)}(t) \left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t) R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right)$$

$$\dot{\mathbf{r}}_c^{(2)}(t) = \mathbf{e}^{(2)}(t) \left(\left(R^{(2/1)}(t)\right)^T \left(\tilde{\omega}^{(1)}(t) s_c^{(2/1)} + \left(R^{(1)}(t)\right)^T \dot{x}_c^{(1)}(t)\right)\right)$$

We want the displacement in the INERTIAL frame.
This form above, gave the displacement to me in the MOVING frame (1)

$$\dot{\mathbf{e}}^{(2)}(t) = \mathbf{e}^{(2)}(t) \left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t) R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right)$$

$$\dot{\mathbf{r}}_c^{(2)}(t) = \mathbf{e}^{(1)}(t) R^{(2/1)}(t) \left(\left(R^{(2/1)}(t)\right)^T \left(\tilde{\omega}^{(1)}(t) s_c^{(2/1)} + \left(R^{(1)}(t)\right)^T \dot{x}_c^{(1)}(t)\right)\right)$$

Move the ***r-dot term*** to the inertial

$$\dot{\mathbf{e}}^{(2)}(t) = \mathbf{e}^{(2)}(t)\left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t)R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right)$$

$$\dot{\mathbf{r}}_c^{(2)}(t) = \mathbf{e}^{(I)}(t)R^{(1)}R^{(2/1)}(t)\left(\left(R^{(2/1)}(t)\right)^T\left(\tilde{\omega}^{(1)}(t)s_c^{(2/1)} + \left(R^{(1)}(t)\right)^T\dot{x}_c^{(1)}(t)\right)\right)$$

Simplify by multiplying out

$$\dot{\mathbf{e}}^{(2)}(t) = \mathbf{e}^{(2)}(t)\left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t)R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right)$$

$$\dot{\mathbf{r}}_c^{(2)}(t) = \mathbf{e}^{I}(t)\left(R^{(1)}(t)\tilde{\omega}^{(1)}(t)s_c^{(2/1)} + \dot{x}_c^{(1)}(t)\right)$$

Pull out what I want by dropping the frame

$$\tilde{\omega}^{(2)}(t) = \left(\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t)R^{(2/1)}(t) + \tilde{\omega}^{(2/1)}(t)\right)$$

$$\dot{x}_c^{(2)}(t) = \left(R^{(1)}(t)\tilde{\omega}^{(1)}(t)s_c^{(2/1)} + \dot{x}_c^{(1)}(t)\right)$$

TRICKY STUFF PENDING: WATCH CAREFULLY THE NEXT FEW STEPS
I will want all omegas at the END of each term (you will see why, soon enough)
For the green one, this means we use that rule of skewing and do the unskew and pull out from the frame and convert to a column.

$$\left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t)R^{(2/1)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t)$$

For the yellow, one, we switch the last two in the first term of the sum (reversing the cross product and introducing a negative), and then transpose to remove the negative

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t) + \omega^{(2/1)}(t)$$

$$\dot{x}_c^{(2)}(t) = R^{(1)}(t)\left(\tilde{s}_c^{(2/1)}\right)\omega^{(1)}(t) + \dot{x}_c^{(1)}(t)$$

MORE TRICKY STUFF PENDING: I will use a special format for that omega 2/1
I do this because I know there is only one variable.
And I must see that e3 is a column with a 1 in the third spot.

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t) + \dot{\phi}^{(2)}e_3$$

$$\dot{x}_c^{(2)}(t) = R^{(1)}(t)\left(\tilde{s}_c^{(2/1)}\right)\omega^{(1)}(t) + \dot{x}_c^{(1)}(t)$$

I have moved a lot of terms to the back. Let me summarize:

$$\omega^{(1)}(t) = \omega^{(1)}(t)$$

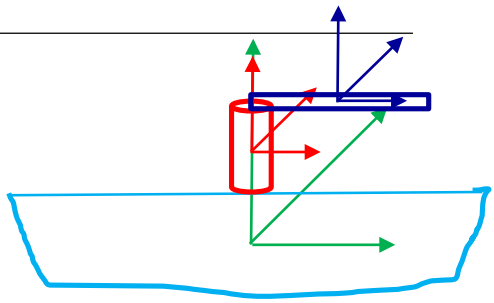
$$\dot{x}_c^{(1)}(t) = \dot{x}_c^{(1)}(t)$$

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t) + \dot{\phi}^{(2)} e_3$$

$$\dot{x}_c^{(2)}(t) = R^{(1)}(t)\left(\tilde{s}_c^{(2/1)}\right)\omega^{(1)}(t) + \dot{x}_c^{(1)}(t)$$

$$\{\dot{X}(t)\} = [B(t)]\{\dot{q}(t)\}$$

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix} \qquad \{\dot{X}(t)\} = \{\dot{q}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(3)} \end{Bmatrix}$$



Last boom

First, we continue up the rest of the way to the top
Then we rotate about the (2) axis of the red frame
Then we move out "d" along the boom

$$\left(\mathbf{e}^{(3)}(t)\ \ \mathbf{r}_c^{(3)}(t)\right)=\left(\mathbf{e}^{(2)}(t)\ \ \mathbf{r}_c^{(2)}(t)\right)\begin{bmatrix} I & \begin{pmatrix}0\\0\\h\end{pmatrix}\\ 0 & 1\end{bmatrix}\begin{bmatrix} R^{(3/2)} & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix} I & \begin{pmatrix}d\\0\\0\end{pmatrix}\\ 0 & 1\end{bmatrix}$$

$$\left(\mathbf{e}^{(3)}(t)\ \ \mathbf{r}_c^{(3)}(t)\right)=\left(\mathbf{e}^{(2)}(t)\ \ \mathbf{r}_c^{(2)}(t)\right)\begin{bmatrix} I & \begin{pmatrix}0\\0\\h\end{pmatrix}\\ 0 & 1\end{bmatrix}\begin{bmatrix} R^{(3/2)} & R^{(3/2)}\begin{pmatrix}d\\0\\0\end{pmatrix}\\ 0 & 1\end{bmatrix}$$

$$\left(\mathbf{e}^{(3)}(t)\ \ \mathbf{r}_c^{(3)}(t)\right)=\left(\mathbf{e}^{(2)}(t)\ \ \mathbf{r}_c^{(2)}(t)\right)\begin{bmatrix} R^{(3/2)} & R^{(3/2)}\begin{pmatrix}d\\0\\0\end{pmatrix}+\begin{pmatrix}0\\0\\h\end{pmatrix}\\ 0 & 1\end{bmatrix}$$

$$s_C^{(2/1)}=\begin{pmatrix}0\\0\\h\end{pmatrix}\qquad s_C^{(3/2)}=\begin{pmatrix}d\\0\\0\end{pmatrix}$$

$$\left(\mathbf{e}^{(3)}(t)\ \ \mathbf{r}_c^{(3)}(t)\right)=\left(\mathbf{e}^{(2)}(t)\ \ \mathbf{r}_c^{(2)}(t)\right)\begin{bmatrix} R^{(3/2)} & R^{(3/2)}\tilde{s}_c^{(3/2)}+s_C^{(2/1)}\\ 0 & 1\end{bmatrix}$$

$$\left(\mathbf{e}^{(2)}(t)\ \ \mathbf{r}_c^{(2)}(t)\right)=\left(\mathbf{e}^{I}(t)\ \ 0\right)E^{(2)}(t)$$

$$E^{(2)}(t)=\begin{bmatrix} R^{(1)}(t)R^{(2/1)}(t) & R^{(1)}(t)s_C^{(2/1)}+x_C^{(1)}(t)\\ 0 & 1\end{bmatrix}$$

$$\left(\mathbf{e}^{(3)}(t)\ \ \mathbf{r}_c^{(3)}(t)\right)$$
$$=\left(\mathbf{e}^{I}(t)\ \ 0\right)\begin{bmatrix} R^{(1)}(t)R^{(2/1)}(t) & R^{(1)}(t)s_C^{(2/1)}+x_C^{(1)}(t)\\ 0 & 1\end{bmatrix}\begin{bmatrix} R^{(3/2)} & R^{(3/2)}s_C^{(3/2)}+s_C^{(2/1)}\\ 0 & 1\end{bmatrix}$$

$$R^{(2/1)}(t)=\begin{bmatrix} cos\xi & 0 & sin\xi\\ 0 & 1 & 0\\ -sin\xi & 0 & cos\xi\end{bmatrix}$$

$$\left(\mathbf{e}^{(3)}(t)\ \ \mathbf{r}_c^{(3)}(t)\right)=\left(\mathbf{e}^{I}(t)\ \ 0\right)E^{(3)}(t)$$

$$E^{(3)}(t)=\begin{bmatrix} R^{(1)}(t)R^{\left(\frac{2}{1}\right)}(t) & R^{(1)}(t)s_C^{\left(\frac{2}{1}\right)}+x_C^{(1)}(t)\\ 0 & 1\end{bmatrix}\begin{bmatrix} R^{\left(\frac{3}{2}\right)} & R^{\left(\frac{3}{2}\right)}s_C^{\left(\frac{3}{2}\right)}+s_C^{\left(\frac{2}{1}\right)}\\ 0 & 1\end{bmatrix}$$

$$E^{(3)}(t)$$
$$=\begin{bmatrix} R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t) & R^{(1)}(t)R^{(2/1)}\left(R^{(3/2)}\tilde{s}_c^{(3/2)}+s_C^{(2/1)}\right)+R^{(1)}(t)s_C^{(2/1)}+x_C^{(1)}(t)\\ 0 & 1\end{bmatrix}$$

$$E^{(3)}(t)^{-1} = \begin{bmatrix} \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T & -\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(R^{(1)}(t)R^{(2/1)}(t)\right)\left(R^{(3/2)}s_C^{(3/2)} + s_C^{(2/1} \\ 0 & 1 \end{bmatrix}$$

Now we take e-dot and multiply. I will only multiply the rows and columns I need for the term under consideration.

$$\tilde{\omega}^{(3)}(t) = \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(\dot{R}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t) + R^{(1)}(t)\dot{R}^{(2/1)}R^{(3/2)} + R^{(1)}(t)R^{(2/1)}\dot{R}^{(3/2)}\right)$$

$$\begin{aligned}\tilde{\omega}^{(3)} = {} & \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(\dot{R}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right) \\ & + \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \left(R^{(1)}(t)\dot{R}^{(2/1)}(t)R^{(3/2)}(t)\right) \\ & + \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T R^{(1)}(t)R^{(2/1)}(t)\dot{R}^{(3/2)}(t)\end{aligned}$$

$$\begin{aligned}\tilde{\omega}^{(3)}(t) = {} & \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \dot{R}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t) \\ & + \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \dot{R}^{(2/1)}(t)R^{(3/2)}(t) + \left(R^{(3/2)}(t)\right)^T \dot{R}^{(3/2)}(t)\end{aligned}$$

$$\begin{aligned}\tilde{\omega}^{(3)}(t) = {} & \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \tilde{\omega}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t) \\ & + \left(R^{(3/2)}(t)\right)^T \tilde{\omega}^{(2/1)}(t)R^{(3/2)}(t) + \tilde{\omega}^{(3/2)}(t)\end{aligned}$$

$$R^{(3/1)}(t) = R^{(2/1)}(t)R^{(3/2)}(t)$$

$$\tilde{\omega}^{(3)}(t) = \left(R^{(3/1)}(t)\right)^T \tilde{\omega}^{(1)}(t)R^{(3/1)}(t) + \left(R^{(3/2)}(t)\right)^T \tilde{\omega}^{(2/1)}(t)R^{(3/2)}(t) + \tilde{\omega}^{(3/2)}(t)$$

$$\omega^{(3)}(t) = \left(R^{(3/1)}(t)\right)^T \omega^{(1)}(t) + \left(R^{(3/2)}(t)\right)^T \omega^{(2/1)}(t) + \omega^{(3/2)}(t)$$

We have omega and we have shifted all terms to the right.

$$\omega^{(3)}(t) = \left(R^{(3/1)}(t)\right)^T \omega^{(1)}(t) + \left(R^{(3/2)}(t)\right)^T \dot{\phi}^{(2)}e_3 + \xi^{(3)}e_2$$

Now we turn to the POSITION

$$E^{(3)}(t) = \begin{bmatrix} R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t) & R^{(1)}(t)R^{(2/1)}\left(R^{(3/2)}s_C^{(3/2)} + s_C^{(2/1)}\right) + R^{(1)}(t)s_C^{(2/1)} + x_C^{(1)}(t) \\ 0 & 1 \end{bmatrix}$$

The element of the position is

$$\begin{aligned}\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \frac{d}{dt}\Big(R^{(1)}(t)R^{(2/1)}(t)\left(R^{(3/2)}s_C^{(3/2)} + s_C^{(2/1)}\right) \\ + R^{(1)}(t)s_C^{(2/1)} + x_C^{(1)}(t)\Big)\end{aligned}$$

$$\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \frac{d}{dt}\Big(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} + R^{(1)}(t)R^{(2/1)}(t)(t)s_C^{(2/1)} + R^{(1)}(t)s_C^{(2/1)} + x_C^{(1)}(t)\Big)$$

$$\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T$$

$$\Big(\dot{R}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} + R^{(1)}(t)\dot{R}^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} + R^{(1)}(t)R^{(2/1)}(t)\dot{R}^{(3/2)}(t)s_C^{(3/2)} + \dot{R}^{(1)}(t)R^{(2/1)}s_C^{(2/1)} + R^{(1)}(t)\dot{R}^{(2/1)}(t)s_C^{(2/1)} + \dot{R}^{(1)}(t)s_C^{(2/1)} + x_C^{(1)}(t)\Big)$$

$$\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T$$

$$\Big(\dot{R}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} + R^{(1)}(t)\dot{R}^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} + R^{(1)}(t)R^{(2/1)}(t)\dot{R}^{(3/2)}(t)s_C^{(3/2)} + \dot{R}^{(1)}(t)R^{(2/1)}(t)s_C^{(2/1)} + R^{(1)}(t)\dot{R}^{(2/1)}(t)s_C^{(2/1)} + \dot{R}^{(1)}(t)s_C^{(2/1)} + x_C^{(1)}(t)\Big)$$

$$\begin{aligned}\Big(&\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \dot{R}^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T R^{(1)}(t)\dot{R}^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)}\Big) \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T R^{(1)}(t)R^{(2/1)}(t)\dot{R}^{(3/2)}(t)s_C^{(3/2)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \dot{R}^{(1)}(t)R^{(2/1)}(t)s_C^{(2/1)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T R^{(1)}(t)\dot{R}^{(2/1)}(t)s_C^{(2/1)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T \dot{R}^{(1)}(t)s_C^{(2/1)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\end{aligned}$$

$$\begin{aligned}&\Big(\left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \widetilde{\omega}^{(1)}R^{(2/1)}(t)R^{(3/2)}(t)s_C^{(3/2)} \\ &+ \left(R^{(3/2)}(t)\right)^T \widetilde{\omega}^{(2/1)}R^{(3/2)}(t)s_C^{(3/2)} + \widetilde{\omega}^{(3/2)}s_C^{(3/2)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \widetilde{\omega}^{(1)}R^{(2/1)}(t)s_C^{(2/1)} \\ &+ \left(R^{(3/2)}(t)\right)^T \widetilde{\omega}^{(2/1)}s_C^{(2/1)} + \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \widetilde{\omega}^{(1)}s_C^{(2/1)} \\ &+ \left(R^{(3/2)}(t)\right)^T \left(R^{(2/1)}(t)\right)^T \left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\Big)\end{aligned}$$

$$R^{(3/1)} = R_2^{(2/1)}(\phi^{(2)})R_3^{(3/2)}(\xi^{(3)})$$

$$\left(\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}R^{(2/1)}(t)R^{(3/2)}(t)+\left(R^{(3/2)}(t)\right)^T\tilde{\omega}^{(2/1)}R^{(3/2)}(t)\right.\right.$$
$$\left.+\tilde{\omega}^{(3/2)}\right)s_C^{(3/2)}$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}R^{(2/1)}(t)+\left(R^{(3/2)}(t)\right)^T\tilde{\omega}^{(2/1)}\right.$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}\right)s_C^{(2/1)}$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\left(\left(\left(R^{(3/2)}(t)\right)^T\left(\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}R^{(2/1)}(t)+\tilde{\omega}^{(2/1)}\right)R^{(3/2)}+\tilde{\omega}^{(3/2)}\right)s_C^{(3/2)}\right.$$
$$+\left(R^{(3/2)}(t)\right)^T\left(\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}R^{(2/1)}(t)+\tilde{\omega}^{(2/1)}\right)$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}\right)s_C^{(2/1)}+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T$$
$$\left.+\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\left((\tilde{\omega}^{(3)})s_C^{(3/2)}+\left(\left(R^{(3/2)}(t)\right)^T\left(\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}R^{(2/1)}(t)+\tilde{\omega}^{(2/1)}\right)\right)\right.$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}\right)s_C^{(2/1)}$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\left((\tilde{\omega}^{(3)})s_C^{(3/2)}+\left(\left(R^{(3/2)}(t)\right)^T(\tilde{\omega}^{(2)})\right)s_C^{(2/1)}+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\tilde{\omega}^{(1)}\right)s_C^{(2/1)}\right.$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\mathbf{e}^{(3)}\left(\left(\tilde{s}_C^{(3/2)}\right)^T(\omega^{(3)})+\left(R^{(3/2)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T(\omega^{(2)})\right.$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

And now we want it in inertial:

$$e^{(2)}\left(R^{(3/2)}(t)\right)\left(\left(\tilde{s}_C^{(3/2)}\right)^T(\omega^{(3)})+\left(R^{(3/2)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T(\omega^{(2)})\right.$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$e^{(1)}\left(R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\left(\tilde{s}_C^{(3/2)}\right)^T\left(\omega^{(3)}\right)+\left(R^{(3/2)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\left(\omega^{(2)}\right)\right.$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\mathbf{e}^I\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\left(\tilde{s}_C^{(3/2)}\right)^T\left(\omega^{(3)}\right)+\left(R^{(3/2)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\left(\omega^{(2)}\right)\right.$$
$$+\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)$$
$$\left.+\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\mathbf{e}^I\left(\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\tilde{s}_C^{(3/2)}\right)^T\left(\omega^{(3)}\right)\right.$$
$$+\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(R^{(3/2)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\left(\omega^{(2)}\right)$$
$$+\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)$$
$$\left.+\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(R^{(3/2)}(t)\right)^T\left(R^{(2/1)}(t)\right)^T\left(R^{(1)}(t)\right)^T x_C^{(1)}(t)\right)$$

$$\mathbf{e}^I\left(\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\tilde{s}_C^{(3/2)}\right)^T\left(\omega^{(3)}\right)+\left(R^{(1)}(t)R^{(2/1)}(t)\right)\left(\tilde{s}_C^{(2/1)}\right)^T\left(\omega^{(2)}\right)\right.$$
$$\left.+\left(R^{(1)}(t)\right)\left(\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)+x_C^{(1)}(t)\right)$$

Summary

$$\dot{x}_C^{(1)}(t)=\dot{x}_C^{(1)}(t)$$

$$\omega^{(1)}(t)=\omega^{(1)}(t)$$

$$\dot{x}_C^{(2)}(t)=R^{(1)}\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}(t)+\dot{x}_C^{(1)}(t)$$

$$\omega^{(2)}(t)=\left(R^{(2/1)}(t)\right)^T\omega^{(1)}(t)+\dot{\phi}^{(2)}e_3$$

$$\dot{x}_C^{(3)}(t)=\left(\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\tilde{s}_C^{(3/2)}\right)^T\left(\omega^{(3)}\right)\right.$$
$$+\left(R^{(1)}(t)R^{(2/1)}(t)\right)\left(\tilde{s}_C^{(2/1)}\right)^T\left(\omega^{(2)}\right)+\left(R^{(1)}(t)\right)\left(\left(\tilde{s}_C^{(2/1)}\right)^T\omega^{(1)}\right)$$
$$\left.+x_C^{(1)}(t)\right)$$

$$\omega^{(3)}(t)=\left(R^{(3/1)}(t)\right)^T\omega^{(1)}(t)+\left(R^{(3/2)}(t)\right)^T\dot{\phi}^{(2)}e_3+\dot{\xi}^{(2)}e_2$$

A few last definitions:

$$R^{(3/2)}(t)=R^{(2/1)}(t)R^{(3/2)}(t)$$

$$\dot{x}_C^{(1)}(t) = \dot{x}_C^{(1)}(t)$$

$$\omega^{(1)}(t) = \omega^{(1)}(t)$$

$$\dot{x}_C^{(2)}(t) = R^{(1)}\left(\tilde{s}_C^{(2/1)}\right)^T \omega^{(1)}(t) + \dot{x}_C^{(1)}(t)$$

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t) + \dot{\phi}^{(2)} e_3$$

$$\dot{x}_C^{(3)}(t) = \left(\left(R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\right)\left(\tilde{s}_C^{(3/2)}\right)^T \left(\omega^{(3)}\right) + \left(R^{(1)}(t)R^{(2/1)}(t)\right)\left(\tilde{s}_C^{(2/1)}\right)^T \left(\omega^{(2)}\right) + \left(R^{(1)}(t)\right)\left(\left(\tilde{s}_C^{(2/1)}\right)^T \omega^{(1)}\right) + x_C^{(1)}(t)\right)$$

$$\omega^{(3)}(t) = \left(R^{(3/1)}(t)\right)^T \omega^{(1)}(t) + \left(R^{(3/2)}(t)\right)^T \dot{\phi}^{(2)} e_3 + \dot{\xi}^{(3)} e_2$$

Now build the B:

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix} \qquad \{\dot{X}(t)\} = \{\dot{q}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(2)} \end{Bmatrix}$$

State Q

$$\{Q(t)\} = \begin{pmatrix} F_C^{(1)I}(t) \\ M_C^{(1)}(t) \\ F_C^{(2)I}(t) \\ M_C^{(2)}(t) \\ F_C^{(3)I}(t) \\ M_C^{(3)}(t) \end{pmatrix} = \begin{pmatrix} F^{(w)I}(t) + F_b^I - m^{(1)} g e_3 \\ M^{(w)}(t) - M_m^{(c)}(t) e_3 \\ -m^{(2)} g e_3 \\ M_m^{(c)}(t) e_3 - M_m^{(b)}(t) e_2 \\ -m^{(3)} g e_3 \\ M_m^{(b)}(t) e_2 \end{pmatrix}$$

Reading DOWN AND ACROSS IN ORDER

$$F^{(w)I} = Force\ from\ waves$$
$$F_b^I = Force\ from\ buoyance$$

$$m^{(1)} g e_3 = Force\ down\ from\ gravity\ on\ ship$$

$$M^{(w)}(t) = Moment\ from\ waves$$

$$-M_m^{(c)}(t) e_3 = Reverse\ moment\ on\ ship\ from\ motor\ that\ turns\ the\ crane$$

$$-m^{(2)}ge_3 = Force\ gravity\ down\ on\ the\ crane$$

$$M_m^{(c)}(t)e_3 = Moment\ that\ turns\ the\ crane$$

$$-M_m^{(b)}(t)e_2 = Reverse\ moment\ on\ crane\ from\ boom$$

$$-m^{(3)}ge_3 = Force\ gravity\ down\ on\ boom$$

$$M_m^{(b)}(t)e_2 = Moment\ on\ crane\ that\ turns\ the\ boom$$

Now we need the rest

$$[D] = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & \overleftrightarrow{\omega^{(1)}(t)} & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & \overleftrightarrow{\omega^{(2)}(t)} & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & \overleftrightarrow{\omega^{(3)}(t)} \end{bmatrix}$$

$$[M] = \begin{bmatrix} m^{(1)}I_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & J_C^{(1)} & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & m^{(2)}I_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & J_C^{(2)} & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & m^{(3)}I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & J_C^{(3)} \end{bmatrix}$$

We need this:

$$B^T MB\ddot{q} + B^T\left(M\dot{B} + DMB\right)\dot{q} - F^* = 0$$

We solve for the q terms

$$B^T MB\begin{Bmatrix} \ddot{x}_C^{(1)}(t) \\ \dot{\omega}^{(1)}(t) \\ \ddot{\phi}^{(2)} \\ \ddot{\xi}^{(3)} \end{Bmatrix} + B^T\left(M\dot{B} + DMB\right)\begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(3)} \end{Bmatrix} - F^* = 0 \dot{x}_C^{(1)}(t) = \dot{x}_C^{(1)}(t)$$

$$\omega^{(1)}(t) = \omega^{(1)}(t)$$

$$\dot{x}_C^{(2)}(t) = R^{(1)}(t)\left(\tilde{s}_C^{(2/1)}\right)\omega^{(1)}(t) + \dot{x}_C^{(1)}(t)$$

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T\omega^{(1)}(t) + \dot{\phi}^{(2)}e_3$$

$$\dot{x}_C^{(3)} = R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\left(\overleftrightarrow{s_C^{(3/2)}}^T\right)\omega^{(3)} + R^{(1)}(t)R^{(2/1)}(t)\left(\overleftrightarrow{s_C^{(2/1)}}^T\right)\omega^{(2)} + R^{(1)}(t)\left(\overleftrightarrow{s_C^{(2/1)}}^T\right)\omega^{(1)} + \dot{x}_C^{(1)}$$

$$\omega^{(3)}(t) = \left(R^{\left(\frac{3}{1}\right)}(t)\right)^T \omega^{(1)}(t) + \left(R^{\left(\frac{3}{2}\right)}(t)\right)^T \dot{\phi}^{(2)}e_3 + \dot{\xi}^{(3)}e_2$$

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix} \qquad \{\dot{X}(t)\} = \{\dot{q}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(2)} \end{Bmatrix}$$

Note that for the x terms above, you must replace the absolute omega for that with the omega to get it in terms of psi and phi. By that, I mean that the YELLOW and BLUE term above must be inserted into the green term.
And then we can construct the B:

$$\{\dot{X}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(2)}(t) \\ \dot{x}_C^{(1)}(t) \\ \omega^{(3)}(t) \end{Bmatrix} \qquad \{\dot{X}(t)\} = \{\dot{q}(t)\} = \begin{Bmatrix} \dot{x}_C^{(1)}(t) \\ \omega^{(1)}(t) \\ \dot{\phi}^{(2)} \\ \dot{\xi}^{(3)} \end{Bmatrix}$$

$$\dot{x}_C^{(1)}(t) = \dot{x}_C^{(1)}(t)$$

$$\omega^{(1)}(t) = \omega^{(1)}(t)$$

$$\dot{x}_C^{(2)}(t) = R^{(1)}(t)\left(\tilde{s}_C^{(2/1)}\right)\omega^{(1)}(t) + \dot{x}_C^{(1)}(t)$$

$$\omega^{(2)}(t) = \left(R^{(2/1)}(t)\right)^T \omega^{(1)}(t) + \dot{\phi}^{(2)}e_3$$

$$\dot{x}_C^{(3)} = R^{(1)}(t)R^{(2/1)}(t)R^{(3/2)}(t)\left(\overleftrightarrow{s_C^{(3/2)}}^T\right)\omega^{(3)} + R^{(1)}(t)R^{(2/1)}(t)\left(\overleftrightarrow{s_C^{(2/1)}}^T\right)\omega^{(2)} + R^{(1)}(t)\left(\overleftrightarrow{s_C^{(2/1)}}^T\right)\omega^{(1)} + \dot{x}_C^{(1)}$$

$$\omega^{(3)}(t) = \left(R^{(3/1)}(t)\right)^T \omega^{(1)}(t) + \left(R^{(3/2)}(t)\right)^T \dot{\phi}^{(2)}e_3 + \dot{\xi}^{(3)}e_2$$

$$[B(t)] = \begin{bmatrix} I_3 & 0_3 & 0_1 & 0_1 \\ 0_3 & I_3 & 0_1 & 0_1 \\ I_3 & R^{(1)}(t)\left(s_C^{(2/1)\longleftrightarrow T}\right) & 0_1 & 0_1 \\ 0_3 & \left(R^{(2/1)}(t)\right)^T & e_3 & 0_1 \\ I_3 & B_{52} & B_{53} & B_{54} \\ 0_3 & \left(R^{(3/1)}(t)\right)^T & \left(R^{(3/2)}(t)\right)^T e_3 & e_2 \end{bmatrix}$$

In the above:

$$0_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad 0_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

$$\boldsymbol{B_{52}} = \boldsymbol{R^{(1)}(t)R^{(3/1)}(t)}\left(\boldsymbol{s_C^{(3/2)\longleftrightarrow T}}\right)\left(\boldsymbol{R^{(3/1)}(t)}\right)^{\boldsymbol{T}} +$$

$$\boldsymbol{R^{(1)}(t)R^{(2/1)}(t)}\left(\boldsymbol{s_C^{(2/1)\longleftrightarrow T}}\right)\left(\boldsymbol{R^{(2/1)}(t)}\right)^{\boldsymbol{T}} + \boldsymbol{R^{(1)}(t)}\left(\boldsymbol{s_C^{(2/1)\longleftrightarrow T}}\right).$$

$$\boldsymbol{B_{53}} = R^{(1)}(t)R^{(3/1)}(t)\left(s_C^{(3/2)\longleftrightarrow T}\right)\left(R^{(3/2)}(t)\right)^T e_3 + R^{(1)}(t)R^{(2/1)}(t)\left(s_C^{(2/1)\longleftrightarrow T}\right)e_3$$

$$\boldsymbol{B_{54}} = R^{(1)}(t)R^{\left(\frac{3}{1}\right)}(t)\left(s_C^{\left(\frac{3}{2}\right)\longleftrightarrow T}\right)e_2$$

The fact is that we will get OMEGA at every time step

What is $R^{(1)}(t)$. We need the *R* to create that *A* matrix, but more important, we need the *R* to show and visualize how the boat moves. I apply the rotation matrix to the boat and I will see it rock.

How to get the *R*:
Well, we must go all the way back to Chapter 4.

We conclude this section with a discussion on how to obtain the rotation matrix, given the angular velocity matrix. Equation (4.22b) reveals that the inertial frame observer can compute the angular velocity matrix using $R^T\dot{R}$.

From the viewpoint of moving frame observers, the frame rotation, *R*(t) , can be computed by integration from the measured angular acceleration and angular velocity. For a moving frame observer who has measured $\overleftrightarrow{\omega(t)}$, (4.22b) gives the differential equation for $R(t)$:

$$\dot{R} = \frac{d}{dt}R(t) = R(t)\overleftrightarrow{\omega(t)} \tag{4.25a}$$

$$\dot{R} = \frac{d}{dt}R(t) = R(t)\overleftrightarrow{\omega(t)} \tag{4.25a}$$

(4.25a)

with the initial condition that *R* is the identity matrix at *t* = 0:

$$R(0) = I_d \tag{4.25b}$$

If the angular velocity remains constant, $\omega(t) = \omega(0)$, the solution can be integrated analytically:

$$R(t) = exp\left(\overleftrightarrow{\omega(0)}t\right) \tag{4.26}$$

However, for time-dependent angular velocity, (4.25a) must be integrated numerically to obtain $R(t)$. Although, we presented an exponential matrix in (4.26), it will not be used extensively in the subsequent sections. Exponential matrices may be explained in any undergraduate textbooks on linear algebra or linear control theory.

So there are two new things….

1. What is: A exp $\left(\overleftrightarrow{\omega(0)}t\right)$.
2. How do I know it is the solution

Our $\overleftrightarrow{\omega(0)}$ for the boat is really the $\overleftrightarrow{\omega_n}$. And we will assume it is constant to the next step.
And we will find R at each of the next time steps as:

$$R(t) = exp(\overleftrightarrow{\omega_n}t_n)$$

It is not a “spectacular coincidence” that there is a solution and that it involves and exponential. This exponential is more than you realize. Is the function that links the SO(3) Group with the so(3) algebra. All we need is a faster way to get the exponential
We turn to the Cayleigh Hamilton theorem (and this is where we need eigenvalues)
And we do not have to solve as a series. We use this: put in equation from Tennis Racket paper.