JULIO ARANTES
PEDRO HENRIQUE DIAS SILVA

MODELAGEM, CONTROLE E DESENVOLVIMENTO DE UM SIMULADOR DE GUINDASTE

São Paulo
2018

JULIO ARANTES
PEDRO HENRIQUE DIAS SILVA

MODELAGEM, CONTROLE E DESENVOLVIMENTO DE UM SIMULADOR DE GUINDASTE

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

São Paulo
2018

JULIO ARANTES
PEDRO HENRIQUE DIAS SILVA

MODELAGEM, CONTROLE E DESENVOLVIMENTO DE UM SIMULADOR DE GUINDASTE

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

Orientador:

Prof. Dr. Eduardo Aoun Tannuri

São Paulo
2018

RESUMO

O aumento das atividades offshore, como plataformas e navios, trazem também a necessidade do desenvolvimento de atividades para o suporte de novos profissionais ou mesmo treinamento de profissionais experientes para equipamentos mais modernos. Com isso, a importância das pesquisas voltadas à construção e evolução de simuladores tem crescido significantemente. Assim, este projeto tem como objetivo modelar, controlar e desenvolver um simulador de guindaste para o Tanque de Provas Numérico da Universidade de São Paulo. O trabalho apresenta um simulador com dinâmica condizente à realidade, além de estar implementado em conjunto com os recursos visuais e de operação já presentes no simulador atual.

Palavras-Chave – Simulador de Guindaste, Controle, Modelagem Dinâmica.

ABSTRACT

The increase in offshore activities, such as platforms and ships, also bring the need for the development of activities to support new professionals or even training of experienced professionals for more modern equipment. With this, the importance of the research oriented to the construction and evolution of simulators has grown significantly. Thus, this project aims to model, control and develop a crane simulator for the Numerical Offshore Tank of the University of São Paulo. The work presents a simulator with dynamics consistent with reality, besides being implemented in set with the visual and operating features already present in the current simulator.

Keywords – Crane Simulator, Control, Dynamic Modeling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos de Caracteres Romanos

A_x	Área no Plano yz
A_y	Área no Plano xz
A_z	Área no Plano xy
C_s	Coeficiente de Forma
dx	Posição da base do Guindaste em x
dy	Posição da base do Guindaste em y
dz	Posição da base do Guindaste em z
$\vec{F}_j^{ext}$	Forças Externas
F_{wx}	Força do Vento no sentido x
F_{wy}	Força do Vento no sentido y
F_{wz}	Força do Vento no sentido z
g	Aceleração Gravitacional
I	Momento de Inércia
k	Constante Elástica
K_p	Ganho Proporcional
K_i	Ganho Integral
K_d	Ganho Derivativo
L	Função Lagrangiana
l	Comprimento do Cabo
$\dot{l}$	Velocidade do Cabo
L	Comprimento da Lança do Guindaste
M	Massa do Guindaste
m	Massa da Carga
Q	Forças Generalizadas
q_i	Coordenadas Generalizadas
$\vec{r}$	Vetor Direcional
t	Tempo
T	Energia Cinética
T	Tração do Cabo
T_i	Fator de Ajuste Integrativo
T_d	Fator de Ajuste Derivativo
U_{mx}	Velocidade Média do Vento em x
U_{my}	Velocidade Média do Vento em y
U_{mz}	Velocidade Média do Vento em z
U	Energia Potencial

v	Velocidade do Centro de massa do Corpo
V_{carga}	Valor Absoluto da Velocidade da Carga
x_{base}	Posição do Guindaste em x
x_{carga}	Posição da Carga em x
$\dot{x}_{carga}$	Velocidade da Carga em x
x_{sup}	Posição da extremidade superior da lança do Guindaste em x
$\dot{x}_{sup}$	Velocidade da extremidade superior da lança do Guindaste em x
y_{base}	Posição do Guindaste em y
y_{carga}	Posição da Carga em y
$\dot{y}_{carga}$	Velocidade da Carga em y
y_{sup}	Posição da extremidade superior da lança do Guindaste em y
$\dot{y}_{sup}$	Velocidade da extremidade superior da lança do Guindaste em y
z_{base}	Posição do Guindaste em z
z_{carga}	Posição da Carga em z
$\dot{z}_{carga}$	Velocidade da Carga em z
z_{sup}	Posição da extremidade superior da lança do Guindaste em z
$\dot{z}_{sup}$	Velocidade da extremidade superior da lança do Guindaste em z
z	Altura do Corpo

Símbolos de Caracteres Gregos

α	Ângulo do Guindaste em relação ao plano xy
β_1	Ângulo de posição da carga com relação ao plano xz
β_2	Ângulo de posição da carga com relação ao plano yz
γ	Ângulo de Rotação em z com relação a x
δ	Deformação Elástica
θ	Rotação da Embarcação em y (Pitch)
ρ	Densidade do Ar
ϕ	Rotação da Embarcação em x (Roll)
ψ	Rotação da Embarcação em z (Yaw)
ω	Velocidade Angular

LISTA DE ACRÔNIMOS

Proporcional	-	P
Proporcional, Integral	-	PI
Proporcional, Derivativo	-	PD
Proporcional, Integral, Derivativo	-	PID
Simulador Marítimo Hidroviário	-	SMH
Tanque de Provas Numérico	-	TPN
Universidade de São Paulo	-	USP

SUMÁRIO

1 INTRODUÇÃO

Os Sistemas Offshore são sistemas cuja instalação e operação acontecem em alto mar, como por exemplo, plataformas petrolíferas, navios petrolíferos ou mesmo grandes embarcações. Os dois maiores ramos de atividades de sistemas offshore são: Petrolíferas e navios de transporte.

(a) Foto aérea de uma plataforma petrolífera

(b) Foto aérea de um navio petrolífero

Figura 1: Instalações petrolíferas offshore [1]

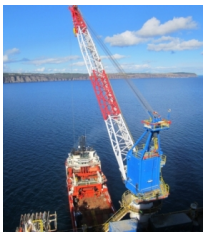
O comércio de petróleo está, provavelmente, em seu ápice econômico, o que torna o estudo e desenvolvimento de seus processos um tópico de extrema relevância nos últimos anos. Estima-se que, cerca de, 7850 plataformas de petróleo estejam em operação no mundo [5] e com crescimento de 2% ao ano [2] da produção mundial de petróleo, o mercado de combustíveis fósseis ainda mostra que é e será por um longo período o pilar na matriz energética global.

Figura 2: Produção mundial diária de petróleo [2]

Tanto na produção petrolífera, como no transporte de carga em embarcações, a necessidade da manipulação de “conteiners”, ou mesmo de objetos de grande porte, é muito significativa, o que torna dentro deste ramo de pesquisa, o desenvolvimento de guindastes embarcados um dos tópicos relevantes neste meio. Os principais tipos de guindaste presentes na área portuária e de embarcações são [6]:

(a) Guindaste Rotacional

(b) Guindaste Bidirecional



(c) Guindaste tipo ”boom”

Figura 3: Diferentes Tipos de Guindaste

O guindaste que será abordado neste projeto é o “Boom Crane”, que consiste de uma estrutura rígida inclinada e um cabo flexível que suporta a carga. Esta estrutura está embarcada em um navio petrolífero e é responsável por transportar cargas desta embarcação para plataformas de petróleo ou portos, ou vice-versa. O operador deste guindaste tem o controle sob sua inclinação, rotação ou içamento da carga, tendo assim a necessidade de manipular o objeto com esses comandos. De modo a possibilitar o treinamento de novos operadores, de qualificar operadores experientes ou mesmo de testar novos equipamentos, o desenvolvimento de simuladores tem se tornado um ramo de pesquisa crescente em inúmeras áreas. Especificamente para este projeto, o trabalho terá como foco um simulador desenvolvido no Tanque de Provas Numérico (TPN) da Escola Politécnica da Universidade de São Paulo, destinado a simulação de guindastes embarcados em navios de grande porte. A figura 4 mostra um dos ambientes do laboratório [7]:

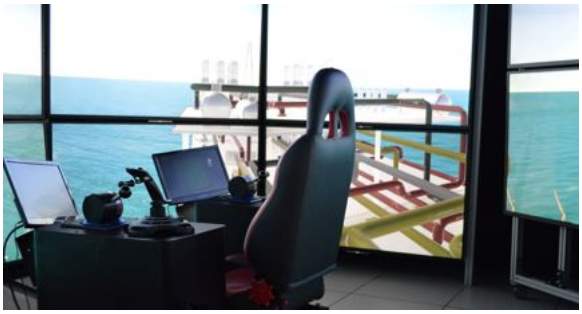


Figura 4: Cabine do Simulador de Guindaste (SMH-TPN-USP)

Portanto, este projeto tem como objetivo modelar dinamicamente o guindaste presente no simulador, projetar um controle de modo a melhorar sua performance e deixá-lo mais

realístico, e por fim, desenvolver um simulador computacional instalado no simulador de guindastes e rebocadores presente no TPN para esta dinâmica encontrada.

O design do guindaste é considerado um dos aspectos mais complexos do projeto, já que apresenta efeitos hidrodinâmicos e respostas dinâmicas diferenciadas de sistemas terrestres. Tal complexidade vem atrelada a presença da não linearidade de inúmeros processos no decorrer do trabalho [8].

O presente trabalho está dividido em três partes, a primeira consiste em uma revisão dos aspectos teóricos necessários para o desenvolvimento da modelagem e do controlador para o simulador. A segunda parte tange a aplicação dos conceitos teóricos, sendo feito a simulação da dinâmica do guindaste, o design do controlador e a implementação da estrutura computacional do simulador. Por fim, dedicamos a terceira e última parte para as conclusões do trabalho e abrimos espaço para futuros desenvolvimentos.

PARTE I

TEORIA

2 FUNDAMENTOS TEÓRICOS

2.1 Modelagem

2.1.1 Equações de Lagrange

As equações de Lagrange são equações diferenciais que combinam a conservação do momento linear com a conservação de energia de um sistema. Tais equações correlacionam a energia total, cinética mais potencial, com as forças externas, obtendo assim a dinâmica completa do modelo [9]. A resolução destas equações fornece a trajetória do sistema de partícula em questão, ou seja, o parâmetro intrínseco ao seu equacionamento [9].

As equações de Lagrange são expressas por [10]:

$$\frac{\delta}{\delta t}\left(\frac{\delta L}{\delta \dot{q}_i}\right) - \frac{\delta L}{\delta q_i} = Q_i \tag{2.1}$$

Em que L é a função Lagrangiana, Q as forças generalizadas não conservativas e q_i as coordenadas generalizadas independentes [10]. A função Lagrangiana é definida pela diferença entre a energia cinética e a potencial, sendo assim expressa por:

$$L = T - U \tag{2.2}$$

Onde T é a energia cinética e U a energia potencial. A energia cinética de um corpo rígido é uma função das coordenadas generalizadas, de suas velocidades e do tempo:

$$T = T(q_i, \dot{q}_i, t) \tag{2.3}$$

E é analiticamente representada por:

$$T = \frac{1}{2}mv^2 + \frac{1}{2}I\omega^2 \tag{2.4}$$

Sendo m a massa do corpo, v a velocidade do centro de massa do corpo, I o momento de inércia sobre o centro de massa do corpo e ω a velocidade angular do corpo [10]. A energia potencial de um corpo rígido é função das coordenadas generalizadas e do tempo:

$$U = U(q_i, t) \tag{2.5}$$

Além disso, pode ser obtida pela soma das energias potenciais elásticas e gravitacionais, e é analiticamente representada por:

$$U = \frac{1}{2}k\delta^2 + mgz \tag{2.6}$$

Sendo k a constante elástica, δ a deformação elástica a partir da configuração não deformada, m a massa do corpo, g a aceleração da gravidade e z a altura do centro de massa do corpo [10]. Com relação as forças generalizadas não conservativas, podemos interpretá-las como as forças externas aplicadas no corpo na direção de cada coordenada generalizada, equacionada por:

$$Q = \sum_{j}^{n} \vec{F_j^{ext}} \frac{\delta \vec{r}}{\delta q_i} \tag{2.7}$$

Em que $\vec{F_j^{ext}}$ são as forças externas e $\vec{r}$ o vetor direcional.

2.1.2 Linearização por Série de Taylor

A maioria dos sistemas e modelos obtidos a partir do equacionamento dinâmico de estruturas reais são representados por equações não-lineares. Assim, de modo a simplificar a resolução numérica sem comprometer o resultado final, ou mesmo, para obter uma gama maior de métodos de manipulação ou aplicação sob o modelo obtido, lineariza-se o sistema. A linearização de uma equação diferencial de ordem superior para uma de primeira ordem pode ser feita por inúmeros métodos, dentre eles, a expansão em Série de Taylor, um dos mais utilizados. Tal linearização adquire um modelo linearizado em torno de um ponto de operação, em que, normalmente, tende a ser um ponto de equilíbrio do sistema [11]. A expansão em Série de Taylor de uma função não-linear aleatória $f(x)$ é dada por:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \dots \tag{2.8}$$

Sendo x_0 o ponto de aplicação.

Para um sistema linear, assume-se que o valor da incógnita x será aproximadamente igual ao seu valor no ponto de operação x_0, logo podemos concluir que a soma dos termos ordem superior, $\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$, sejam aproximadamente 0. Logo, temos que:

$$\sum_{j=2}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n \approx 0 \tag{2.9}$$

Assim, utilizando 2.8 e 2.9 obtemos a função da aproximação de primeira ordem, dita linearizada:

$$f(x) \approx g(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) \tag{2.10}$$

Para exemplificar essa aplicação, será apresentado abaixo duas linearizações que serão utilizadas durante a modelagem do sistema em questão.

$$\begin{aligned} sin(x) &\approx sin(x_0) + \frac{cos(x_0)}{1!}(x-x_0) \\ cos(x) &\approx cos(x_0) - \frac{sin(x_0)}{1!}(x-x_0) \end{aligned} \tag{2.11}$$

Estas equações produzem uma linearização muito comum em diversas aplicações em engenharia. Trata-se de que para ângulos pequenos vale:

$$\begin{aligned} sin(\theta) &\approx \theta \\ cos(\theta) &\approx 1 \end{aligned} \tag{2.12}$$

2.2 Controle

2.2.1 Controlador Proporcional Integral Derivativo

O controlador Proporcional Integral Derivativo, PID, trata-se de um método empírico e de fácil implementação, que na maioria dos casos, apresenta um bom desempenho, o que, concomitantemente ao fator de sua facilidade de projeto, torna-o um artifício de ótimo custo benefício. O esquema de um controlador PID pode ser visto a seguir.

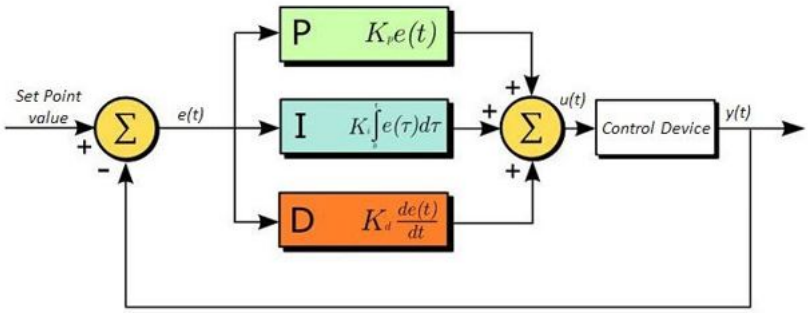


Figura 5: Diagrama de Blocos PID

A formulação de um controlador PID pode ser dada de duas formas distintas, geralmente, sendo elas: a relação entre os sinais de entrada e de saída por meio de uma equação diferencial-integral; ou por meio de sua função transferência, que seria a transformada de Laplace da equação citada anteriormente. A equação diferencial-integral que representa a relação entre os sinais de entrada e de saída um controlador PID é dada por:

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int e(t)dt + K_p T_d \frac{de(t)}{dt} \tag{2.13}$$

Em que K_p é o ganho proporcional, T_i o fator de ajuste do ganho proporcional para o integrativo, T_d o fator de ajuste do ganho proporcional para o derivativo, $e(t)$ o sinal de entrada e o $u(t)$ o sinal de saída. Aplicando a Transformada de Laplace, temos a função de transferência do controlador PID, como:

$$H(s) = \frac{U(s)}{E(s)} = K_p[1 + \frac{1}{T_i s} + T_d s] \tag{2.14}$$

Sendo $H(s)$ a função transferência do controlador, $U(s)$ a saída e $E(s)$ a entrada. Para obter o controlador PID adequado para o sistema em que deseja aplicá-lo, deve-se ajustar as constantes K_p, T_d e T_i para o conjunto do sistema presente, para que o projeto, em malha fechada, seja estável e apresente o desempenho desejado pelo usuário. Desse modo, dois métodos de obtenção dessas constantes se destacam dos demais: Método de Sintonia de Ziegler & Nichols; e Método do Ciclo Máximo [12].

2.2.2 Método de Sintonia – Ziegler & Nichols

Este método consiste na aplicação de uma entrada em degrau sob o sistema em malha aberta, e com relação a resposta do sistema, obtém-se os parâmetros presentes na figura a seguir.

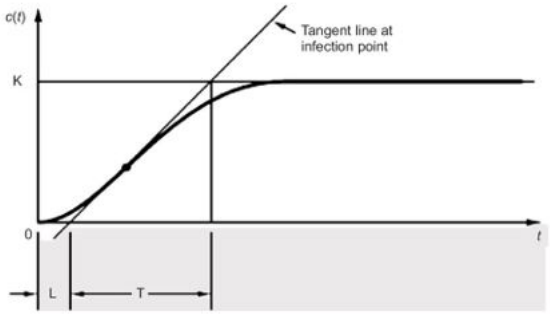


Figura 6: Resposta do Sistema a uma entrada degrau

Com estes parâmetros obtidos, utiliza-se da tabela a seguir para conseguir as constantes do controlador PID e assim as informações necessárias para sua implementação [12].

Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	inf	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Tabela 1: Parâmetros de Ziegler & Nichols [4]

2.2.3 Método do Ciclo Máximo

Este método consiste em aplicar uma entrada na forma de degrau sob o sistema em malha fechada, na presença de um controlador proporcional. Feito isso, altera-se a constante do ganho proporcional de forma a obter o limiar de um sistema oscilante, como visto na figura a seguir.

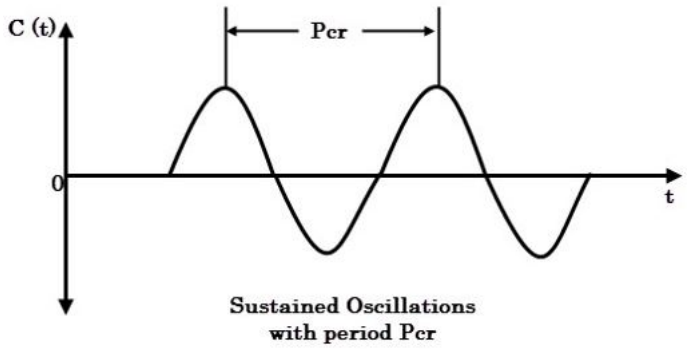


Figura 7: Resposta do Sistema para entrada degrau em limiar de estabilidade [3]

Assim, pode-se obter o ganho no limiar da instabilidade, K_{cr}, e o período de oscilação do sinal no limite da estabilidade, P_{cr}. Com estes dois parâmetros, pode-se calcular os parâmetros do controlador PID para este sistema, a partir da tabela abaixo, e assim obter o controlador desejado.

Controller	K_p	T_i	T_d
P	$0.5K_{er}$		0
PI	$0.45K_{er}$	$\frac{1}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$height

Tabela 2: Parâmetros do Ciclo Máximo

2.2.4 Compensador de Heave

O controle de Heave tem como objetivo controlar a posição e/ou velocidade de elevação da carga do guindaste. Assim, como método para obter o controlador temos:

2.2.4.1 Controle de Posição Simples

Para a confecção do controle de posição iniciamos com a hipótese do cabo não ser flexível, assim podemos equacionar a dinâmica do cabo como:

$$l = \int \dot{l} dt \tag{2.15}$$

Ou seja, que o tamanho do cabo é igual a integral da taxa de variação do comprimento do cabo. Em guindastes é extremamente comum que consigamos atuar diretamente sobre a taxa de variação, os comandos do cockpit atuam diretamente na velocidade do motor que por sua vez é diretamente proporcional à velocidade do cabo, logo esta será manipulada

com o objetivo de controlar a altura da carga em relação à um referencial inercial solidário à Terra (hipoteticamente inercial).

Assim podemos partir para a função transferência da equação acima, com a velocidade do cabo como entrada e a dimensão do cabo como saída, temos:

$$G(s) = \frac{L(s)}{sL(s)} = \frac{1}{s} \tag{2.16}$$

Obs.: o controlador mais utilizado para um controle simples de posição é um controlador proporcional [13].

2.3 Modelo Dinâmico

O modelo dinâmico que será apresentado refere-se a um guindaste do tipo "Boom Crane", muito utilizado em navios e plataformas de petróleo. Sua estrutura é constituída de três motores e dois corpos rígidos. Os motores são responsáveis pela rotação da lança do guindaste, pela inclinação da lança e pelo içamento da carga, e os dois corpos rígidos são a estrutura da lança e o cabo de içamento. Tal modelagem pode ser verificada na figura 8:

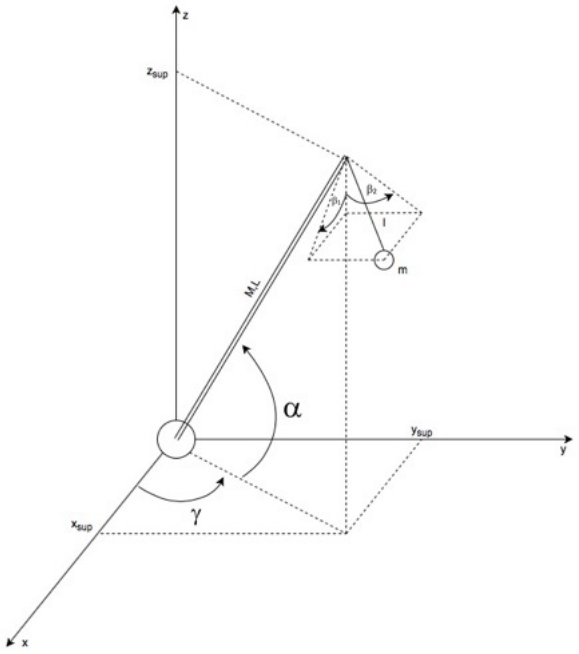


Figura 8: Modelo Corpo Rígido do Guindaste

Na mesma ilustração, podemos verificar as variáveis presentes no modelo, que podem ser descritas como:

- x_{base} = posição do guindaste no eixo x
- y_{base} = posição do guindaste no eixo y
- z_{base} = posição do guindaste no eixo z
- α = ângulo de inclinação do guindaste com relação ao plano xy
- γ = ângulo de rotação do guindaste no eixo z com relação ao eixo x
- L = comprimento da lança do guindaste
- M = massa da lança do guindaste
- x_{sup} =posição da extremidade superior da lança do guindaste no eixo x
- y_{sup} =posição da extremidade superior da lança do guindaste no eixo y
- z_{sup} =posição da extremidade superior da lança do guindaste no eixo z
- β_1 = ângulo de posição da carga com relação ao plano xz
- β_2 = ângulo de posição da carga com relação ao plano yz
- l = comprimento do cabo de içamento da carga
- m = massa da carga
- g = aceleração da gravidade $[g = 9.8\frac{m}{s^2}]$

A modelagem da estrutura apresentada anteriormente será realizada de uma forma direta, ou seja, partindo do início da estrutura fixa à embarcação até a carga. Sendo assim, temos inicialmente a base do guindaste fixa à algum local da embarcação, e esta embarcação apresenta deslocamentos no eixo x, y e z, além de rotações nas componentes Roll, Pitch, Yaw, devido às forças externas como vento, ondas, correntes, etc. Podemos verificar tais componentes na figura 9.

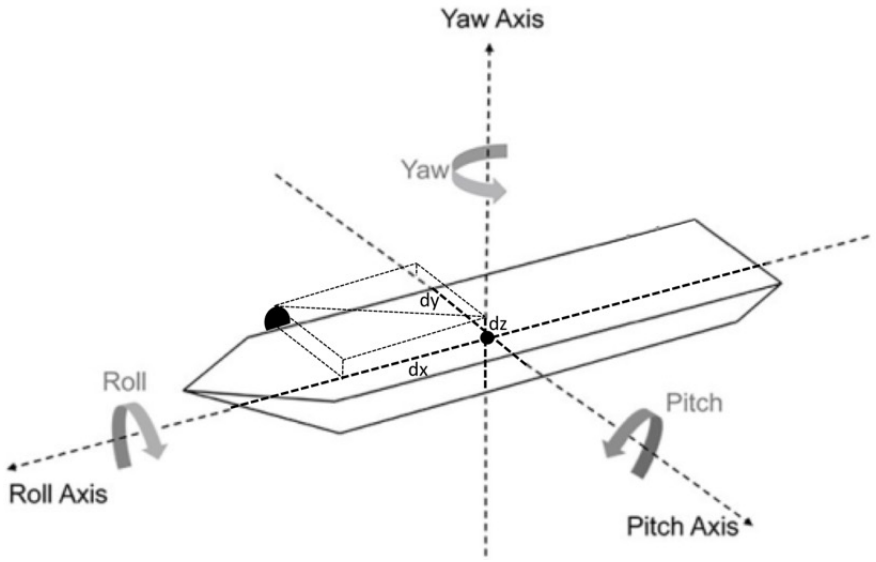


Figura 9: Ângulos de Euler em uma embarcação

sendo

- x = posição da embarcação em x
- y = posição da embarcação em y
- z = posição da embarcação em z
- ϕ = rotação da embarcação em x (roll)
- θ = rotação da embarcação em y (pitch)
- ψ = rotação da embarcação em z (yaw)
- dx = posição da base do guindaste com relação a embarcação no eixo x
- dy = posição da base do guindaste com relação a embarcação no eixo y
- dz = posição da base do guindaste com relação a embarcação no eixo z

Logo, para determinarmos as coordenadas da base do guindaste com relação a posição da embarcação, temos de obter a matriz de rotação da embarcação, R. Assim, teríamos:

$$\begin{bmatrix} x_{base} \\ y_{base} \\ z_{base} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + R \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.17}$$

com

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.18}$$

$$R_y = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{2.19}$$

$$R_z = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

$$R = R_z R_y R_x \tag{2.21}$$

$$\begin{aligned} R_{11} &= \cos\psi\cos\theta \\ R_{12} &= \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi \\ R_{13} &= \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ R_{21} &= \sin\psi\cos\theta \\ R_{22} &= \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi \\ R_{23} &= \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ R_{31} &= -\sin\theta \\ R_{32} &= \cos\theta\sin\phi \\ R_{33} &= \cos\theta\cos\phi \end{aligned} \tag{2.22}$$

A seguir, com base na mesma equação da posição da base do guindaste, podemos obter a velocidade do mesmo ponto, derivando a equação no tempo. Assim, temos:

$$\begin{bmatrix} \dot{x}_{base} \\ \dot{y}_{base} \\ \dot{z}_{base} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \dot{R} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.23}$$

Onde $\dot{R}$ é a primeira derivada temporal da matriz R.

$$\dot{R} = \begin{bmatrix} \dot{R_{11}} & \dot{R_{12}} & \dot{R_{13}} \\ \dot{R_{21}} & \dot{R_{22}} & \dot{R_{23}} \\ \dot{R_{31}} & \dot{R_{32}} & \dot{R_{33}} \end{bmatrix} \tag{2.24}$$

$$\begin{aligned}
\dot{R}_{11} &= -\sin\psi\cos\theta\dot{\psi} - \cos\psi\sin\theta\dot{\theta} \\
\dot{R}_{12} &= -\sin\psi\sin\theta\sin\phi\dot{\psi} + \cos\psi\cos\theta\sin\phi\dot{\theta} + \cos\psi\sin\theta\cos\phi\dot{\phi} - \cos\psi\cos\phi\dot{\psi} + \sin\psi\sin\phi\dot{\phi} \\
\dot{R}_{13} &= -\sin\psi\sin\theta\cos\phi\dot{\psi} + \cos\psi\cos\theta\cos\phi\dot{\theta} - \cos\psi\sin\theta\sin\phi\dot{\phi} + \cos\psi\sin\phi\dot{\psi} + \sin\psi\cos\phi\dot{\phi} \\
\dot{R}_{21} &= \cos\psi\cos\theta\dot{\psi} - \sin\psi\sin\theta\dot{\theta} \\
\dot{R}_{22} &= \cos\psi\sin\theta\sin\phi\dot{\psi} + \sin\psi\cos\theta\sin\phi\dot{\theta} + \sin\psi\sin\theta\cos\phi\dot{\phi} - \sin\psi\cos\phi\dot{\psi} - \cos\psi\sin\phi\dot{\phi} \\
\dot{R}_{23} &= \cos\psi\sin\theta\cos\phi\dot{\psi} + \sin\psi\cos\theta\cos\phi\dot{\theta} - \sin\psi\sin\theta\sin\phi\dot{\phi} + \sin\psi\sin\phi\dot{\psi} - \cos\psi\cos\phi\dot{\phi} \\
\dot{R}_{31} &= -\cos\theta\dot{\theta} \\
\dot{R}_{32} &= -\sin\theta\sin\phi\dot{\theta} + \cos\theta\cos\phi\dot{\phi} \\
\dot{R}_{33} &= -\sin\theta\cos\phi\dot{\theta} - \cos\theta\sin\phi\dot{\phi}
\end{aligned} \tag{2.25}$$

Para finalizar a relação entre a embarcação e a base do guindaste, deriva-se novamente esta equação, para obter a aceleração no ponto. Tendo assim:

$$\begin{bmatrix} \ddot{x}_{base} \\ \ddot{y}_{base} \\ \ddot{z}_{base} \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} + \ddot{R} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.26}$$

em que $\ddot{R}$ é a segunda derivada temporal da matriz R .

$$\ddot{R} = \begin{bmatrix} \ddot{R_{11}} & \ddot{R_{12}} & \ddot{R_{13}} \\ \ddot{R_{21}} & \ddot{R_{22}} & \ddot{R_{23}} \\ \ddot{R_{31}} & \ddot{R_{32}} & \ddot{R_{33}} \end{bmatrix} \tag{2.27}$$

$$
\begin{aligned}
\ddot{R}_{11} &= -\cos\psi\cos\theta\dot{\psi}^2 + \sin\psi\sin\theta\dot{\psi}\dot{\theta} \\
&\quad - \sin\psi\cos\theta\ddot{\psi} + \sin\psi\sin\theta\dot{\psi}\dot{\theta} \\
&\quad - \cos\psi\cos\theta\dot{\theta}^2 - \cos\psi\sin\theta\ddot{\theta} \\
\ddot{R}_{12} &= -\cos\psi\sin\theta\sin\phi\dot{\psi}^2 - \sin\psi\cos\theta\sin\phi\dot{\psi}\dot{\theta} \\
&\quad - \sin\psi\sin\theta\cos\phi\dot{\psi}\dot{\phi} - \sin\psi\sin\theta\sin\phi\ddot{\psi} \\
&\quad - \sin\psi\cos\theta\sin\phi\dot{\psi}\dot{\theta} - \cos\psi\sin\theta\sin\phi\dot{\theta}^2 \\
&\quad + \cos\psi\cos\theta\sin\phi\ddot{\theta} - \sin\psi\sin\theta\cos\phi\dot{\psi}\dot{\phi} \\
&\quad + 2\cos\psi\cos\theta\cos\phi\dot{\theta}\dot{\phi} - \cos\psi\sin\theta\sin\phi\dot{\phi}^2 \\
&\quad + \cos\psi\sin\theta\cos\phi\ddot{\phi} + \sin\psi\cos\phi\dot{\psi}^2 \\
&\quad + \cos\psi\sin\phi\dot{\psi}\dot{\phi} - \cos\psi\cos\phi\ddot{\psi} \\
&\quad + \cos\psi\sin\phi\dot{\psi}\dot{\phi} + \sin\psi\cos\phi\dot{\phi}^2 \\
&\quad + \sin\psi\sin\phi\ddot{\phi} \\
\ddot{R}_{13} &= -\cos\psi\sin\theta\cos\phi\dot{\psi}^2 \\
&\quad - \sin\psi\cos\theta\cos\phi\dot{\psi}\dot{\theta} + \sin\psi\sin\theta\sin\phi\dot{\psi}\dot{\phi} \\
&\quad - \sin\psi\sin\theta\cos\phi\ddot{\psi} - \sin\psi\cos\theta\cos\phi\dot{\psi}\dot{\theta} \\
&\quad - \cos\psi\sin\theta\cos\phi\dot{\theta}^2 - \cos\psi\cos\theta\sin\phi\dot{\theta}\dot{\phi} \\
&\quad + \cos\psi\cos\theta\cos\phi\ddot{\theta} + \sin\psi\sin\theta\sin\phi\dot{\psi}\dot{\phi} \\
&\quad - \cos\psi\cos\theta\sin\phi\dot{\theta}\dot{\phi} - \cos\psi\sin\theta\cos\phi\dot{\phi}^2 \\
&\quad - \cos\psi\sin\theta\sin\phi\ddot{\phi} - \sin\psi\sin\phi\dot{\psi}^2 \\
&\quad + 2\cos\psi\cos\phi\dot{\psi}\dot{\phi} + \cos\psi\sin\phi\ddot{\psi} \\
&\quad - \sin\psi\sin\phi\dot{\phi}^2 + \sin\psi\cos\phi\ddot{\phi} \\
\ddot{R}_{21} &= -\sin\psi\cos\theta\dot{\psi}^2 - \cos\psi\sin\theta\dot{\psi}\dot{\theta} \\
&\quad + \cos\psi\cos\theta\ddot{\psi} - \cos\psi\sin\theta\dot{\psi}\dot{\theta} \\
&\quad - \sin\psi\cos\theta\dot{\theta}^2 - \sin\psi\sin\theta\ddot{\theta}
\end{aligned}
\tag{2.28}
$$

$$
\begin{aligned}
\ddot{R}_{22} = & -\sin\psi\sin\theta\sin\phi\dot{\psi}^2 + \cos\psi\cos\theta\sin\phi\dot{\psi}\dot{\theta} \\
& + \cos\psi\sin\theta\cos\phi\dot{\psi}\dot{\phi} + \cos\psi\sin\theta\sin\phi\ddot{\psi} \\
& + \cos\psi\cos\theta\sin\phi\dot{\psi}\dot{\theta} - \sin\psi\sin\theta\sin\phi\dot{\theta}^2 \\
& + \sin\psi\cos\theta\cos\phi\dot{\theta}\dot{\phi} + \sin\psi\cos\theta\sin\phi\ddot{\theta} \\
& + \cos\psi\sin\theta\cos\phi\dot{\psi}\dot{\phi} + \sin\psi\cos\theta\cos\phi\dot{\theta}\dot{\phi} \\
& - \sin\psi\sin\theta\sin\phi\dot{\phi}^2 + \sin\psi\sin\theta\cos\phi\ddot{\phi} \\
& - \cos\psi\cos\phi\dot{\psi}^2 + \sin\psi\sin\phi\dot{\psi}\dot{\phi} \\
& - \sin\psi\cos\phi\ddot{\psi} + \sin\psi\sin\phi\dot{\psi}\dot{\phi} \\
& - \cos\psi\cos\phi\dot{\phi}^2 - \cos\psi\sin\phi\ddot{\phi} \\
\ddot{R}_{23} = & -\sin\psi\sin\theta\cos\phi\dot{\psi}^2 + \cos\psi\cos\theta\cos\phi\dot{\psi}\dot{\theta} \\
& - \cos\psi\sin\theta\sin\phi\dot{\psi}\dot{\phi} + \cos\psi\sin\theta\cos\phi\ddot{\psi} \\
& + \cos\psi\cos\theta\cos\phi\dot{\psi}\dot{\theta} - \sin\psi\sin\theta\cos\phi\dot{\theta}^2 \\
& - \sin\psi\cos\theta\sin\phi\dot{\theta}\dot{\phi} + \sin\psi\cos\theta\cos\phi\ddot{\theta} \\
& - \cos\psi\sin\theta\sin\phi\dot{\psi}\dot{\phi} - \sin\psi\cos\theta\sin\phi\dot{\theta}\dot{\phi} \\
& - \sin\psi\sin\theta\cos\phi\dot{\phi}^2 - \sin\psi\sin\theta\sin\phi\ddot{\phi} \\
& + \cos\psi\sin\phi\dot{\psi}^2 + \sin\psi\cos\phi\dot{\psi}\dot{\phi} \\
& + \sin\psi\sin\phi\ddot{\psi} + \sin\psi\cos\phi\dot{\psi}\dot{\phi} \\
& + \cos\psi\sin\phi\dot{\phi}^2 - \cos\psi\cos\phi\ddot{\phi} \\
\ddot{R}_{31} = & \sin\theta\dot{\theta}^2 - \cos\theta\ddot{\theta} \\
\ddot{R}_{32} = & -\cos\theta\sin\phi\dot{\theta}^2 - 2\sin\theta\cos\phi\dot{\theta}\dot{\phi} \\
& - \sin\theta\sin\phi\ddot{\theta} - \cos\theta\sin\phi\dot{\phi}^2 \\
& + \cos\theta\cos\phi\ddot{\phi} \\
\ddot{R}_{33} = & -\cos\theta\cos\phi\dot{\theta}^2 + 2\sin\theta\sin\phi\dot{\theta}\dot{\phi} \\
& - \sin\theta\cos\phi\ddot{\theta} - \cos\theta\cos\phi\dot{\phi}^2 \\
& - \cos\theta\sin\phi\ddot{\phi}
\end{aligned}
$$

Em seguida, para obter os dados para posição superior da lança do guindaste lançamos a hipótese de que a lança comporta-se como um corpo rígido, tendo assim a posição da parte superior expressa por:

$$\begin{aligned} x_{sup} =& x_{base} + L\cos(\alpha+\theta)\cos(\gamma+\psi) \\ y_{sup} =& y_{base} + L\cos(\alpha+\phi)\sin(\gamma+\psi) \\ z_{sup} =& z_{base} + L\sin(\alpha+\phi+\theta) \end{aligned} \tag{2.29}$$

Para obter a velocidade derivamos as equações acima no tempo, resultando:

$$\begin{aligned} \dot{x}_{sup} =& \dot{x}_{base} - L\sin(\alpha+\theta)\cos(\gamma+\psi)(\dot{\alpha}+\dot{\theta}) - L\cos(\alpha+\theta)\sin(\gamma+\psi)(\dot{\gamma}+\dot{\psi}) \\ \dot{y}_{sup} =& \dot{y}_{base} - L\sin(\alpha+\phi)\sin(\gamma+\psi)(\dot{\alpha}+\dot{\phi}) + L\cos(\alpha+\phi)\cos(\gamma+\psi)(\dot{\gamma}+\dot{\psi}) \\ \dot{z}_{sup} =& \dot{z}_{base} + L\cos(\alpha+\phi+\theta)(\dot{\alpha}+\dot{\phi}+\dot{\theta}) \end{aligned} \tag{2.30}$$

De maneira semelhante para a aceleração, deriva-se novamente com relação ao tempo.

$$\begin{aligned} \ddot{x}_{sup} = \ddot{x}_{base} &- L\sin(\alpha+\theta)\cos(\gamma+\psi)(\ddot{\alpha}+\ddot{\theta}) \\ &- L\cos(\alpha+\theta)\cos(\gamma+\psi)(\dot{\alpha}+\dot{\theta})^2 \\ &+ 2L\sin(\alpha+\theta)\sin(\gamma+\psi)(\dot{\alpha}+\dot{\phi})(\dot{\gamma}+\dot{\psi}) \\ &- L\cos(\alpha+\phi)\cos(\gamma+\psi)(\dot{\gamma}+\dot{\psi})^2 \\ &- L\cos(\alpha+\phi)\sin(\gamma+\psi)(\ddot{\gamma}+\ddot{\psi}) \end{aligned} \tag{2.31}$$

$$\begin{aligned} \ddot{y}_{sup} = \ddot{y}_{base} &- L\sin(\alpha+\phi)\sin(\gamma+\psi)(\ddot{\alpha}+\ddot{\phi}) \\ &- L\cos(\alpha+\phi)\sin(\gamma+\psi)(\dot{\alpha}+\dot{\theta})^2 \\ &- 2L\sin(\alpha+\theta)\cos(\gamma+\psi)(\dot{\alpha}+\dot{\phi})(\dot{\gamma}+\dot{\psi}) \\ &- L\cos(\alpha+\phi)\sin(\gamma+\psi)(\dot{\gamma}+\dot{\psi})^2 \\ &+ L\cos(\alpha+\phi)\cos(\gamma+\psi)(\ddot{\gamma}+\ddot{\psi}) \end{aligned} \tag{2.32}$$

$$\begin{aligned} \ddot{z}_{sup} = \ddot{z}_{base} &+ L\cos(\alpha+\phi+\theta)(\ddot{\alpha}+\ddot{\phi}+\ddot{\theta}) \\ &- L\sin(\alpha+\phi+\theta)(\dot{\alpha}+\dot{\phi}+\dot{\theta})^2 \end{aligned} \tag{2.33}$$

Agora que temos todas as variáveis para a parte superior da lança, iremos obter as variáveis da carga, sendo a posição expressa por:

$$\begin{aligned} x_{carga} &= x_{sup} + l\sin\beta_1\cos\beta_2 \\ y_{carga} &= y_{sup} + l\cos\beta_1\sin\beta_2 \\ z_{carga} &= z_{sup} - l\cos\beta_1\cos\beta_2 \end{aligned} \tag{2.34}$$

Ao derivar com relação ao tempo, obtém-se a velocidade da carga:

$$\begin{aligned}
\dot{x}_{carga} =& \dot{x}_{sup} + l\cos(\beta_1)\cos(\beta_2)\dot{\beta}_1 - l\sin(\beta_1)\sin(\beta_2)\dot{\beta}_2 + \dot{l}\sin(\beta_1)\cos(\beta_2) \\
\dot{y}_{carga} =& \dot{y}_{sup} - l\sin(\beta_1)\sin(\beta_2)\dot{\beta}_1 + l\cos(\beta_1)\cos(\beta_2)\dot{\beta}_2 + \dot{l}\cos(\beta_1)\sin(\beta_2) \\
\dot{z}_{carga} =& \dot{z}_{sup} + l\sin(\beta_1)\cos(\beta_2)\dot{\beta}_1 + l\cos(\beta_1)\sin(\beta_2)\dot{\beta}_2 - \dot{l}\cos(\beta_1)\cos(\beta_2)
\end{aligned} \tag{2.35}$$

Com essas equações pode-se calcular alguns parâmetros da carga, como sua velocidade total:

$$V_{carga} = \sqrt{\dot{x}_{carga}^2 + \dot{y}_{carga}^2 + \dot{z}_{carga}^2} \tag{2.36}$$

E a tração no cabo:

$$T = m\left[\frac{V_{carga}^2}{l} + g\cos(\beta_1)\cos(\beta_2)\right] \tag{2.37}$$

Para finalizar a modelagem estática, expressaremos uma variável externa ao corpo, a força do vento sob a carga, que será utilizada na modelagem dinâmica do sistema. Nesse modelo, a força do vento pode ser calculada por [14]:

$$\begin{aligned}
F_{vx} &= \rho C_s A_x U_{mx}^2 \\
F_{vy} &= \rho C_s A_y U_{my}^2 \\
F_{vz} &= \rho C_s A_z U_{mz}^2
\end{aligned} \tag{2.38}$$

sendo

- F_{vx} = força do vento em x
- F_{vy} = força do vento em y
- F_{vz} = força do vento em z
- ρ = densidade do ar
- C_s = coeficiente de forma
- A_x = área no plano yz
- A_y = área no plano xz

- A_z = área no plano xy
- U_{mx} = velocidade média do vento em x
- U_{my} = velocidade média do vento em y
- U_{mz} = velocidade média do vento em z

Deve-se lembrar que o valor do fator de forma pode ser utilizado no valor de 1.2 para números de Reynolds menores que 5.10^5 [15]. Posteriormente, deve-se obter a dinâmica do sistema, por meio das equações de Lagrange, de modo a possibilitar o cálculo dos ângulos da carga. Inicialmente calcula-se as energias cinéticas e potenciais por:

$$\begin{aligned} T =& T_{carga} + T_{lança} = \frac{1}{2} m V_{carga}^2 + \frac{1}{6} M L^2 (\dot{\alpha}^2 + \dot{\gamma}^2) \\ U =& U_{carga} + U_{lança} = m g z_{sup} - m g l \cos(\beta_1) \cos(\beta_2) + \frac{1}{2} M g z_{sup} \end{aligned} \tag{2.39}$$

Assim, temos como a função Lagrangiana:

$$L = T - U = \frac{1}{2} m V_{carga}^2 + \frac{1}{6} M L^2 (\dot{\alpha}^2 + \dot{\gamma}^2) - m g z_{sup} + m g l \cos(\beta_1) \cos(\beta_2) - \frac{1}{2} M g z_{sup} \tag{2.40}$$

Antes de inserir a função Lagrangiana nas equações de Lagrange, iremos definir as expressões das forças não conservativas:

$$\begin{aligned} Q_{\beta 1} =& F_{vx} \frac{\delta x}{\delta \beta_1} + F_{vy} \frac{\delta y}{\delta \beta_1} + F_{vz} \frac{\delta z}{\delta \beta_1} \\ =& F_{vx} l \cos(\beta_1) \cos(\beta_2) - F_{vy} l \sin(\beta_1) \sin(\beta_2) + F_{vz} l \sin(\beta_1) \cos(\beta_2) \\ Q_{\beta 2} =& F_{vx} \frac{\delta x}{\delta \beta_2} + F_{vy} \frac{\delta y}{\delta \beta_2} + F_{vz} \frac{\delta z}{\delta \beta_2} \\ =& - F_{vx} l \sin(\beta_1) \sin(\beta_2) + F_{vy} l \cos(\beta_1) \cos(\beta_2) + F_{vz} l \cos(\beta_1) \sin(\beta_2) \end{aligned} \tag{2.41}$$

Outro passo é lembrar que na equação final, os termos em função dos ângulos da carga serão linearizados para o caso de ângulos pequenos, ou seja, dado θ, um ângulo pequeno, assim podemos aproximar

$$\begin{aligned} \sin(\theta) =& \theta \\ \cos(\theta) =& 1 \end{aligned} \tag{2.42}$$

Substituindo assim a função Lagrangiana L nas equações de Lagrange dos fundamentos teóricos, e realizando as derivadas parciais, além de aplicar algumas propriedades trigonométricas clássicas, obtemos as equações dinâmicas do guindaste.

$$\begin{aligned}
&\frac{m}{2}\{2\ddot{x}_{sup}l\cos(\beta_1)\cos(\beta_2) - 2\ddot{y}_{sup}l\sin(\beta_1)\sin(\beta_2) + 2\ddot{z}_{sup}l\sin(\beta_1)\cos(\beta_2)\\
&\quad + l\dot{l}\dot{\beta}_1(3 - \cos(2\beta_1) + \cos(2\beta_2) + \cos(2\beta_1)\cos(2\beta_2)) + l\dot{l}\dot{\beta}_2\sin(2\beta_1)\sin(2\beta_2)\\
&\quad + l^2[\frac{\dot{\beta}_1^2}{2}(\sin(2\beta_1) - \sin(2\beta_1)\cos(2\beta_2))\\
&\quad + \frac{\dot{\beta}_2^2}{2}(\sin(2\beta_1) - \sin(2\beta_1)\cos(2\beta_2))\\
&\quad - \dot{\beta}_1\dot{\beta}_2(\sin(2\beta_2) + \cos(2\beta_1)\sin(2\beta_2))\\
&\quad + \frac{\ddot{\beta}_1}{2}(3 - \cos(2\beta_1) + \cos(2\beta_2) + \cos(2\beta_1)\cos(2\beta_2))\\
&\quad - \frac{\ddot{\beta}_2}{2}\sin(2\beta_1)\cos(2\beta_2)] + \dot{l}^2(\sin(2\beta_1) - \sin(2\beta_1)\cos(2\beta_2))\\
&\quad + \frac{l\dot{l}}{2}(\sin(2\beta_1) - \sin(2\beta_1)\cos(2\beta_2)) + 2gl\sin(\beta_1)\cos(\beta_2)\}\\
&\quad = F_{vx}l\cos(\beta_1)\cos(\beta_2) - F_{vy}l\sin(\beta_1)\sin(\beta_2) + F_{vz}l\sin(\beta_1)\cos(\beta_2)
\end{aligned} \tag{2.43}$$

$$\begin{aligned}
&\frac{m}{2}\{-2\ddot{x}_{sup}l\sin(\beta_1)\sin(\beta_2) + 2\ddot{y}_{sup}l\cos(\beta_1)\cos(\beta_2) + 2\ddot{z}_{sup}l\cos(\beta_1)\sin(\beta_2)\\
&\quad + l\dot{l}\dot{\beta}_2(3 + \cos(2\beta_1) - \cos(2\beta_2) + \cos(2\beta_1)\cos(2\beta_2)) - 3l\dot{l}\dot{\beta}_2\sin(2\beta_1)\sin(2\beta_2)\\
&\quad + l^2[\frac{\dot{\beta}_1^2}{2}(\sin(2\beta_1) - \cos(2\beta_1)\sin(2\beta_2))\\
&\quad + \frac{\beta_2^2}{2}(\sin(2\beta_1) - \cos(2\beta_1)\sin(2\beta_2))\\
&\quad - \dot{\beta}_1\dot{\beta}_2(\sin(2\beta_2) + \sin(2\beta_1)\cos(2\beta_2))\\
&\quad + \frac{\ddot{\beta}_2}{2}(3 + \cos(2\beta_1) - \cos(2\beta_2) + \cos(2\beta_1)\cos(2\beta_2))\\
&\quad - \frac{\ddot{\beta}_1}{2}\sin(2\beta_1)\sin(2\beta_2)] - \frac{l\ddot{l}}{2}(\sin(2\beta_2) - \cos(2\beta_1)\sin(2\beta_2))\\
&\quad + 2gl\cos(\beta_1)\sin(\beta_2)\}\\
&\quad = -F_{vx}l\sin(\beta_1)\sin(\beta_2) + F_{vy}l\cos(\beta_1)\cos(\beta_2) + F_{vz}l\cos(\beta_1)\sin(\beta_2)
\end{aligned} \tag{2.44}$$

Em seguida aplica-se o processo de linearização de pequenos ângulos para as variáveis β_1 e β_2, obtemos:

$$\begin{aligned} m(\ddot{x}_{sup} + \ddot{z}_{sup}\beta_1 + 2\dot{l}\dot{\beta}_1 - 2l\dot{\beta}_1\dot{\beta}_2\beta_2 + l\ddot{\beta}_1 + g\beta_1) &= F_{vx} + F_{vz}\beta_1 \\ m(\ddot{y}_{sup} + \ddot{z}_{sup}\beta_2 + 2\dot{l}\dot{\beta}_2 - 2l\dot{\beta}_2\dot{\beta}_1\beta_1 + l\ddot{\beta}_2 + g\beta_2) &= F_{vy} + F_{vz}\beta_2 \end{aligned} \tag{2.45}$$

Ao manipular as equações acima, podemos obter as equações dinâmicas finais do guindaste:

$$\begin{aligned} \ddot{\beta}_1 =& (2\dot{\beta}_2\beta_2 - \frac{2\dot{l}}{l})\dot{\beta}_1 + (\frac{F_{vz}}{ml} - \frac{\ddot{z}_{sup}}{l} - \frac{g}{l})\beta_1 + \frac{F_{vx}}{ml} - \frac{\ddot{x}_{sup}}{l} \\ \ddot{\beta}_2 =& (2\dot{\beta}_1\beta_1 - \frac{2\dot{l}}{l})\dot{\beta}_2 + (\frac{F_{vz}}{ml} - \frac{\ddot{z}_{sup}}{l} - \frac{g}{l})\beta_2 + \frac{F_{vy}}{ml} - \frac{\ddot{y}_{sup}}{l} \end{aligned} \tag{2.46}$$

PARTE II

APLICAÇÃO

3 SIMULAÇÃO DO MODELO DINÂMICO

A simulação do modelo dinâmico será realizada em paralelo com uma validação desse modelo em comparação com um modelo computacional de guindaste 2D [16] , ou seja, iremos simular três casos no software e no programa deste projeto (Anexo A), e compará-los, de modo a demonstrar o comportamento do guindaste e validar o programa feito neste trabalho. As simulações vão consistir de três casos, sendo eles: movimento periódico senoidal no eixo x (Sway), movimento periódico senoidal de rotação em torno do eixo x (roll) e um movimento periódico senoidal no eixo x com período próximo a ressonância do guindaste. Além disso, iremos simular dois programas, sendo eles o modelo dinâmico completo do sistema e o modelo linearizado, para analisar a performance de cada com relação à precisão de cada um deles. De modo a ilustrar os movimentos citados acima, vide figura abaixo.

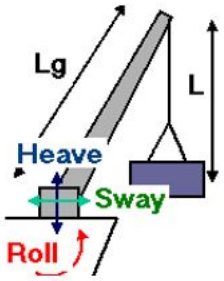


Figura 10: Modelo Guindaste 2D

3.1 Movimento de Sway

Para validar o modelo com movimento Sway iremos excitar o sistema com uma entrada de 1 metro e um período de 10 segundos. Adquirindo assim, os seguintes resultados para o software do TPN e para o programa do projeto, respectivamente:

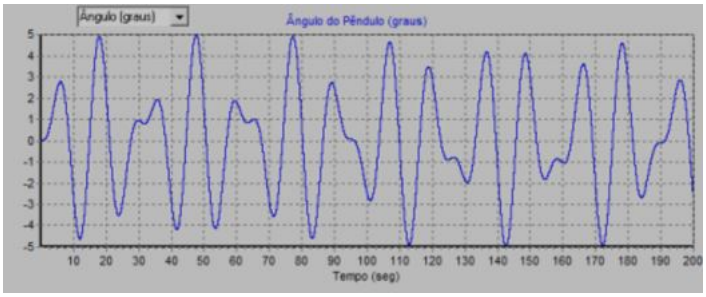


Figura 11: Ângulo da carga na validação do Sway

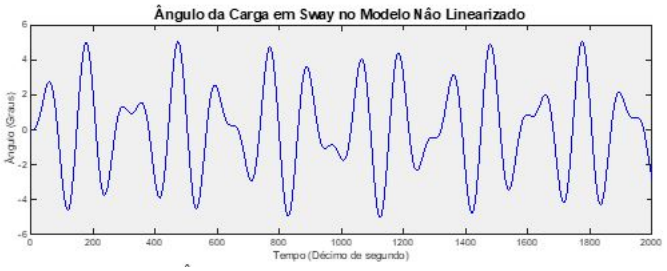


Figura 12: Ângulo da carga no modelo não linearizado do Sway

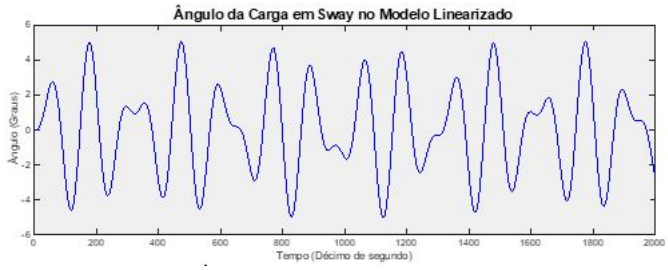


Figura 13: Ângulo da carga no modelo linearizado do Sway

Pode-se obter um erro do modelo com relação ao software, pelos valores extremos de cada simulação, sendo assim o erro deste teste: 1.20% para o modelo não linearizado e 0.96% para o modelo linearizado.

Outro parâmetro que pode ser comparado, além do ângulo da carga, visto acima, é a tração no cabo, sendo ela:

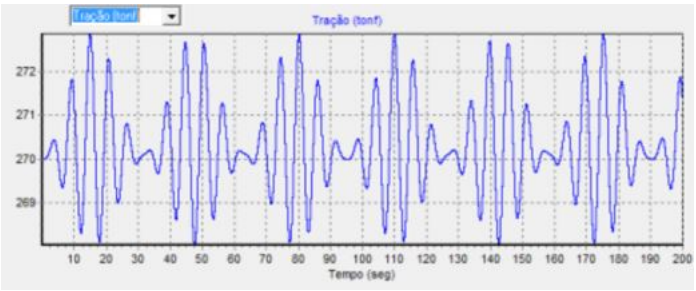


Figura 14: Tração no Cabo na validação do Sway

Figura 15: Tração no Cabo no modelo não linearizado do Sway

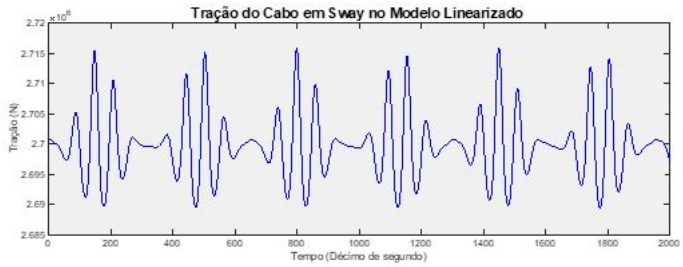


Figura 16: Tração no Cabo no modelo linearizado do Sway

Tendo assim, um erro neste quesito, de: -0.48% para ambos os modelos.

3.2 Movimento de Roll

Para validar o modelo com movimento Roll iremos excitar o sistema com uma entrada de 2 graus e um período de 11 segundos. Tendo assim, os seguintes resultados para o software do TPN e para o programa do projeto, respectivamente:

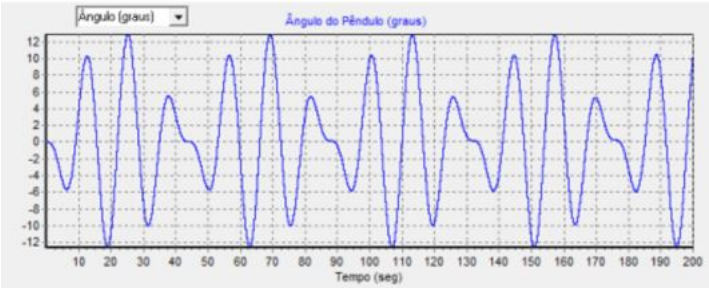


Figura 17: Ângulo da carga na validação do Roll

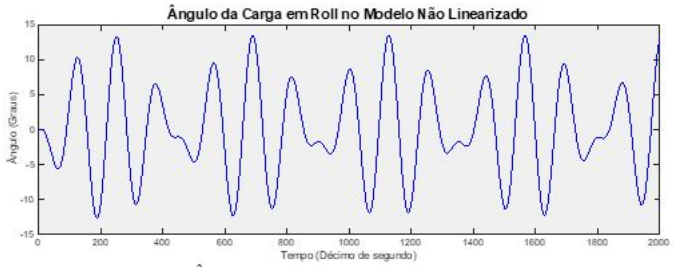


Figura 18: Ângulo da carga no modelo não linearizado do Roll

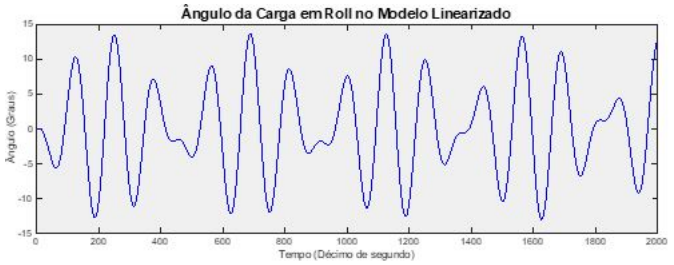


Figura 19: Ângulo da carga no modelo linearizado do Roll

Pode-se obter um erro do modelo com relação ao software, pelos valores extremos de cada simulação, sendo assim o erro deste teste: 4.97% para o modelos não linearizado e 5.83% para o modelo linearizado.

Outro parâmetro que pode ser comparado, além do ângulo da carga, visto acima, é a tração no cabo, sendo ela:

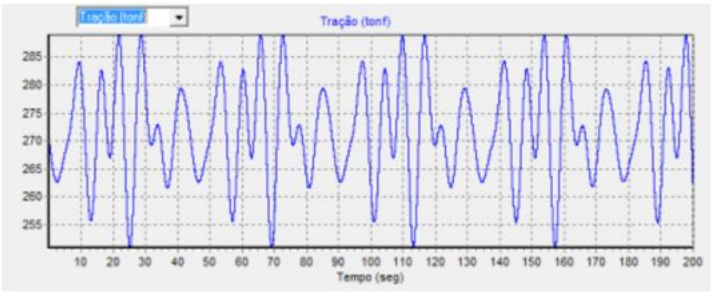


Figura 20: Tração no cabo na validação do Roll

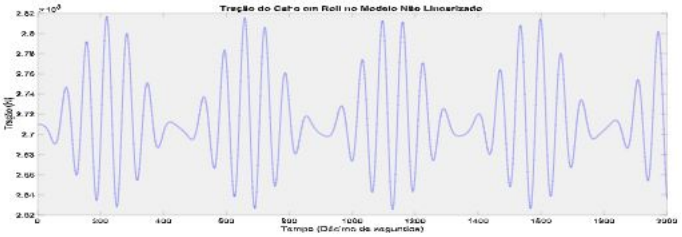


Figura 21: Tração no cabo no modelo não linearizado do Roll

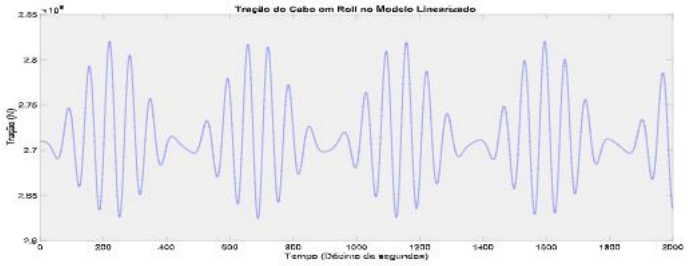


Figura 22: Tração no cabo no modelo linearizado do Roll

Tendo assim, um erro neste quesito, de: 2,51% para o modelo não linearizado e 2,37% para o modelo linearizado.

3.3 Movimento Sway na Ressonância

Para validar o modelo com movimento Sway na ressonância iremos excitar o sistema com uma entrada de 5 metros e um período de 14 segundos. Tendo assim, os seguintes resultados para o software do TPN e para o programa do projeto, respectivamente:

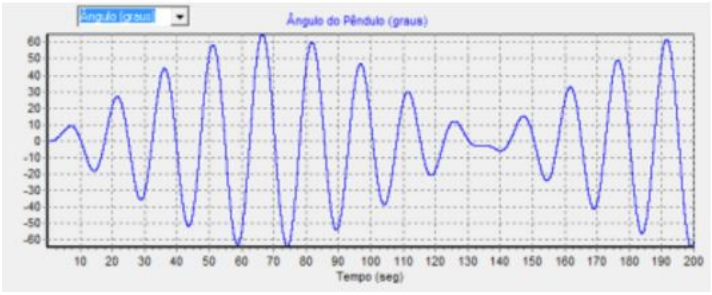


Figura 23: Ângulo da carga na validação do Sway na ressonância

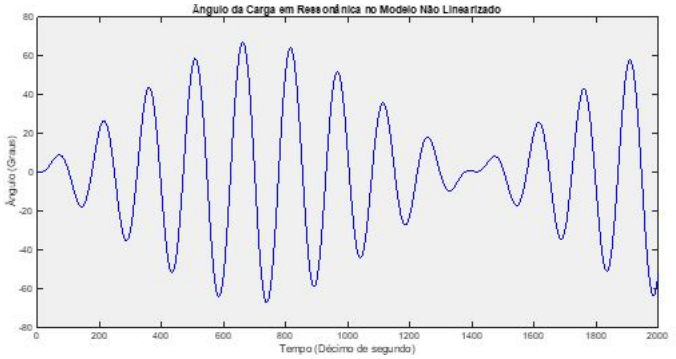


Figura 24: Ângulo da carga no modelo não linearizado do Sway na ressonância

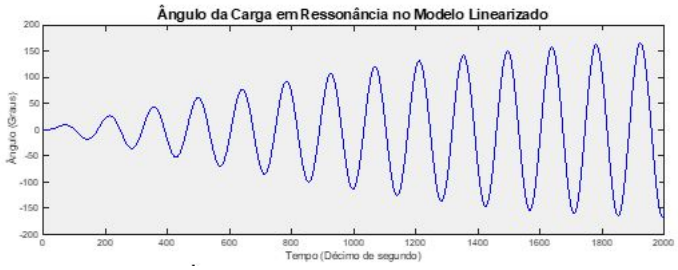


Figura 25: Ângulo da carga no modelo linearizado do Sway na ressonância

Pode-se obter um erro do modelo com relação ao software, pelos valores extremos de cada simulação, sendo assim o erro deste teste: 3.25% para o modelo não linearizado e 154.54% para o modelo linearizado.

Outro parâmetro que pode ser comparado, além do ângulo da carga, visto acima, é a tração no cabo, sendo ela:

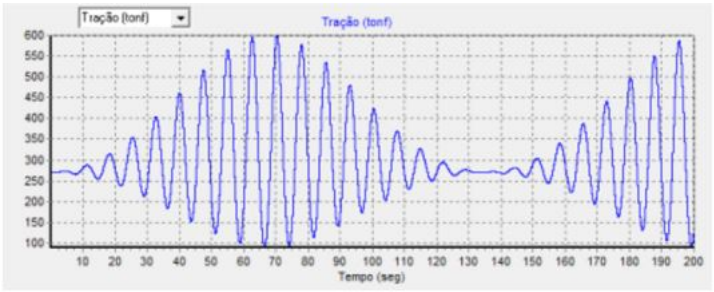


Figura 26: Tração no cabo na validação do Sway na ressonância

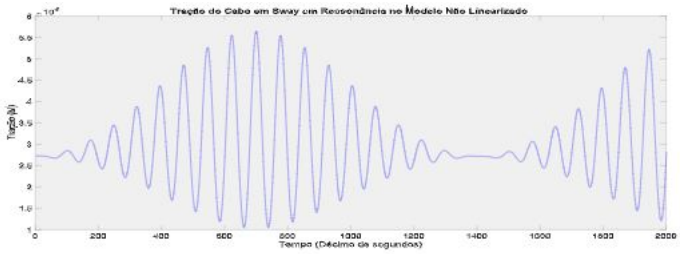


Figura 27: Tração no cabo no modelo não linearizado do Sway na ressonância

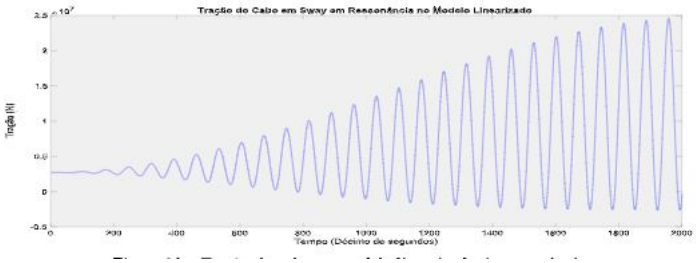


Figura 28: Tração no cabo no modelo linearizado do Sway na ressonância

Tendo assim, neste quesito, um erro de: 3,94% para o modelo não linearizado e 317,89% para o modelo linearizado.

Quando comparamos o nosso modelo com o descrito em [16] obtivemos pequenos desvios, os erros mostraram-se bastante irrisórios. Mesmo pequenas, houveram diferenças, o que poderia revelar uma disparidade entre os modelos, todavia ao olharmos mais atentamente o modelo de comparação vemos que este leva em conta aspectos que foram desconsiderados neste trabalho, como por exemplo a dinâmica elástica do cabo do guindaste. Logo podemos afirmar que os modelos concordam quanto a dinâmica do guindaste.

4 CONTROLE

4.1 Design do Controlador

Conforme solicitado por integrantes do TPN, o controle que será projetado será um Compensador de Heave para a posição vertical da carga. Desse modo, iremos testar 4 tipos de controladores: Proporcional (P), Proporcional Derivativo (PD), Proporcional Integral (PI) e Proporcional Integral Derivativo (PID). Para este projeto, iremos considerar três aspectos: a planta do sistema a ser controlada é dada pela equação apresentada na seção de controle de posição, sendo esta igual a $\frac{1}{s}$, como mostra o diagrama a seguir.

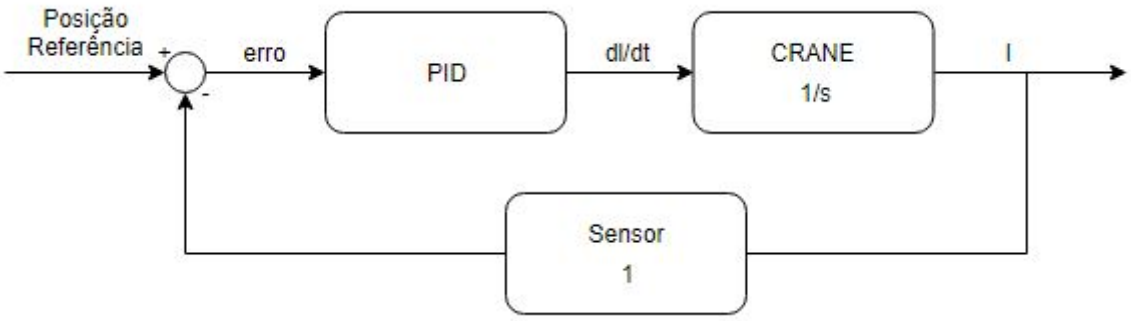


Figura 29: Diagrama de Blocos do Sistema

O tempo de resposta do sistema deve ser de 1 segundo; e o sistema deve apresentar a maior robustez possível. Com a utilização do toolbox de PID tuner do MATLAB, que utiliza dos métodos de controle citados na seção do PID no embasamento teórico, obtemos os seguintes resultados:

- Controlador P: $K_p = 2$, e sobressinal nulo
- Controlador PD: $K_p = 2$, $K_d = 0$, e sobressinal nulo.
- Controlador PI: $K_p = 2$, $K_i = 0.07$, e sobressinal nulo
- Controlador PID: $K_p = 2$, $K_i = 0.4084$, $K_d = 0.1021$, e sobressinal de 7%

Como o controlador PD possui o termo derivativo nulo este será desconsiderado,podendo-se observar então que há três possibilidades de controladores: P, PI e PID. Como o controlador PID neste caso apresenta sobressinal, que não é algo desejado para o sistema, iremos descartá-lo. Tendo assim as opções de um controlador P e PI com comportamento quase idênticos, portanto o controlador que será aplicado como compensador de heave será um controlador Proporcional com ganho igual a 2, devido a simplicidade ser maior do que o proporcional integral.

4.2 Resultados do Controlador

Testamos o controlador face a uma referência de um sinal de entrada em degrau, comparando o resultado obtido com o caso na ausência de controlador.

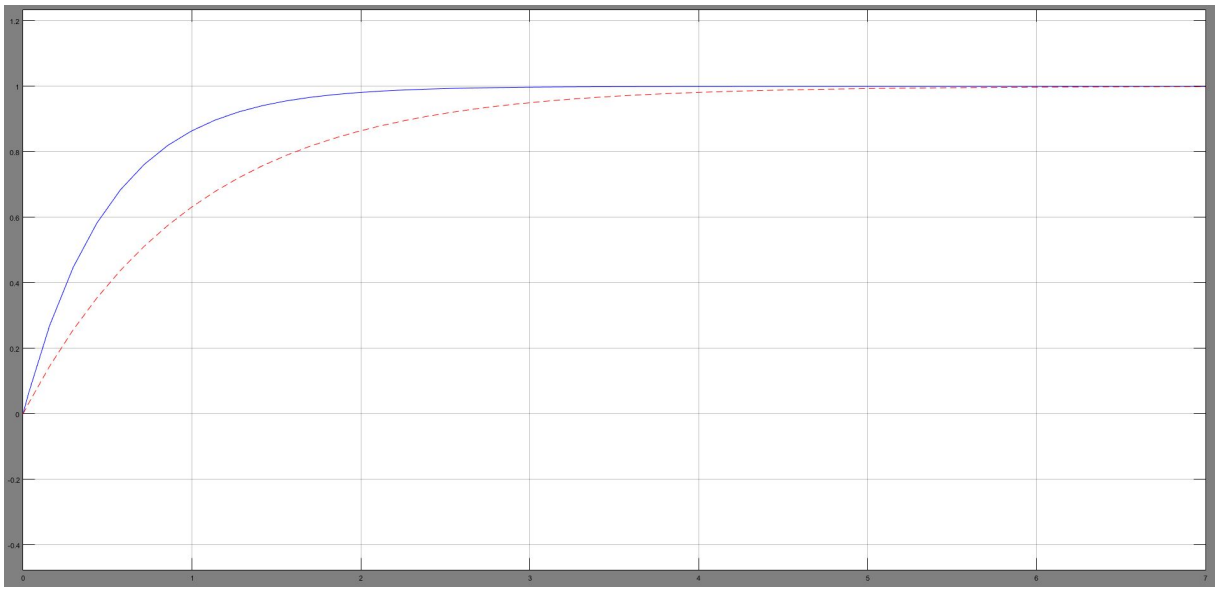


Figura 30: Comparação entre o modelo com e sem controle

Como pode-se observar, o modelo sem ação do controle (linha rosa) apresenta um tempo de resposta de cerca de 2 segundos (valor para atingir 86,5% do valor final sem sobressinal), enquanto isso, com o controlador (linha negra), temos que esse valor cai para 1 segundo, que é o desejado.

Todavia o controlador proposto parte do pressuposto que o modelo é linear, porém um efeito inerente de qualquer sistema que trabalha com motores é a saturação, uma faixa de atuação para a velocidade do motor, cujo efeito deve ser levado em conta em nossas simulações. Na imagem a seguir está ilustrado o diagrama de blocos usado na ferramenta SIMULINK do MATLAB aplicado ao sistema com a não linearidade descrita.

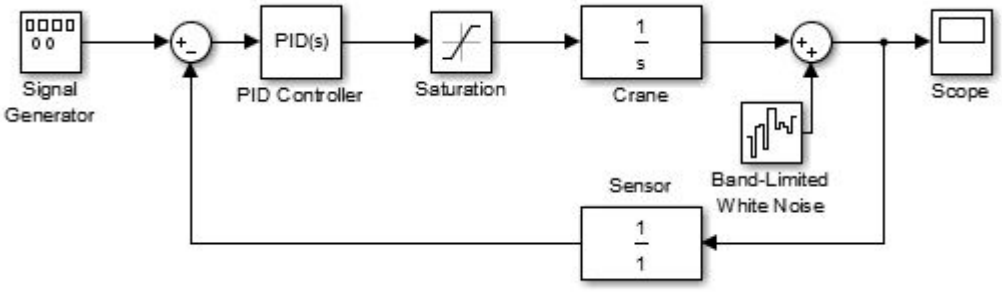


Figura 31: Diagrama Simulink do Sistema

Utilizando esta ferramenta, foram feitos testes para entender a performance do controlador face ao que esperamos de perturbações que o sistema pode enfrentar.

Foram feitos testes para uma entrada degrau de amplitude 4, diferentes perturbações foram simuladas para avaliar o funcionamento do controlador.

Iniciamos com um teste com perturbações com períodos de 2 segundos e amplitude 0.3, que representa uma oscilação de $30cm$ a cada segundo.

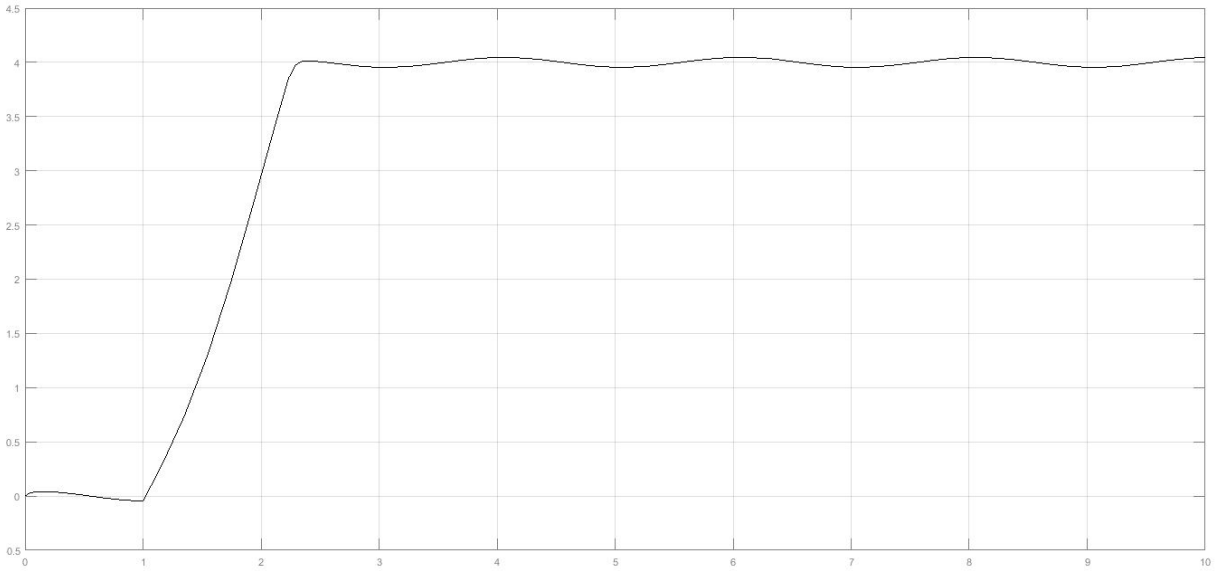


Figura 32: Resultados da Simulação para oscilações de 30cm a cada 2 segundos

Em seguida aumentamos intensidade das oscilações com objetivo de perceber as limitações do sistema. Para tanto aplicamos oscilações de $1m$ por segundo.

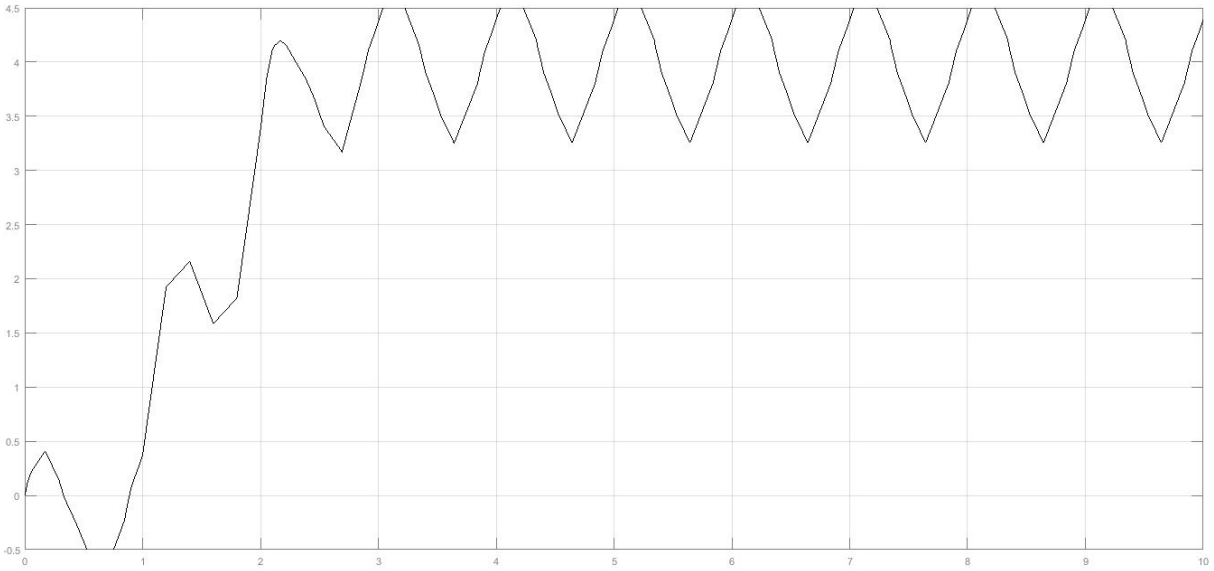


Figura 33: Resultados da Simulação para oscilações de 1m a cada segundo

Fica assim claro que para condições de operação reais o controlador funciona bem, como era esperado, o aumento da frequência das oscilações e da amplitude acaba por instabilizar o sistema. O que é evidente quando olhamos o diagrama de Bode da planta, para altas frequências o ganho é reduzido.

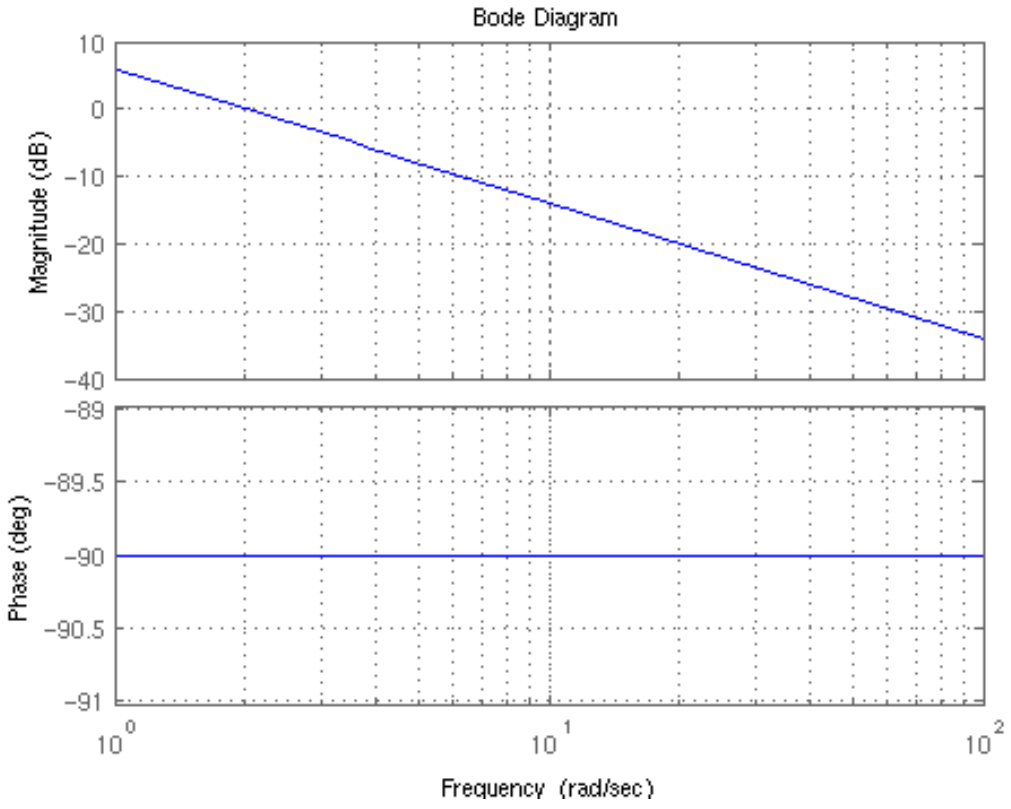


Figura 34: Diagrama de Bode do Sistema

5 IMPLEMENTAÇÃO NO SIMULADOR

A implementação do programa no simulador de guindastes do TPN é realizada em três etapas mais um teste final de validação. A etapas são: integração com o banco de dados, integração com os equipamentos do operador e integração visual.

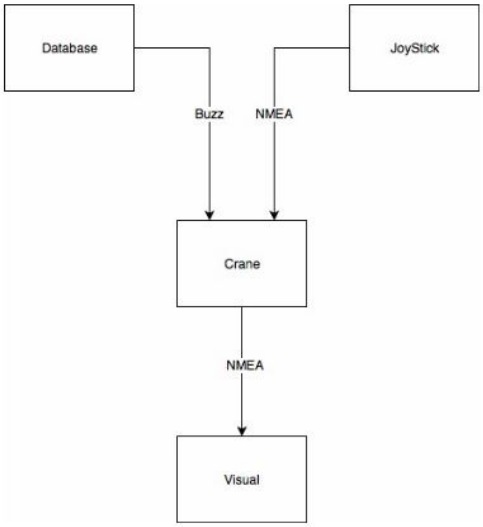


Figura 35: Arquitetura de Integração

5.1 Integração com o Banco de Dados

A integração do programa do guindaste com o banco de dados do simulador é feita via Buzz, um sistema de comunicação, que dá ao projeto o acesso aos dados utilizados durante a simulação, como informações da embarcação, das ondas, vento, e outros aspectos do ambiente. Essa integração é responsável por fornecer dados constantes à simulação, logo um ponto vital para o funcionamento do sistema. O programa funciona basicamente de forma à conectar a dinâmica do guindaste com o IP do banco de dados e com o IP do servidor do simulador, desse modo é possível extrair deste banco de dados, as informações que estão sendo utilizadas na simulação que está em execução no servidor, e assim executar o programa do guindaste do trabalho.

5.2 Integração com equipamentos do operador

A integração do programa do guindaste com os equipamentos do operador é feita via NMEA, que é um protocolo de comunicação via uma mensagem universal que contém todas as informações de uma determinada ação concatenadas. Assim, o programa filtra as informações que deseja saber, como movimentos dos joysticks, e extrai-as para o simulador, de forma a transformar essas informações em efeitos operacionais para o sistema.

5.3 Integração visual

A integração do programa do guindaste com a parte visual também é feita via NMEA, e consiste basicamente de enviar mensagem classificadas para o sistema de comunicação global, de modo a possibilitar que o programa do visual consiga extrair tais informações para atualizar seu estado presente.

A seguir apresentamos algumas imagens para ilustrar o simulador em operação. As imagens foram captadas na sala *Alphacrusis* do Simulador Marítimo Hidroviário do Tanque de Provas Numérico da Universidade de São Paulo, normalmente, nesta sala são feitas simulações de embarcações de apoio, por esta razão os cockpit diverge de um guindaste comum, porém do ponto de vista de implementação estes detalhes são irrelevantes.

Figura 36: Simulador em Operação - Vista Panorâmica

Figura 37: Arquitetura de Integração - Vista Cockpit

PARTE III

CONCLUSÃO

6 CONCLUSÃO E FUTUROS TRABALHOS

Ao final da modelagem dinâmica do guindaste, foi realizado um teste de confirmação do modelo em comparação com um software já existente, e como visto anteriormente, confirmou-se que o modelo obtido é fidedigno à realidade, reagindo com um comportamento semelhante aos dados fornecido por este software de apoio, sendo isto comprovado com base na obtenção de erros muito pequenos entre o modelo obtido e o modelo referência. Além desta modelagem, foi projetado também um controlador de heave que, caso ativado pelo operador, mantém a carga na altura presente, ou até mesmo em uma altura indicada pelo mesmo. Feito isto, o programa foi integrado ao simulador de guindastes com sucesso, o que possibilitou testes em campo com o mesmo, e viu-se um comportamento muito bom com relação a sua dinâmica visual, além de uma parte operacional coerente e robusta. Como trabalho futuro, seria conveniente dois ramos de desenvolvimento para o simulador de guindaste: por um lado desenvolver uma interface que possibilite a alteração de parâmetros do código fonte em tempo real, como por exemplo a aplicação de uma carga com massa variável; e por outro lado o projeto de outros controladores que possam agregar valor ao simulador, como um controlador de Anti-Sway. Outros projetos de desenvolvimento futuro, como uma interação da dinâmica do guindaste com elementos externos, como o mar, ou parte do navio, também seriam de extrema relevância para o projeto.

REFERÊNCIAS

[1] IMAGEM estruturas petroleiras. Acessado em 03-11-2018. Disponível em: <http://agenciabrasil.ebc.com.br/sites/default/files/atoms/image/plataforma_fpso_cidade_de_itag _foto_andre_motta-petrobras.jpg>.

[2] GRÁFICO produção e consumo de petróleo. Acessado em 03-11-2018. Disponível em: <https://www.nexojornal.com.br/grafico/2018/06/06/A-produção-o-consumo-e-o-preço-do-petróleo-no-mundo-ao-longo-dos-anos>.

[3] IMAGEM Limiar da Estabilidade. Acessado em 03-11-2018. Disponível em: <https://www.electronicshub.org/wp-content/uploads/2015/12/Sustained-oscillations.jpg>.

[4] KABIR, S. A human brachial artery system prototype controller calibration (static model). *International Journal of Emerging Sciences (IJES)*, v. 1, p. 706–722, 12 2011.

[5] AMORIM, T. Plataformas offshore uma breve análise desde a construção ao descomissionamento. 12 2010.

[6] ABDEL-RAHMAN, E. M.; NAYFEH, A. H.; MASOUD, Z. N. Dynamics and control of cranes: A review. *Modal Analysis*, v. 9, n. 7, p. 863–908, 2003. Disponível em: <https://doi.org/10.1177/1077546303009007007>.

[7] SIMULADOR SMH TPN. Acessado em 03-11-2018. Disponível em: <http://jornal.usp.br/wp-content/uploads/20171121_02_SimuladorPoli.jpg>.

[8] HARITOS, N. Introduction to the analysis and design of offshore structures- an overview. v. 7, p. 55–65, 01 2007.

[9] CARVALHO M. I. B., M. A. C. M. 2001.

[10] ESFANDIARI, R.; LU, B. *Modeling and Analysis of Dynamic Systems*. [S.l.]: Boca Raton: CRC Press, 2018.

[11] STEWART, J. *Cálculo*. Pioneira Thomson Learning, 2006. ISBN 9788522104840. Disponível em: <https://books.google.com.br/books?id=Q97BJwAACAAJ>.

[12] OGATA, K. *Modern Control Engineering*. 4th. ed. [S.l.]: Prentice Hall PTR, 2001. ISBN 0130609072.

[13] WALID, A. A. et al. Modeling and simulation of an active heave compensated drawworks. In: . [S.l.: s.n.].

[14] Liebherr-Werk Ehingen GmbH. As influências do vento na operação do guindaste. v. 4, 2017.

[15] NORSOK. Actions and action effects. n-003. *Norwegian Technology Standards Institution*, 1999.

[16] TANNURI, E. A. *Simulador de Movimento de Carga 2D*. 2000. Programa não registrado.

ANEXO A

Software de Validação

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                          Validation Program
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: September, 2018
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%% System Data %%%%%

L = 70;
m = 270000;
g = 10;
l = 53;
A = 1;
T = 10;

rho = 1;
C = 1.2;

Ax = 50;
Ay = 50;
Az = 50;
```

```
dx = 0;
dy = 0;
dz = 0;

beta_11 = 0;
beta_11_i = 0;
beta_11_ii = 0;
beta_22 = 0;
beta_22_i = 0;
beta_22_ii = 0;

alpha = 1.186823891;
phi = 0;

index_i_alpha_i = 0;
index_i_phi_i = 0;
index_i_l_i = 0;

%%%%%%%% Function for %%%%%%%%%%

for time = 0 : 0.1 : 200

%%%%%%%%%%%%%%%% Input %%%%%%%%%%

    x = A*sin(2*pi*time/T);
    y = 0;
    z = 0;
    x_i = (2*pi*A/T)*cos(2*pi*time/T);
    y_i = 0;
    z_i = 0;
    x_ii = -A*((2*pi/T)^2)*sin(2*pi*time/T);
    y_ii = 0;
    z_ii = 0;
    roll = 0;
    pitch = 0;
```

```
yaw = 0;
roll_i = 0;
pitch_i = 0;
yaw_i = 0;
roll_ii = 0;
pitch_ii = 0;
yaw_ii = 0;
index_alpha_i = 0;
index_phi_i = 0;
index_l_i = 0;
Vx_vento = 0;
Vy_vento = 0;
Vz_vento = 0;

%%%%%%%%% Rotational Matrix %%%%%%%%%%%

R11 = cos(yaw)*cos(pitch);
R12 = cos(yaw)*sin(pitch)*sin(roll) - sin(yaw)*cos(roll);
R13 = cos(yaw)*sin(pitch)*cos(roll) + sin(yaw)*sin(roll);
R21 = sin(yaw)*cos(pitch);
R22 = sin(yaw)*sin(pitch)*sin(roll) + cos(yaw)*cos(roll);
R23 = sin(yaw)*sin(pitch)*cos(roll) - cos(yaw)*sin(roll);
R31 = -sin(pitch);
R32 = cos(pitch)*sin(roll);
R33 = cos(pitch)*cos(roll);

R = [ R11 R12 R13; R21 R22 R23; R31 R32 R33];

%%%%%%%% Crane position %%%%%%%%%%%%%%%%

x_base = x + R11*dx + R12*dy + R13*dz;
y_base = y + R21*dx + R22*dy + R23*dz;
z_base = z + R31*dx + R32*dy + R33*dz;

%%%% First Derivative of the Rotation Matrix %%%%%%%%%%%
```

```
R11_i = - sin(yaw)*cos(pitch)*yaw_i - cos(yaw)*sin(pitch)*pitch_i;
R12_i = - sin(yaw)*sin(pitch)*sin(roll)*yaw_i + cos(yaw)*cos(pitch)
*sin(roll)*pitch_i +cos(yaw)*sin(pitch)*cos(roll)*roll_i
- cos(yaw)*cos(roll)*yaw_i + sin(yaw)*sin(roll)*roll_i;
R13_i = - sin(yaw)*sin(pitch)*cos(roll)*yaw_i + cos(yaw)*cos(pitch)
*cos(roll)*pitch_i -cos(yaw)*sin(pitch)*sin(roll)*roll_i
+ cos(yaw)*sin(roll)*yaw_i + sin(yaw)*cos(roll)*roll_i;
R21_i = cos(yaw)*cos(pitch)*yaw_i - sin(yaw)*sin(pitch)*pitch_i;
R22_i = cos(yaw)*sin(pitch)*sin(roll)*yaw_i
+ sin(yaw)*cos(pitch)*sin(roll)*pitch_i
+sin(yaw)*sin(pitch)*cos(roll)*roll_i  - sin(yaw)*cos(roll)*yaw_i
- cos(yaw)*sin(roll)*roll_i;
R23_i = cos(yaw)*sin(pitch)*cos(roll)*yaw_i
+ sin(yaw)*cos(pitch)*cos(roll)*pitch_i
- sin(yaw)*sin(pitch)*sin(roll)*roll_i
+ sin(yaw)*sin(roll)*yaw_i - cos(yaw)*cos(roll)*roll_i;
R31_i = - cos(pitch)*pitch_i;
R32_i = - sin(pitch)*sin(roll)*pitch_i
+ cos(pitch)*cos(roll)*roll_i;
R33_i = - sin(pitch)*cos(roll)*pitch_i
- cos(pitch)*sin(roll)*roll_i;

R_i = [ R11_i R12_i R13_i;
        R21_i R22_i R23_i;
        R31_i R32_i R33_i];

%%%%%%%% Crane Speed %%%%%%%%%%%%

x_base_i = x_i + R11_i*dx + R12_i*dy + R13_i*dz;
y_base_i = y_i + R21_i*dx + R22_i*dy + R23_i*dz;
z_base_i = z_i + R31_i*dx + R32_i*dy + R33_i*dz;

%%% Second Derivative of the Rotational Speed %%%%%%%%

R11_ii = - cos(yaw)*cos(pitch)*yaw_i^2
+ sin(yaw)*sin(pitch)*yaw_i*pitch_i
```

```
- sin(yaw)*cos(pitch)*yaw_ii + sin(yaw)*sin(pitch)*yaw_i*pitch_i
- cos(yaw)*cos(pitch)*pitch_i^2 - cos(yaw)*sin(pitch)*pitch_ii;
R12_ii = - cos(yaw)*sin(pitch)*sin(roll)*yaw_i^2
- sin(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i
- sin(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i
- sin(yaw)*sin(pitch)*sin(roll)*yaw_ii
- sin(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i
- cos(yaw)*sin(pitch)*sin(roll)*pitch_i^2
+ cos(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i
+ cos(yaw)*cos(pitch)*sin(roll)*pitch_ii
- sin(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i
+ cos(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i
- cos(yaw)*sin(pitch)*sin(roll)*roll_i^2
+ cos(yaw)*sin(pitch)*cos(roll)*roll_ii
+ sin(yaw)*cos(roll)*yaw_i^2
+ cos(yaw)*sin(roll)*yaw_i*roll_i - cos(yaw)*cos(roll)*yaw_ii
+ cos(yaw)*sin(roll)*yaw_i*roll_i + sin(yaw)*cos(roll)*roll_i^2
+ sin(yaw)*sin(roll)*roll_ii;
R13_ii = - cos(yaw)*sin(pitch)*cos(roll)*yaw_i^2
- sin(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i
+ sin(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i
- sin(yaw)*sin(pitch)*cos(roll)*yaw_ii
- sin(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i
- cos(yaw)*sin(pitch)*cos(roll)*pitch_i^2
- cos(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i
+ cos(yaw)*cos(pitch)*cos(roll)*pitch_ii
+ sin(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i
- cos(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i
- cos(yaw)*sin(pitch)*cos(roll)*roll_i^2
- cos(yaw)*sin(pitch)*sin(roll)*roll_ii
- sin(yaw)*sin(roll)*yaw_i^2
+ cos(yaw)*cos(roll)*yaw_i*roll_i + cos(yaw)*sin(roll)*yaw_ii
+ cos(yaw)*cos(roll)*yaw_i*roll_i - sin(yaw)*sin(roll)*roll_i^2
+ sin(yaw)*cos(roll)*roll_ii;
R21_ii = - sin(yaw)*cos(pitch)*yaw_i^2
- cos(yaw)*sin(pitch)*yaw_i*pitch_i
```

```
+ cos(yaw)*cos(pitch)*yaw_ii
- cos(yaw)*sin(pitch)*yaw_i*pitch_i
- sin(yaw)*cos(pitch)*pitch_i^2
- sin(yaw)*sin(pitch)*pitch_ii;
R22_ii = - sin(yaw)*sin(pitch)*sin(roll)*yaw_i^2
+ cos(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i
+ cos(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i
+ cos(yaw)*sin(pitch)*sin(roll)*yaw_ii
+ cos(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i
- sin(yaw)*sin(pitch)*sin(roll)*pitch_i^2
+ sin(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i
+ sin(yaw)*cos(pitch)*sin(roll)*pitch_ii
+ cos(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i
+ sin(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i
- sin(yaw)*sin(pitch)*sin(roll)*roll_i^2
+ sin(yaw)*sin(pitch)*cos(roll)*roll_ii
- cos(yaw)*cos(roll)*yaw_i^2
+ sin(yaw)*sin(roll)*yaw_i*roll_i - sin(yaw)*cos(roll)*yaw_ii
+ sin(yaw)*sin(roll)*yaw_i*roll_i - cos(yaw)*cos(roll)*roll_i^2
- cos(yaw)*sin(roll)*roll_ii;
R23_ii = - sin(yaw)*sin(pitch)*cos(roll)*yaw_i^2
+ cos(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i
- cos(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i
+ cos(yaw)*sin(pitch)*cos(roll)*yaw_ii
+ cos(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i
- sin(yaw)*sin(pitch)*cos(roll)*pitch_i^2
- sin(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i
+ sin(yaw)*cos(pitch)*cos(roll)*pitch_ii
- cos(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i
- sin(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i
- sin(yaw)*sin(pitch)*cos(roll)*roll_i^2
- sin(yaw)*sin(pitch)*sin(roll)*roll_ii
+ cos(yaw)*sin(roll)*yaw_i^2
+ sin(yaw)*cos(roll)*yaw_i*roll_i + sin(yaw)*sin(roll)*yaw_ii
+ sin(yaw)*cos(roll)*yaw_i*roll_i + cos(yaw)*sin(roll)*roll_i^2
- cos(yaw)*cos(roll)*roll_ii;
```

```
    R31_ii = sin(pitch)*pitch_i^2 - cos(pitch)*pitch_ii;
    R32_ii = - cos(pitch)*sin(roll)*pitch_i^2
    - sin(pitch)*cos(roll)*pitch_i*roll_i
    - sin(pitch)*sin(roll)*pitch_ii
    - sin(pitch)*cos(roll)*pitch_i*roll_i
    - cos(pitch)*sin(roll)*roll_i^2
    + cos(pitch)*cos(roll)*roll_ii;
    R33_ii = - cos(pitch)*cos(roll)*pitch_i^2
    + sin(pitch)*sin(roll)*pitch_i*roll_i
    - sin(pitch)*cos(roll)*pitch_ii
    + sin(pitch)*sin(roll)*pitch_i*roll_i
    - cos(pitch)*cos(roll)*roll_i^2
    - cos(pitch)*sin(roll)*roll_ii;

    R_ii = [ R11_ii R12_iiR13_ii;
             R21_ii R22_ii R23_ii;
             R31_ii R32_ii R33_ii];

%%%%%%%% Crane Acceleration %%%%%%%%%%

    x_base_ii = x_ii + R11_ii*dx + R12_ii*dy + R13_ii*dz;
    y_base_ii = y_ii + R21_ii*dx + R22_ii*dy + R23_ii*dz;
    z_base_ii = z_ii + R31_ii*dx + R32_ii*dy + R33_ii*dz;

%%%% Alpha, Phi and l speed %%%%%%%%%%

    alpha_i = index_alpha_i*pi/180; %1 grau por segundo
    phi_i = index_phi_i*pi/30; % 6 graus por segundo
    l_i = index_l_i*0.1; % 10cm por segundo

%%%% Update Alpha, Phi and l values %%%%%%%%

    alpha = alpha + alpha_i*0.1;
    phi = phi + phi_i*0.1;
    l = l + l_i*0.1;
```

```
%%%% Alpha, Phi and l Acceleration %%%%%%%%

    alpha_ii = ((index_alpha_i - index_i_alpha_i)*pi/180)/0.1;
    phi_ii = ((index_phi_i - index_i_phi_i)*pi/30)/0.1;
    l_ii = ((index_l_i - index_i_l_i)*0.1)/0.1;

%%%% Superior part of the crane parameters %%%%%%%%%%

    x_sup = x_base + L*cos(alpha+pitch)*cos(phi+yaw);
    y_sup = y_base + L*cos(alpha+roll)*sin(phi+yaw);
    z_sup = z_base + L*sin(alpha+roll+pitch);

    x_sup_i = x_base_i - L*cos(alpha+pitch)*sin(phi+yaw)
    *(phi_i+yaw_i) - L*sin(alpha+pitch)*cos(phi+yaw)
    *(alpha_i+pitch_i);
    y_sup_i = y_base_i - L*sin(alpha+roll)*sin(phi+yaw)
    *(alpha_i+roll_i) + L*cos(alpha+roll)*cos(phi+yaw)
    *(phi_i+yaw_i);
    z_sup_i = z_base_i + L*cos(alpha+roll+pitch)
    *(alpha_i+roll_i+pitch_i);

    x_sup_ii = x_base_ii - L*cos(alpha+pitch)*cos(phi+yaw)
    *((phi_i+yaw_i)^2) -L*cos(alpha+pitch)
    *sin(phi+yaw)*(phi_ii+yaw_ii)
    + 2*L*sin(alpha+pitch)*sin(phi+yaw)*(alpha_i+pitch_i)
    *(phi_i+yaw_i)-L*cos(alpha+pitch)*cos(phi+yaw)
    *((alpha_i+pitch_i)^2)-L*sin(alpha+pitch)
    *cos(phi+yaw)*(alpha_ii+pitch_ii);
    y_sup_ii = y_base_ii - L*cos(alpha+roll)*sin(phi+yaw)
    *((alpha_i+roll_i)^2)-L*sin(alpha+roll)
    *sin(phi+yaw)*(alpha_ii+roll_ii)
    - 2*L*sin(alpha+roll)*cos(phi+yaw)*(alpha_i+roll_i)
    *(phi_i+yaw_i)
    - L*cos(alpha+roll)*sin(phi+yaw)*((phi_i+yaw_i)^2)
    + L*cos(alpha+roll)*cos(phi+yaw)*(phi_ii+yaw_ii);
    z_sup_ii = z_base_ii - L*sin(alpha+roll+pitch)
```

```
    *((alpha_i+roll_i+pitch_i)^2) + L*cos(alpha+roll+pitch)
    *(alpha_ii+roll_ii+pitch_ii);

%%%%%%%%%% Wind Force %%%%%%%%%%%%

    Fx_vento = rho*C*Ax*(Vx_vento)^2;
    Fy_vento = rho*C*Ay*(Vy_vento)^2;
    Fz_vento = rho*C*Az*(Vz_vento)^2;

%%%%%%%%%% Differential Equations %%%%%%%%%%%%

    A_1 = 2*beta_22_i*beta_22 - 2*l_i/l;
    A_2 = 2*beta_11_i*beta_11 - 2*l_i/l;

    B = Fz_vento/(m*l) - z_sup_ii/l - g/l;

    C_1 = Fx_vento/(m*l) - x_sup_ii/l;
    C_2 = Fy_vento/(m*l) - y_sup_ii/l;

    tspan = [time time+0.1];

    beta_10 = [beta_11 beta_11_i];
    beta_20 = [beta_22 beta_22_i];

%%%%%%%%%% Non Linear %%%%%%%%%%%%%%

    [t_1,sol_1] = ode45(@(t,y) odefcn_b1(t,y,m,l,Fx_vento,Fy_vento
    ,Fz_vento,x_sup_ii,y_sup_ii,z_sup_ii,l_i,g,l_ii,beta_22,beta_22_i,be
    ,tspan, beta_10);
    [t_2,sol_2] = ode45(@(t,y) odefcn_b2(t,y,m,l,Fx_vento,Fy_vento
    ,Fz_vento,x_sup_ii,y_sup_ii,z_sup_ii,l_i,g,l_ii,beta_11,beta_11_i,
    beta_11_ii), tspan, beta_20);



%%%%%%%%% Solutions %%%%%%%%%%%%%
```

```
a_1 = size(sol_1);
a_2 = size(sol_2);

b1 = sol_1(a_1(1));
b_1_i = sol_1(2*a_1(1));

b2 = sol_2(a_2(1));
b_2_i = sol_2(2*a_2(1));

%%%%%%%%%% Load Data %%%%%%%%%%%%

beta_1_ii = (2*b_2_i*b2 - 2*l_i/l)*b_1_i
+ (Fz_vento/(m*l) - z_sup_ii/l
- g/l)*b1 + Fx_vento/(m*l)
- x_sup_ii/l;
beta_2_ii = (2*b_1_i*b1 -
2*l_i/l)*b_2_i
+ (Fz_vento/(m*l) - z_sup_ii/l
- g/l)*b2 + Fy_vento/(m*l)
- y_sup_ii/l;

x_carga = x_sup + l*sin(b1)*cos(b2);
y_carga = y_sup + l*cos(b1)*sin(b2);
z_carga = z_sup - l*cos(b1)*cos(b2);

x_carga_i = x_sup_i
+l*cos(b1)*cos(b2)*b_1_i
- l*sin(b1)*sin(b2)*b_2_i
+ l_i*sin(b1)*cos(b2);
y_carga_i = y_sup_i
- l*sin(b1)*sin(b2)*b_1_i
+ l*cos(b1)*cos(b2)*b_2_i
+ l_i*cos(b1)*sin(b2);
z_carga_i = z_sup_i
+ l*sin(b1)*cos(b2)*b_1_i
```

```
+ l*cos(b1)*sin(b2)*b_2_i
- l_i*cos(b1)*cos(b2);

x_carga_ii = x_sup_ii + l_i*cos(b1)*cos(b2)*b_1_i -
l*sin(b1)*cos(b2)*(b_1_i^2) - l*cos(b1)*sin(b2)*b_1_i*b_2_i
+ l*cos(b1)*cos(b2)*beta_1_ii - l_i*sin(b1)*sin(b2)*(b_2_i^2)
- l*cos(b1)*sin(b2)*b_1_i*b_2_i - l*sin(b1)*cos(b2)*(b_2_i^2)
- l*sin(b1)*sin(b2)*beta_2_ii + l_i*cos(b1)*cos(b2)*b_1_i
- l_i*sin(b1)*sin(b2)*b_2_i;
y_carga_ii = y_sup_ii - l_i*sin(b1)*sin(b2)*b_1_i
- l*cos(b1)*sin(b2)*(b_1_i^2) - l*sin(b1)*cos(b2)*b_1_i*b_2_i
- l*sin(b1)*sin(b2)*beta_1_ii + l_i*cos(b1)*cos(b2)*(b_2_i^2)
- l*sin(b1)*cos(b2)*b_1_i*b_2_i - l*cos(b1)*sin(b2)*(b_2_i^2)
+ l*cos(b1)*cos(b2)*beta_2_ii - l_i*sin(b1)*sin(b2)*b_1_i
+ l_i*cos(b1)*cos(b2)*b_2_i;
z_carga_ii = z_sup_ii + l_i*sin(b1)*cos(b2)*b_1_i
+ l*cos(b1)*cos(b2)*(b_1_i^2) - l*sin(b1)*sin(b2)*b_1_i*b_2_i
+ l*sin(b1)*cos(b2)*beta_1_ii - l_i*cos(b1)*sin(b2)*(b_2_i^2)
+ l*sin(b1)*sin(b2)*b_1_i*b_2_i - l*cos(b1)*cos(b2)*(b_2_i^2)
- l*cos(b1)*sin(b2)*beta_2_ii - l_i*sin(b1)*cos(b2)*b_1_i
- l_i*cos(b1)*sin(b2)*b_2_i;

%%%%%%%%%%% Load Total Speed %%%%%%%%%%%

carga_i = ((x_carga_i^2) + (y_carga_i^2) + (z_carga_i^2))^0.5;

%%%%%%%%%%% Cable Tension %%%%%%%%%%%%%%%

a_rel_1 = (z_sup_ii - z_carga_ii)*cos(b1)*cos(b2)
+ (y_sup_ii - y_carga_ii)*cos(b1)*sin(b2)
+ (x_sup_ii - x_carga_ii)*sin(b1)*cos(b2);
a_rel_2 = z_sup_ii*cos(b1)*cos(b2)
+ y_sup_ii*cos(b1)*sin(b2)
+ x_sup_ii*sin(b1)*cos(b2);

Tracao = m*(((carga_i^2)/l) + g*cos(b1)*cos(b2));
```

```
Tracao_1 = m*(((carga_i^2)/l) + g*cos(b1)*cos(b2) + a_rel_1);
Tracao_2 = m*(((carga_i^2)/l) + g*cos(b1)*cos(b2) + a_rel_2);

Fx = Tracao*sin(b1)*cos(b2);
Fy = Tracao*cos(b1)*sin(b2);
Fz = -Tracao*cos(b1)*cos(b2);
Mx = -Tracao*cos(b1)*cos(b2)*L*cos(alpha)*sin(phi)
- Tracao*cos(b1)*sin(b2)*L*sin(alpha);
My = Tracao*cos(b1)*cos(b2)*L*cos(alpha)*cos(phi)
+ Tracao*sin(b1)*cos(b2)*L*sin(alpha);
Mz = Tracao*sin(b1)*cos(b2)*L*cos(alpha)*sin(phi)
+ Tracao*cos(b1)*sin(b2)*L*cos(alpha)*cos(phi);
```

%%%%%%%%%%%% Grafs %%%%%%%%%%%%%%%

```
X_graf = [ x_base x_sup];
Y_graf = [ y_base y_sup];
Z_graf = [ z_base z_sup];

x_graf = [ x_sup x_carga];
y_graf = [ y_sup y_carga];
z_graf = [ z_sup z_carga];

disp(time);

a = 10*time + 1;

b = uint64(a);

angulo_1(b) = (180/pi)*b1;

angulo_2(b) = (180/pi)*b2;

Forca_x(b) = Fx;
Forca_y(b) = Fy;
Forca_z(b) = Fz;
```

```
        Momento_x(b) = Mx;
        Momento_y(b) = My;
        Momento_z(b) = Mz;

        tracao_cabo(b) = Tracao;
        tracao_cabo_1(b) = Tracao_1;
        tracao_cabo_2(b) = Tracao_2;

%%%%%%%% Update Values %%%%%%%%%%%%%%%

        beta_11 = b1;
        beta_11_i = b_1_i;
        beta_11_ii = beta_1_ii;
        beta_22 = b2;
        beta_22_i = b_2_i;
        beta_22_ii = beta_2_ii;

        index_i_alpha_i = index_alpha_i;
        index_i_phi_i = index_phi_i;
        index_i_l_i = index_l_i;

end
```

ANEXO B

Manual

JULIO ARANTES
PEDRO HENRIQUE DIAS SILVA

MANUAL DO USUÁRIO
SIMULADOR DE GUINDASTE DO SMH

São Paulo
2018

SUMÁRIO

4

1 DESCRIÇÃO DO SOFTWARE

O software do simulador de guindaste consiste em um modelo dinâmico de um pêndulo, feito em MATLAB, que proporciona um suporte ao Simulador Marítimo Hidroviário da Universidade de São Paulo, no ramo de simulação de guindastes embarcados em sistemas Offshore. Este modelo é baseado na estrutura de um guindaste do tipo "Boom Crane", que se encontra alocado em uma embarcação. Deste modo, o usuário pode usufruir da possibilidade de manejar um guindaste com a física semelhante à realidade, porém computacionalmente. Com isso, o software tem como objetivo auxiliar o desenvolvimento de novos profissionais para a área de manejo de cargas portuárias, e também proporcionar teste de novos equipamentos para situações ainda não testadas em campo.

5

2 INFORMAÇÕES TÉCNICAS

O software do guindaste foi projetado para funcionar em um ambiente semelhante ao Simulador Marítimo Hidroviário da Universidade de São Paulo, logo possui algumas restrições e requisitos para sua performance. Para que o software possa ser executado corretamente, o simulador em que o mesmo será instalado deve conter:

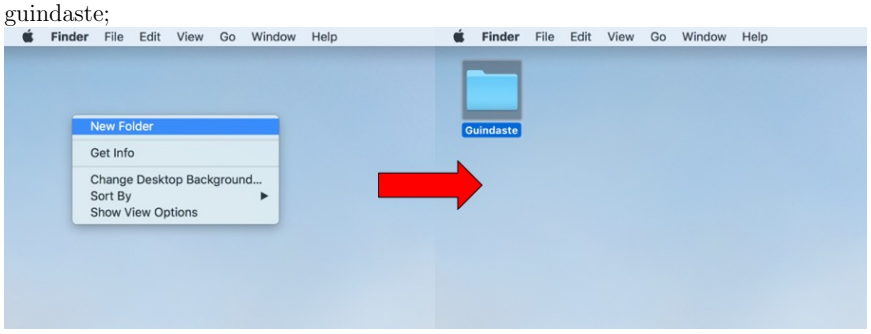
- MATLAB 2013b, ou mais recente;
- Conexão MODBUS(BUZZ);
- NMEA;
- Sistema de processamento para UNITY independente;
- Sistema de processamento para Servidor independente;
- Conexão com a internet;
- Conexão com a Skynet

6

3 INSTALAÇÃO

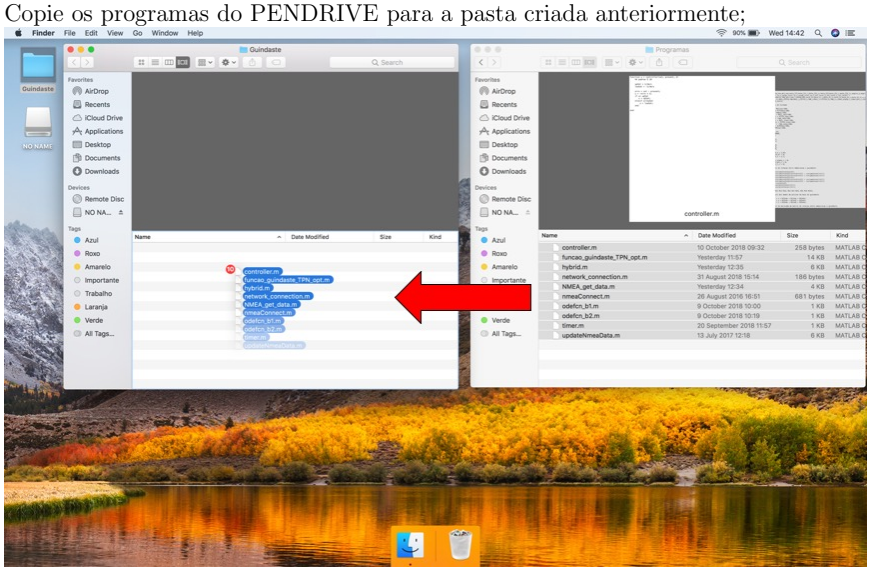
A instalação do software do guindaste consiste na alocação de todos os seus programas para uma mesma pasta do MATLAB. Para isso, basta seguir os seguintes passos:

1. Crie uma pasta em seu computador com o nome que deseja para o software do guindaste;



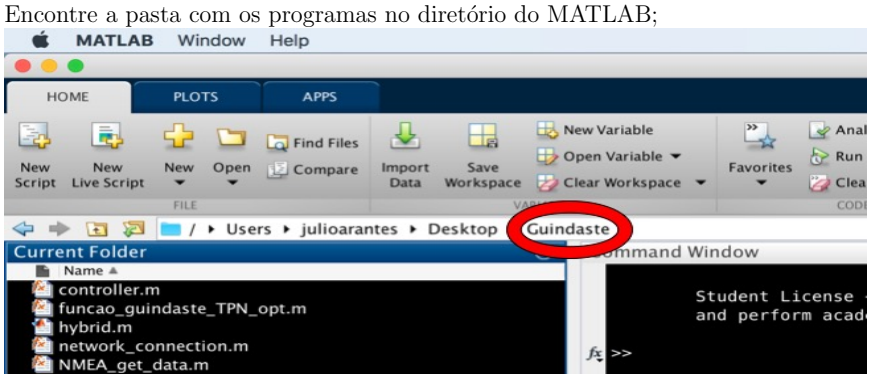
2. Insira o PENDRIVE com os arquivos no USB de seu computador;

3. Copie os programas do PENDRIVE para a pasta criada anteriormente;

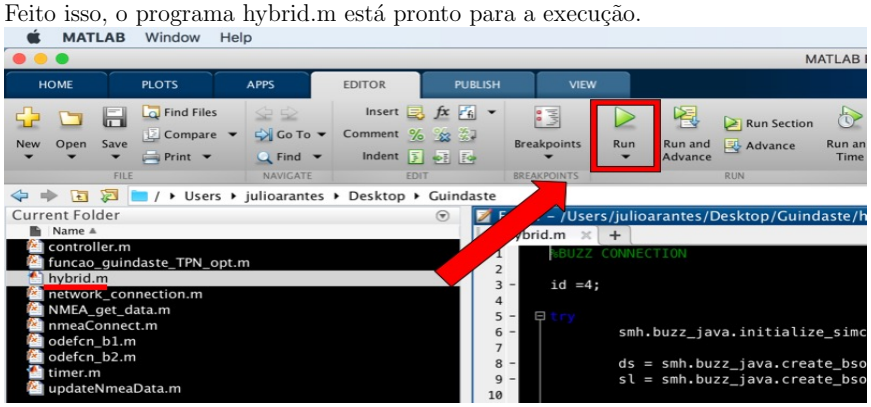


4. Inicie o MATLAB;

5. Encontre a pasta com os programas no diretório do MATLAB;



6. Feito isso, o programa hybrid.m está pronto para a execução.



Obs.: a pasta do MATLAB deve conter os seguintes programas:

- controller.m
- funcao_guindaste_TPN_opt.m
- hybrid.m
- network_connection.m
- NMEA_get_data.m
- nmeaConnect.m
- odefcn.m
- odefcn_b1.m
- odefcn_b2.m

9

4 UTILIZAÇÃO

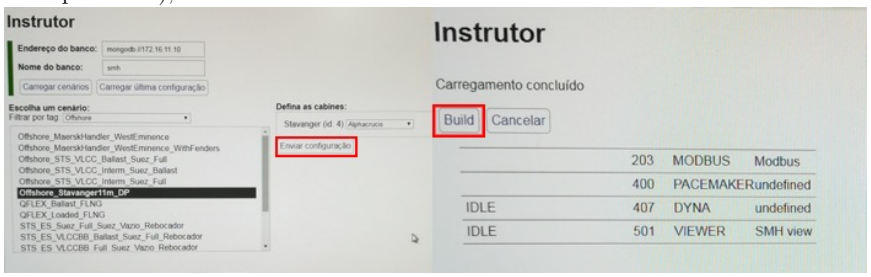
4.1 Utilização do Técnico

A utilização por parte do técnico deve seguir os seguintes passos:

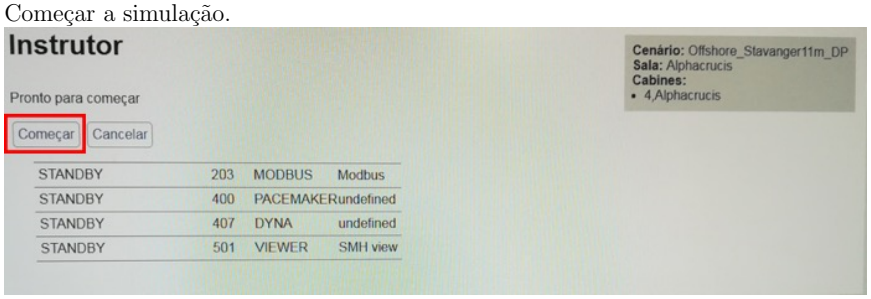
1. Ligar todos os computadores (servidor, visual e operador);
2. No programa hybrid.m adequar o IP do servidor em que a simulação irá ocorrer (vide alteração de IP de servidor na seção de alteração de parâmetros do código-fonte);
3. No programa hybrid.m adequar o IP do banco de dados que serão obtidos os dados para a simulação (vide alteração de IP de banco de dados na seção de alteração de parâmetros do código-fonte);
4. No programa hybrid.m adequar o ID do navio que ocorrerá a simulação (vide alteração de ID da embarcação na seção de alteração de parâmetros do código-fonte);
5. Iniciar todos os processos do simulador no Skynet;

6. Iniciar o programa hybrid.m no MATLAB (vide figura do passo 6 da instalação);

7. Construir a simulação desejada no computador do servidor (por exemplo Offshore_Stavanger11m no Alphacrusis);



8. Começar a simulação.



4.2 Utilização do Operador

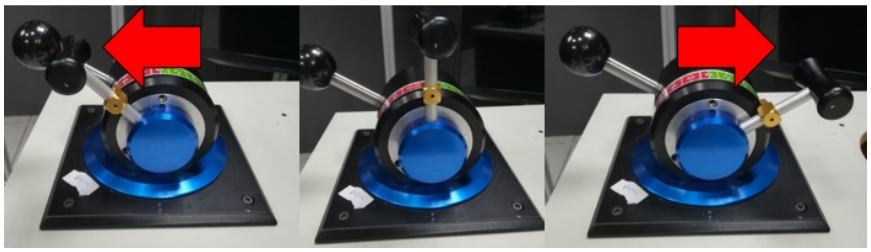
Com a simulação iniciada, o operador tem as seguintes opções de atividades:

- Manipulação da rotação do guindaste

  O operador tem a capacidade, a partir do movimento de rotação no manete abaixo, de alterar a velocidade de rotação do guindaste entre 6 graus por segundo no sentido anti-horário, até 6 graus por segundo no sentido horário, como indica a figura a seguir.

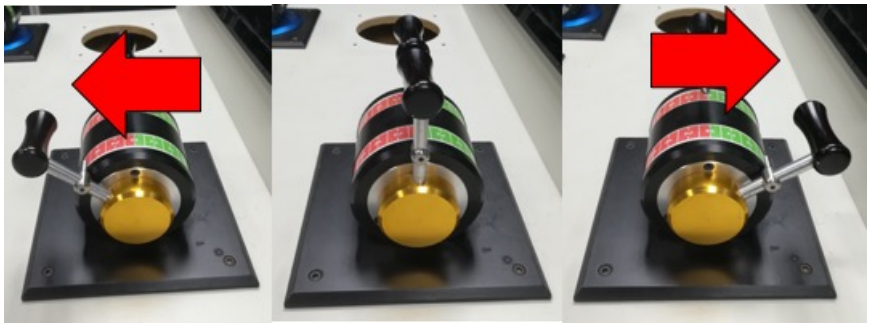
- Manipulação da inclinação do guindaste

  O operador tem a capacidade, a partir do movimento de alavanca do manete abaixo, de alterar a velocidade de inclinação do guindaste entre 1 grau por segundo para cima, ou 1 grau por segundo para baixo, como indica a figura a seguir.



- Manipulação do Tamanho do Cabo

  O operador tem a capacidade, a partir do movimento de alavanca do manete abaixo, de alterar a velocidade do cabo do guindaste entre 1 metro por segundo para cima, ou 1 metro por segundo para baixo, como indica a figura a seguir.



- Ativação do controle de altura da carga O operador tem a capacidade, a partir do movimento de rotação do comando abaixo, de ligar o controlador de altura da carga, vermelho, ou de desligar o controlador de altura da carga, no verde, como mostra a figura abaixo.

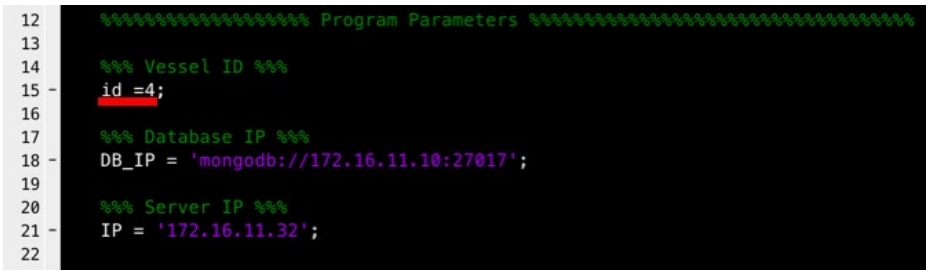
12

13

5 ALTERAÇÕES DE PARÂMETRO NO CÓDIGO-FONTE

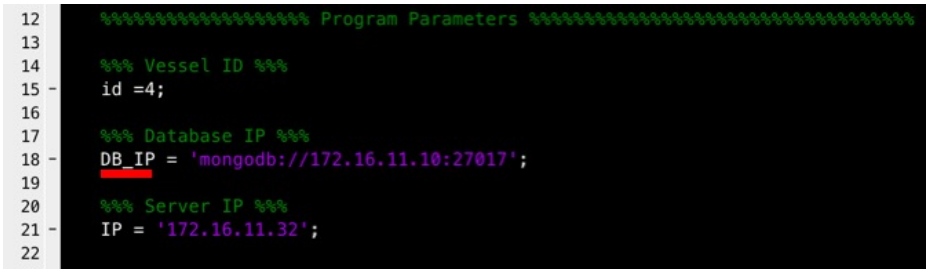
5.1 Alteração do ID da Embarcação

Para alterar o ID da embarcação da simulação deve-se ir ao código do programa hybrid.m, na linha 15, e colocar o ID desejado.



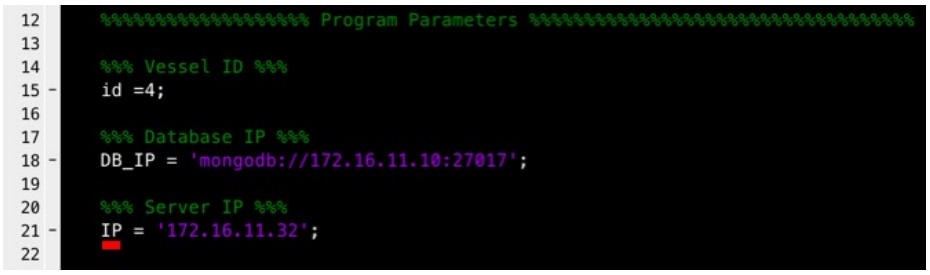
5.2 Alteração do IP do Banco de Dados

Para alterar o IP do banco de dados da simulação deve-se ir ao código do programa hybrid.m, na linha 18, e inserir o IP do banco de dados desejado.



5.3 Alteração do IP do Servidor

Para alterar o IP do servidor da simulação deve-se ir ao código do programa hybrid.m, na linha 21, e inserir o IP do servidor desejado.



5.4 Alteração dos Parâmetros Iniciais da Simulação

Para alterar os parâmetros iniciais da simulação deve-se ir ao código do programa hybrid.m, nas linhas 58 a 72, e inserir os valores desejados.

Na figura podemos identificar os seguintes parâmetros:

- OnOff: controle de altura da carga em booleano;
- Set_i: altura da carga com o controle ligado em metros;
- I: comprimento do cabo em metros;
- alpha: inclinação do guindaste em graus;
- phi: rotação do guindaste em graus;

- beta_11: ângulo β_1 em radianos;
- beta_11_i: velocidade do ângulo β_1 em radianos;
- beta_11_ii: aceleração do ângulo β_1 em radianos por segundo ao quadrado;
- beta_22: ângulo β_2 em radianos;
- beta_22_i: velocidade do ângulo β_2 em radianos;
- beta_22_ii: aceleração do ângulo β_2 em radianos por segundo ao quadrado;
- index_alpha_i: velocidade de inclinação do guindaste parametrizada de -1 a 1;
- index_l_i: velocidade de içamento do cabo parametrizada de -1 a 1;
- Vx_wind: velocidade do vento na direção x;
- Vy_wind: velocidade do vento na direção y;
- Vz_wind: velocidade do vento na direção z;
- time: tempo de simulação em segundos;
- limit: parâmetros de atuação do limite do cabo;
- l_limit: limite inferior do tamanho do cabo.

5.5 Alteração dos Parâmetros do Guindaste

Para alterar o valor do comprimento do guindaste e sua localização na embarcação deve-se ir ao código do programa funcao_guindaste_TPN_opt.m, na linha 79, e inserir o tamanho desejado, e nas linhas 83 a 85, e inserir a posição do guindaste na embarcação mais a localização do início da lança com relação a base do guindaste em cada eixo.

5.6 Alteração dos Parâmetros da Carga

Para alterar os parâmetros da carga deve-se ir ao código do programa funcao_guindaste_TPN_opt.m, na linha 89, e inserir a massa da carga, e nas linhas 93 a 95, e inserir as áreas da carga em cada plano.

```
86
87      %%%%% Load Mass %%%%%%%
88
89 -        m = 270000;
90
91      %%%%% Load Area %%%%%%
92
93 -        Ax = 50;
94 -        Ay = 50;
95 -        Az = 50;
96
```

5.7 Alteração dos Limites de Velocidade do Operador

Para alterar as velocidades de operação do operador deve-se ir ao código do programa funcao_guindaste_TPN_opt.m, nas linhas 99 a 101, e inserir os valores das velocidades em radianos por segundo ou metros por segundo.

```
96
97      %%%% Operational Velocities %%%%%
98
99 -        alpha_vel = pi/180;
100 -       phi_vel = pi/30;
101 -       l_vel = 1;
102
```

Na figura podemos identificar as seguintes velocidades:

- alpha_vel: velocidade de inclinação do guindaste;
- phi_vel: velocidade de rotação do guindaste;
- l_vel: velocidade de içamento do cabo.

17

6 CÓDIGOS

6.1 Programa do Controlador(controller.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       Controller Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: October, 2018
%
% Input: - set (set value for the measured value)
%        - present (present value of the measured value)
%        - k (proportional gain of the controller)
%
% Output:  u (controlled input for the plant)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%% Function %%%%%%%%

function u = controller(set, present, k)

%%%% Saturation Values %%%

    upSat = 1; % [m/s]
    lowSat = -1; % [m/s]

%%%% Error Calculation %%%
```

```
    erro = set - present;
    u = -erro * k;

%%%%%% Saturation %%%%%%

    if u> upSat
       u = upSat;
    elseif u<lowSat
        u = lowSat;
    end

%%%%%%%%%%%%%%%%%%%%%%%%

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.2 Programa da Dinâmica do Guindaste (funcao_guindaste_TPN_opt.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       Crane Dynamics Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: September, 2018
%
% Input: - time (time)
%        - l (cable length)
%        - ALPHA (inclination angle)
%        - PHI (rotation angle)
%        - beta_11 (initial value for Beta 1)
%        - beta_11_i (intial value for Beta 1 first derivative)
%        - beta_11_ii (intial value for Beta 1 second derivative)
```

```
%         - beta_22 (initial value for Beta 2)
%         - beta_22_i (initial value for Beta 2 first derivative)
%         - beta_22_ii (initial value for Beta 2 second derivative)
%         - x (Vessel position on x-axis)
%         - y (Vessel position on y-axis)
%         - z (Vessel position on z-axis)
%         - x_i (Vessel velocity on x-axis)
%         - y_i (Vessel velocity on y-axis)
%         - z_i (Vessel velocity on z-axis)
%         - x_ii (Vessel acceleration on x-axis)
%         - y_ii (Vessel acceleration on y-axis)
%         - z_ii (Vessel acceleration on z-axis)
%         - ROLL (Vessel roll angle)
%         - PITCH (Vessel pitch angle)
%         - YAW (Vessel yaw angle)
%         - ROLL_i (Vessel roll velocity)
%         - PITCH_i (Vessel pitch velocity)
%         - YAW_i (Vessel yaw velocity)
%         - ROLL_ii (Vessel roll acceleration)
%         - PITCH_ii (Vessel pitch acceleration)
%         - YAW_ii (Vessel yaw acceleration)
%         - index_alpha_i (alpha velocity acquired by the JoyStick)
%         - index_phi_i (phi velocity acquired by the JoyStick)
%         - index_l_i (cable velocity acquired by the JoyStick)
%         - Vx_wind (Wind speed on x-axis)
%         - Vy_wind (Wind speed on y-axis)
%         - Vz_wind (Wind speed on z-axis)
%
% Output: - l_new (new value for the cable length)
%          - alpha_new (new value for the inclination angle)
%          - phi_new (new value for the rotation angle)
%          - beta_111 (Beta 1 value)
%          - beta_111_i (Beta 1 first derivative value)
%          - beta_111_ii (Beta 1 second derivative value)
%          - beta_222 (Beta 2 value)
%          - beta_222_i (Beta 2 first derivative value)
```

```
%         - beta_222_ii (Beta 2 second derivative value)
%         - Tension (Cable tension)
%         - Fx (Reactive force on the crane base on x-axis)
%         - Fy (Reactive force on the crane base on y-axis)
%         - Fz (Reactive force on the crane base on z-axis)
%         - Mx (Reactive moment on the crane base on x-axis)
%         - My (reactive moment on the crane base on y-axis)
%         - Mz (Reactive moment on the crane base on z-azis)
%         - PHI (phi angle)
%         - ALPHA (alpha angle)
%         - x_load_local_zy (hook position on crane x-axis)
%         - y_load_local_zy (hook position on crane y-axis)
%         - z_load_local_zy (hook position on crane z-axis)
%         - x_inf_local (load position on ship x-axis)
%         - y_inf_local (load position on ship y-axis)
%         - z_inf_local (load position on ship z-axis)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Function %%%%%%%

function [l_new,alpha_new,phi_new,beta_111,beta_111_i,beta_111_ii,
beta_222,beta_222_i,beta_222_ii,Tension,Fx,Fy,Fz,Mx,My,Mz,
x_load_local_zy,y_load_local_zy,z_load_local_zy,x_inf_local,
y_inf_local,z_inf_local]
= funcao_guindaste_TPN_opt(time,l,ALPHA,PHI,beta_11,beta_11_i,
beta_11_ii,beta_22,beta_22_i,beta_22_ii,x,y,z,x_i,y_i,z_i,x_ii,
y_ii,z_ii,ROLL,PITCH,YAW,ROLL_i,PITCH_i,YAW_i,ROLL_ii,PITCH_ii,
YAW_ii,index_alpha_i,index_phi_i,index_l_i,Vx_wind,Vy_wind,Vz_wind)

%%%%%%%%%%%%%%%%%%%%%%%%% Program Parameters %%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Crane Length %%%%%%

    L = 41.43;
```

```
%%%%% Crane Position %%%%%%

    dx = 74.4 + 2.55;
    dy = -16.67 + 0;
    dz = 23.2 + 3.7;

%%%%% Load Mass %%%%%%%

    m = 270000;

%%%%% Load Area %%%%%%

    Ax = 50;
    Ay = 50;
    Az = 50;

%%%% Operational Velocities %%%%

    alpha_vel = pi/180;
    phi_vel = pi/30;
    l_vel = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%% Angles in radians %%%%%%%%

    roll = ROLL*pi/180;
    pitch = PITCH*pi/180;
    yaw = -YAW*pi/180;
    roll_i = ROLL_i*pi/180;
    pitch_i = PITCH_i*pi/180;
    yaw_i = -YAW_i*pi/180;
    roll_ii = ROLL_ii*pi/180;
    pitch_ii = PITCH_ii*pi/180;
    yaw_ii = -YAW_ii*pi/180;
    alpha = ALPHA*pi/180;
```

```
    phi = PHI*pi/180;

%%%%% System Parameters %%%%%%%

    g = 9.8;
    rho = 1;
    C = 1.2;

%%%%% Initial values %%%%%%%%

    index_i_alpha_i = 0;
    index_i_phi_i = 0;
    index_i_l_i = 0;

%%%%% Ratation Matrix %%%%%%%

    R11 = cos(yaw)*cos(pitch);
    R12 = cos(yaw)*sin(pitch)*sin(roll) - sin(yaw)*cos(roll);
    R13 = cos(yaw)*sin(pitch)*cos(roll) + sin(yaw)*sin(roll);
    R21 = sin(yaw)*cos(pitch);
    R22 = sin(yaw)*sin(pitch)*sin(roll) + cos(yaw)*cos(roll);
    R23 = sin(yaw)*sin(pitch)*cos(roll) - cos(yaw)*sin(roll);
    R31 = -sin(pitch);
    R32 = cos(pitch)*sin(roll);
    R33 = cos(pitch)*cos(roll);

    R = [ R11 R12 R13;
          R21 R22 R23;
          R31 R32 R33];

    x_base = x + R11*dx + R12*dy + R13*dz;
    y_base = y + R21*dx + R22*dy + R23*dz;
    z_base = z + R31*dx + R32*dy + R33*dz;

%%% First derivative of Rotation Matrix %%%%%%%%
```

```
R11_i = - sin(yaw)*cos(pitch)*yaw_i - cos(yaw)*sin(pitch)*pitch_i;
R12_i = - sin(yaw)*sin(pitch)*sin(roll)*yaw_i +
cos(yaw)*cos(pitch)*sin(roll)*pitch_i
+cos(yaw)*sin(pitch)*cos(roll)*roll_i -
cos(yaw)*cos(roll)*yaw_i + sin(yaw)*sin(roll)*roll_i;
R13_i = - sin(yaw)*sin(pitch)*cos(roll)*yaw_i +
cos(yaw)*cos(pitch)*cos(roll)*pitch_i -
cos(yaw)*sin(pitch)*sin(roll)*roll_i +
cos(yaw)*sin(roll)*yaw_i + sin(yaw)*cos(roll)*roll_i;
R21_i = cos(yaw)*cos(pitch)*yaw_i -
sin(yaw)*sin(pitch)*pitch_i;
R22_i = cos(yaw)*sin(pitch)*sin(roll)*yaw_i +
sin(yaw)*cos(pitch)*sin(roll)*pitch_i +
sin(yaw)*sin(pitch)*cos(roll)*roll_i -
sin(yaw)*cos(roll)*yaw_i - cos(yaw)*sin(roll)*roll_i;
R23_i = cos(yaw)*sin(pitch)*cos(roll)*yaw_i +
sin(yaw)*cos(pitch)*cos(roll)*pitch_i -
sin(yaw)*sin(pitch)*sin(roll)*roll_i +
sin(yaw)*sin(roll)*yaw_i - cos(yaw)*cos(roll)*roll_i;
R31_i = - cos(pitch)*pitch_i;
R32_i = - sin(pitch)*sin(roll)*pitch_i +
cos(pitch)*cos(roll)*roll_i;
R33_i = - sin(pitch)*cos(roll)*pitch_i -
cos(pitch)*sin(roll)*roll_i;

R_i = [ R11_i R12_i R13_i;
        R21_i R22_i R23_i;
        R31_i R32_i R33_i];

x_base_i = x_i + R11_i*dx + R12_i*dy + R13_i*dz;
y_base_i = y_i + R21_i*dx + R22_i*dy + R23_i*dz;
z_base_i = z_i + R31_i*dx + R32_i*dy + R33_i*dz;

%%%% Second derivative of Rotation Matrix %%%%%%%%%

R11_ii = - cos(yaw)*cos(pitch)*yaw_i^2 +
```

```
sin(yaw)*sin(pitch)*yaw_i*pitch_i -
sin(yaw)*cos(pitch)*yaw_ii +
sin(yaw)*sin(pitch)*yaw_i*pitch_i -
cos(yaw)*cos(pitch)*pitch_i^2 -
cos(yaw)*sin(pitch)*pitch_ii;
R12_ii = - cos(yaw)*sin(pitch)*sin(roll)*yaw_i^2 -
sin(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i -
sin(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i -
sin(yaw)*sin(pitch)*sin(roll)*yaw_ii -
sin(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i -
cos(yaw)*sin(pitch)*sin(roll)*pitch_i^2 +
cos(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i +
cos(yaw)*cos(pitch)*sin(roll)*pitch_ii -
sin(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i +
cos(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i -
cos(yaw)*sin(pitch)*sin(roll)*roll_i^2 +
cos(yaw)*sin(pitch)*cos(roll)*roll_ii +
sin(yaw)*cos(roll)*yaw_i^2 +
cos(yaw)*sin(roll)*yaw_i*roll_i -
cos(yaw)*cos(roll)*yaw_ii +
cos(yaw)*sin(roll)*yaw_i*roll_i +
sin(yaw)*cos(roll)*roll_i^2 +
sin(yaw)*sin(roll)*roll_ii;
R13_ii = - cos(yaw)*sin(pitch)*cos(roll)*yaw_i^2 -
sin(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i +
sin(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i -
sin(yaw)*sin(pitch)*cos(roll)*yaw_ii -
sin(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i -
cos(yaw)*sin(pitch)*cos(roll)*pitch_i^2 -
cos(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i +
cos(yaw)*cos(pitch)*cos(roll)*pitch_ii +
sin(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i -
cos(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i -
cos(yaw)*sin(pitch)*cos(roll)*roll_i^2 -
cos(yaw)*sin(pitch)*sin(roll)*roll_ii -
sin(yaw)*sin(roll)*yaw_i^2 +
```

```
cos(yaw)*cos(roll)*yaw_i*roll_i +
cos(yaw)*sin(roll)*yaw_ii +
cos(yaw)*cos(roll)*yaw_i*roll_i -
sin(yaw)*sin(roll)*roll_i^2 +
sin(yaw)*cos(roll)*roll_ii;
R21_ii = - sin(yaw)*cos(pitch)*yaw_i^2 -
cos(yaw)*sin(pitch)*yaw_i*pitch_i +
cos(yaw)*cos(pitch)*yaw_ii -
cos(yaw)*sin(pitch)*yaw_i*pitch_i -
sin(yaw)*cos(pitch)*pitch_i^2 -
sin(yaw)*sin(pitch)*pitch_ii;
R22_ii = - sin(yaw)*sin(pitch)*sin(roll)*yaw_i^2 +
cos(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i +
cos(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i +
cos(yaw)*sin(pitch)*sin(roll)*yaw_ii +
cos(yaw)*cos(pitch)*sin(roll)*yaw_i*pitch_i -
sin(yaw)*sin(pitch)*sin(roll)*pitch_i^2 +
sin(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i +
sin(yaw)*cos(pitch)*sin(roll)*pitch_ii +
cos(yaw)*sin(pitch)*cos(roll)*yaw_i*roll_i +
sin(yaw)*cos(pitch)*cos(roll)*pitch_i*roll_i -
sin(yaw)*sin(pitch)*sin(roll)*roll_i^2 +
sin(yaw)*sin(pitch)*cos(roll)*roll_ii -
cos(yaw)*cos(roll)*yaw_i^2 +
sin(yaw)*sin(roll)*yaw_i*roll_i -
sin(yaw)*cos(roll)*yaw_ii +
sin(yaw)*sin(roll)*yaw_i*roll_i -
cos(yaw)*cos(roll)*roll_i^2 -
cos(yaw)*sin(roll)*roll_ii;
R23_ii = - sin(yaw)*sin(pitch)*cos(roll)*yaw_i^2 +
cos(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i -
cos(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i +
cos(yaw)*sin(pitch)*cos(roll)*yaw_ii +
cos(yaw)*cos(pitch)*cos(roll)*yaw_i*pitch_i -
sin(yaw)*sin(pitch)*cos(roll)*pitch_i^2 -
sin(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i +
```

```
    sin(yaw)*cos(pitch)*cos(roll)*pitch_ii -
    cos(yaw)*sin(pitch)*sin(roll)*yaw_i*roll_i -
    sin(yaw)*cos(pitch)*sin(roll)*pitch_i*roll_i -
    sin(yaw)*sin(pitch)*cos(roll)*roll_i^2 -
    sin(yaw)*sin(pitch)*sin(roll)*roll_ii +
    cos(yaw)*sin(roll)*yaw_i^2 +
    sin(yaw)*cos(roll)*yaw_i*roll_i +
    sin(yaw)*sin(roll)*yaw_ii +
    sin(yaw)*cos(roll)*yaw_i*roll_i +
    cos(yaw)*sin(roll)*roll_i^2 -
    cos(yaw)*cos(roll)*roll_ii;
    R31_ii = sin(pitch)*pitch_ii - cos(pitch)*pitch_ii;
    R32_ii = - cos(pitch)*sin(roll)*pitch_i^2 -
    sin(pitch)*cos(roll)*pitch_i*roll_i -
    sin(pitch)*sin(roll)*pitch_ii -
    sin(pitch)*cos(roll)*pitch_i*roll_i -
    cos(pitch)*sin(roll)*roll_i^2 +
    cos(pitch)*cos(roll)*roll_ii;
    R33_ii = - cos(pitch)*cos(roll)*pitch_i^2 +
    sin(pitch)*sin(roll)*pitch_i*roll_i -
    sin(pitch)*cos(roll)*pitch_ii +
    sin(pitch)*sin(roll)*pitch_i*roll_i -
    cos(pitch)*cos(roll)*roll_i^2 -
    cos(pitch)*sin(roll)*roll_ii;

    R_ii = [ R11_ii R12_ii R13_ii;
             R21_ii R22_ii R23_ii;
             R31_ii R32_ii R33_ii];

    x_base_ii = x_ii + R11_ii*dx + R12_ii*dy + R13_ii*dz;
    y_base_ii = y_ii + R21_ii*dx + R22_ii*dy + R23_ii*dz;
    z_base_ii = z_ii + R31_ii*dx + R32_ii*dy + R33_ii*dz;

%%% Rotational, inclination and cabler lifting speed %%%%%%%%%

    alpha_i = index_alpha_i*alpha_vel;
```

```
phi_i = index_phi_i*phi_vel;
l_i = index_l_i*l_vel;

%%%% Update angles %%%%%%

alpha_new = alpha + alpha_i*0.005;
phi_new = phi + phi_i*0.005;
l_new = l + l_i*0.005;


%%%% Angles acceleration %%%%%%%

alpha_ii = 0;
phi_ii = 0;
l_ii = 0;

%%%% Crane superior position, velocity and acceleration %%%%%%%

x_sup = x_base + L*cos(alpha+pitch)*cos(phi+yaw);
y_sup = y_base + L*cos(alpha+roll)*sin(phi+yaw);
z_sup = z_base + L*sin(alpha+roll+pitch);

x_sup_i = x_base_i - L*cos(alpha+pitch)*sin(phi+yaw)*(phi_i+yaw_i) -
L*sin(alpha+pitch)*cos(phi+yaw)*(alpha_i+pitch_i);
y_sup_i = y_base_i -
L*sin(alpha+roll)*sin(phi+yaw)*(alpha_i+roll_i) +
L*cos(alpha+roll)*cos(phi+yaw)*(phi_i+yaw_i);
z_sup_i = z_base_i +
L*cos(alpha+roll+pitch)*(alpha_i+roll_i+pitch_i);

x_sup_ii = x_base_ii -
L*cos(alpha+pitch)*cos(phi+yaw)*((phi_i+yaw_i)^2) -
L*cos(alpha+pitch)*sin(phi+yaw)*(phi_ii+yaw_ii) +
2*L*sin(alpha+pitch)*sin(phi+yaw)*(alpha_i+pitch_i)
*(phi_i+yaw_i) -
L*cos(alpha+pitch)*cos(phi+yaw)*((alpha_i+pitch_i)^2) -
```

```
L*sin(alpha+pitch)*cos(phi+yaw)*(alpha_ii+pitch_ii);
y_sup_ii = y_base_ii -
L*cos(alpha+roll)*sin(phi+yaw)*((alpha_i+roll_i)^2) -
L*sin(alpha+roll)*sin(phi+yaw)*(alpha_ii+roll_ii) -
2*L*sin(alpha+roll)*cos(phi+yaw)*(alpha_i+roll_i)*
(phi_i+yaw_i) -
L*cos(alpha+roll)*sin(phi+yaw)*((phi_i+yaw_i)^2) +
L*cos(alpha+roll)*cos(phi+yaw)*(phi_ii+yaw_ii);
z_sup_ii = z_base_ii -
L*sin(alpha+roll+pitch)*((alpha_i+roll_i+pitch_i)^2) +
L*cos(alpha+roll+pitch)*(alpha_ii+roll_ii+pitch_ii);

%%%% Wind speed and forces %%%%%%%

Vx = Vx_wind*sin(yaw) + Vy_wind*cos(yaw);
Vy = Vx_wind*cos(yaw) + Vy_wind*sin(yaw);
Vz = Vz_wind;

Fx_wind = rho*C*Ax*(Vx)^2;
Fy_wind = rho*C*Ay*(Vy)^2;
Fz_wind = rho*C*Az*(Vz)^2;

%%%% Differential equations %%%%%%%%%%

A_1 = 2*beta_22_i*beta_22 - 2*l_i/l;
A_2 = 2*beta_11_i*beta_11 - 2*l_i/l;

B = Fz_wind/(m*l) - z_sup_ii/l - g/l;

C_1 = Fx_wind/(m*l) - x_sup_ii/l;
C_2 = Fy_wind/(m*l) - y_sup_ii/l;

tspan = [time time+0.005];

beta_10 = [beta_11 beta_11_i];
beta_20 = [beta_22 beta_22_i];
```

```
%%% Non-linear equationa %%%%%%%

    [t_1,sol_1] = ode45(@(t,y) odefcn_b1(t,y,m,l,Fx_wind,Fy_wind,
    Fz_wind,     x_sup_ii,y_sup_ii,z_sup_ii,l_i,g,l_ii,beta_22,
    beta_22_i,beta_22_ii), tspan, beta_10);
    [t_2,sol_2] = ode45(@(t,y) odefcn_b2(t,y,m,l,Fx_wind,Fy_wind,
    Fz_wind,     x_sup_ii,y_sup_ii,z_sup_ii,l_i,g,l_ii,beta_11,
    beta_11_i,beta_11_ii), tspan, beta_20);

%%% Linearized equations %%%%%%%%%%

    %[t_1,sol_1] = ode45(@(t,y) odefcn(t,y,A_1,B,C_1), tspan, beta_10);
    %[t_2,sol_2] = ode45(@(t,y) odefcn(t,y,A_2,B,C_2), tspan, beta_20);

%%%% Equation solutions %%%%%%%%

    a_1 = size(sol_1);
    a_2 = size(sol_2);

    b1 = sol_1(a_1(1));
    b_1_i = sol_1(2*a_1(1));

    b2 = sol_2(a_2(1));
    b_2_i = sol_2(2*a_2(1));

%%%% Load position %%%%%%%

    beta_1_ii = (2*b_2_i*b2 - 2*l_i/l)*b_1_i +
    (Fz_wind/(m*l) - z_sup_ii/l - g/l)*b1 + Fx_wind/(m*l) - x_sup_ii/l;
    beta_2_ii = (2*b_1_i*b1 - 2*l_i/l)*b_2_i +
    (Fz_wind/(m*l) - z_sup_ii/l - g/l)*b2 + Fy_wind/(m*l) - y_sup_ii/l;

    x_load = x_sup + l*sin(b1)*cos(b2);
    y_load = y_sup + l*cos(b1)*sin(b2);
    z_load = z_sup - l*cos(b1)*cos(b2);
```

30

```
x_load_i = x_sup_i + l*cos(b1)*cos(b2)*b_1_i -
l*sin(b1)*sin(b2)*b_2_i + l_i*sin(b1)*cos(b2);
y_load_i = y_sup_i - l*sin(b1)*sin(b2)*b_1_i +
l*cos(b1)*cos(b2)*b_2_i + l_i*cos(b1)*sin(b2);
z_load_i = z_sup_i + l*sin(b1)*cos(b2)*b_1_i +
l*cos(b1)*sin(b2)*b_2_i - l_i*cos(b1)*cos(b2);

x_load_ii = x_sup_ii +
l_i*cos(b1)*cos(b2)*b_1_i -
l*sin(b1)*cos(b2)*(b_1_i^2) -
l*cos(b1)*sin(b2)*b_1_i*b_2_i +
l*cos(b1)*cos(b2)*beta_1_ii -
l_i*sin(b1)*sin(b2)*(b_2_i^2) -
l*cos(b1)*sin(b2)*b_1_i*b_2_i -
l*sin(b1)*cos(b2)*(b_2_i^2) -
l*sin(b1)*sin(b2)*beta_2_ii +
l_i*cos(b1)*cos(b2)*b_1_i -
l_i*sin(b1)*sin(b2)*b_2_i;
y_load_ii = y_sup_ii -
l_i*sin(b1)*sin(b2)*b_1_i -
l*cos(b1)*sin(b2)*(b_1_i^2) -
l*sin(b1)*cos(b2)*b_1_i*b_2_i -
l*sin(b1)*sin(b2)*beta_1_ii +
l_i*cos(b1)*cos(b2)*(b_2_i^2) -
l*sin(b1)*cos(b2)*b_1_i*b_2_i -
l*cos(b1)*sin(b2)*(b_2_i^2) +
l*cos(b1)*cos(b2)*beta_2_ii -
l_i*sin(b1)*sin(b2)*b_1_i +
l_i*cos(b1)*cos(b2)*b_2_i;
z_load_ii = z_sup_ii +
l_i*sin(b1)*cos(b2)*b_1_i +
l*cos(b1)*cos(b2)*(b_1_i^2) -
l*sin(b1)*sin(b2)*b_1_i*b_2_i +
l*sin(b1)*cos(b2)*beta_1_ii -
l_i*cos(b1)*sin(b2)*(b_2_i^2) +
```

```
    l*sin(b1)*sin(b2)*b_1_i*b_2_i -
    l*cos(b1)*cos(b2)*(b_2_i^2) -
    l*cos(b1)*sin(b2)*beta_2_ii -
    l_i*sin(b1)*cos(b2)*b_1_i -
    l_i*cos(b1)*sin(b2)*b_2_i;

%%% Load velocity %%%%%%

    load_i = ((x_load_i^2) + (y_load_i^2) + (z_load_i^2))^0.5;

%%% Cable tension %%%%%%%

    a_rel_1 = (z_sup_ii - z_load_ii)*cos(b1)*cos(b2) +
    (y_sup_ii - y_load_ii)*cos(b1)*sin(b2) +
    (x_sup_ii - x_load_ii)*sin(b1)*cos(b2);
    a_rel_2 = z_sup_ii*cos(b1)*cos(b2) +
    y_sup_ii*cos(b1)*sin(b2) + x_sup_ii*sin(b1)*cos(b2);

    Tension = m*(((load_i^2)/l) + g*cos(b1)*cos(b2));
    Tension_1 = m*(((load_i^2)/l) + g*cos(b1)*cos(b2) +
    a_rel_1);
    Tension_2 = m*(((load_i^2)/l) + g*cos(b1)*cos(b2) +
    a_rel_2);

%%%% Reactive forces and moments %%%%%%%

    Fx = Tension*sin(b1)*cos(b2);
    Fy = Tension*cos(b1)*sin(b2);
    Fz = -Tension*cos(b1)*cos(b2);
    Mx = -Tension*cos(b1)*cos(b2)*L*cos(alpha)*sin(phi) -
    Tension*cos(b1)*sin(b2)*L*sin(alpha);
    My = Tension*cos(b1)*cos(b2)*L*cos(alpha)*cos(phi) +
    Tension*sin(b1)*cos(b2)*L*sin(alpha);
    Mz = Tension*sin(b1)*cos(b2)*L*cos(alpha)*sin(phi) +
    Tension*cos(b1)*sin(b2)*L*cos(alpha)*cos(phi);
```

```
%%%%% Update values %%%%%%

    beta_111 = b1;
    beta_111_i = b_1_i;
    beta_111_ii = beta_1_ii;

    beta_222 = b2;
    beta_222_i = b_2_i;
    beta_222_ii = beta_2_ii;

    alpha_new = alpha_new*180/pi;
    phi_new = phi_new*180/pi;

%%%% Global to local positions %%%%%%

    x_sup_local =  L*cos(alpha)*cos(phi);
    y_sup_local =  L*cos(alpha)*sin(phi);
    z_sup_local =  L*sin(alpha);

    x_load_local = x_sup_local + l*sin(b1+pitch)*cos(-b2+roll);
    y_load_local = y_sup_local + l*cos(b1+pitch)*sin(-b2+roll);
    z_load_local = z_sup_local - l*cos(b1+pitch)*cos(-b2+roll);

    x_inf_local = dx + x_load_local + 2.55*(cos(phi)-1);
    y_inf_local = dy + y_load_local + 2.55*sin(phi);
    z_inf_local = dz + z_load_local - 6.25;

    x_load_local_z = cos(phi)*x_load_local +
    sin(phi)*y_load_local;
    y_load_local_z = -sin(phi)*x_load_local +
    cos(phi)*y_load_local;
    z_load_local_z = z_load_local;

    x_load_local_zy = cos(-alpha)*x_load_local_z -
    sin(-alpha)*z_load_local_z;
    y_load_local_zy = y_load_local_z;
```

```
    z_load_local_zy = sin(-alpha)*x_load_local_z +
    cos(-alpha)*z_load_local_z;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.3 Função Principal do Simulador(hybrid.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                        Main Program
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: October, 2018
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%% Program Parameters %%%%%%%%%%%%%%%%%%%%

%%% Vessel ID %%%
id =4;

%%% Database IP %%%
DB_IP = 'mongodb://172.16.11.10:27017';

%%% Server IP %%%
IP = '172.16.11.32';

%%%%% Initial values %%%%%%%%%%%

OnOff = 0;
```

```
set_i = 30;
l = 30;
alpha = 63;
phi = 0;
beta_11=0;
beta_11_i=0;
beta_11_ii = 0;
beta_22=0;
beta_22_i=0;
beta_22_ii = 0;
index_alpha_i = 0;
index_phi_i = 0;
index_l_i = 0;
Vx_wind = 0;
Vy_wind = 0;
Vz_wind = 0;
time = 0;
limit = 0;
l_limit = 10;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Buzz Connection %%%%%%%
try

%%%% Inicialization %%%%%%%

        smh.buzz_java.initialize_simco();

%%%%% Database connection %%%%%%

        ds = smh.buzz_java.create_bson_data_source(DB_IP, 'smh');

%%%% Session inicialization %%%%%

        sl = smh.buzz_java.create_bson_serializer(ds);
```

```
        global s;
        s = smh.buzz_java.join_simco_session('12', sl);

%%%% Buzz inicialization %%%%%%%

        subscriber = tpn.io.BuzzSubscriber(s);
        hBuzzsubscriber = handle(subscriber, 'CallbackProperties');
        set(hBuzzsubscriber,
        'networkConnectionChangedEventCallback',
        @(h,e)network_connection(h,e));

%%%% Server connection %%%%%

        s.connect(IP);

%%%%% Current connection state %%%%

        old = s.get_current_state();
        disp(s.get_current_state());

%%% Connection running %%%%%%%

        while (s.get_current_state() ~= smh.state_type.RUNNING)
            new = s.get_current_state();
            if ~strcmp(new,old)
                disp(s.get_current_state());
                old = new;
            end
        end
        disp('current state = ')
        disp(s.get_current_state());

%%%%%% Getting vessels %%%%%%

        v = s.get_vessels();
```

```
%%%%% Error exceptions %%%%%%

catch e
        disp('Exception: ')
        disp(e.message)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%% NMEA Conncetion %%%%%%%%%%%%%%%%%%%%%%%%

%%%% Filters %%%%%
filterCell = {'HLWIN';'LEVER'};

%%%% server connection %%%%%%%

connHandle = nmeaConnect(IP, filterCell);

%%%% NMEA messages %%%%%%%

while(s.get_current_state() == smh.state_type.RUNNING)

%%%% Selecting messages %%%%%%

      data = char(connHandle.get_msg());
      [type,value] = NMEA_get_data(data);

      if (strcmp(type,'cable'))
          index_l_i = value;
      elseif (strcmp(type,'inclination'))
          index_alpha_i = value;
      elseif (strcmp(type,'rotation'))
          index_phi_i = value;
      elseif (strcmp(type,'control'))
          if (value >= 0)
              OnOff = 1;
```

```
        else
            OnOff = 0;
        end
    elseif (strcmp(type,'wind'))
        Vx_wind = value(1);
        Vy_wind = value(2);
        Vz_wind = value(3);
    end

%%%%%%%% Create object vessel %%%%%%%%

        for i = 0:v.size()-1
            ves = v.get(i);
            ID = str2double(ves.get_id());
            if (ID == id)
                break
            end
        end

%%%%%%% Update vessel %%%%%%%

        s.sync(ves);

%%%%%%% GET BUZZ POSITION %%%%%%%

%%%%% Linear States %%%%%%%%%%

        x = ves.get_linear_position().get(0);
        y = ves.get_linear_position().get(1);
        z = ves.get_linear_position().get(2);
        x_i = ves.get_linear_velocity().get(0);
        y_i = ves.get_linear_velocity().get(1);
        z_i = ves.get_linear_velocity().get(2);
        x_ii = ves.get_linear_acceleration().get(0);
        y_ii = ves.get_linear_acceleration().get(1);
        z_ii = ves.get_linear_acceleration().get(2);
```

```
%%%%% Angular states %%%%%

        roll = ves.get_angular_position().get(0);
        pitch = ves.get_angular_position().get(1);
        yaw = ves.get_angular_position().get(2);
        roll_i = ves.get_angular_velocity().get(0);
        pitch_i = ves.get_angular_velocity().get(1);
        yaw_i = ves.get_angular_velocity().get(2);
        roll_ii = ves.get_angular_acceleration().get(0);
        pitch_ii = ves.get_angular_acceleration().get(1);
        yaw_ii = ves.get_angular_acceleration().get(2);

%%%%% Time %%%%%%

        time_cicle = s.get_current_time_step();
        time_increment = s.get_current_time_increment();

%%%%%% Crane %%%%

        [l_new,alpha_new,phi_new,beta_111,beta_111_i,
        beta_111_ii,beta_222,beta_222_i,beta_222_ii,
        Tension,Fx,Fy,Fz,Mx,My,Mz,x_load,y_load,
        z_load,x_inf,y_inf,z_inf]
        =funcao_guindaste_TPN_opt(time,l,alpha,phi,
        beta_11,beta_11_i,beta_11_ii,beta_22,
        beta_22_i,beta_22_ii,x,y,z,x_i,y_i,z_i,x_ii,
        y_ii,z_ii,roll,pitch,yaw,
        roll_i,pitch_i,yaw_i,roll_ii,pitch_ii,
        yaw_ii,index_alpha_i,index_phi_i,
        index_l_i,Vx_wind,Vy_wind,Vz_wind);

%%%%%%%% Control %%%%%%%%%%

        if OnOff == 1
            index_l_i = controller(set_i,z_inf,2);
```

```
        else
            set_i = z_inf;
        end

%%%%%%% Cable limitation %%%%%%%

        if (l_new <= l_limit)
            l_new = l_limit;
        end

%%%%% Update values %%%%%%%

        beta_11=beta_111;
        beta_11_i=beta_111_i;
        beta_11_ii=beta_111_ii;
        beta_22=beta_222;
        beta_22_i=beta_222_i;
        beta_22_ii=beta_222_ii;
        l = l_new;
        phi = phi_new;
        alpha = alpha_new;

        time = time + 0.005;

%%%% Send NMEA messages %%%%%%%

        msg_1 = ['HLCJT,1001,0,',num2str(x_load),',',
        num2str(y_load),',',num2str(z_load),',0,'
        ,num2str(alpha),',0'];
        msg_2 = ['HLCJT,1001,1,2.55,0,3.7,0,',num2str(-alpha),
        ',0'];
        msg_3 = ['HLCJT,1001,2,74.4,-16.67,23.2,0,0,',num2str(phi)];
        connHandle.set_msg(msg_1);
        connHandle.set_msg(msg_2);
        connHandle.set_msg(msg_3);
        msg = ['HLCOT,1,',num2str(x_inf),',',
```

```
        num2str(y_inf),',',num2str(z_inf),',0,0,0'];
        connHandle.set_msg(msg);
        msg_force = ['XFORC,4,111,',num2str(Fx),',',num2str(Fy),',',
        num2str(Fz),',',num2str(Mx),',',num2str(My),',',
        num2str(Mz),',76.95,-16.67,26.9,',
        num2str(time_cicle),',',num2str(time_increment)];
        connHandle.set_msg(msg_force);
        msg_tension =
        ['CRINF,1001,Tension,',num2str(Tension)];
        connHandle.set_msg(msg_tension);

end

%%%% Disconnect connection and session %%%%%%%

s.disconnect(); % Desconectar conexo
pause(1);
subscriber.terminate(s)
s.terminate()
clear(s)
smh.buzz_java.terminate_simco();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.4 Função de Conexão à Rede(network_connection.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    Network Connection Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: September, 2018
%
% Input:
%
```

```
% Output: Display (Provides connection status |
% Connected or Disconnected)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Function %%%%
function [] = network_connection(hObject, ev)

%%% Connection Variable %%%

    global s;

%%%% Connection Status %%%

    if s.is_connected()
        disp('Connected!\n')
    else
        disp('Disconnected!\n')
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.5 Função de Obtenção de Dados do NMEA (nmea_get_data.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    NMEA Data Collection Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: October, 2018
```

```
%
% Input: - set (set value for the measured value)
%        - present (present value of the measured value)
%
% Output: - type (type of the message)
%         - value (number associated with type)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Function %%%%%

function [type,value] = NMEA_get_data(data)

%%%% Split Message %%%

        splitData = strsplit2(data,',');

%%%%% Identify Message %%%%%%

        switch splitData{1}

%%%%% Wind Data %%%%%%%%%%%%

            case 'HLWIN'

                type = 'wind';
                value = [ str2double(splitData(2))
                str2double(splitData(3)) str2double(splitData(4))];

%%%%%% Cable Data %%%%%%%%%%

            case 'LEVER'

                if (strcmp(splitData(2),'crane_1001_hook_height'))
                    type = 'cable';
                elseif (strcmp(splitData(2),'crane_1001_yaw'))
```

```
                type = 'rotation';
            elseif (strcmp(splitData(2),'crane_1001_pitch'))
                type = 'inclination';
            elseif (strcmp(splitData(2),'crane_1001_auto_heave'))
                type = 'control';
            else
                type = 'Unknown';
            end
            value = str2double(splitData(3));

%%%%%% Unknown Message %%%%%%%%

        otherwise
            type = 'Unknown';
            value = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.6 Função de Conexão do NMEA(nmeaConnect.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                   NMEA Connection Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva (adapted from SMH POLI USP
% Date: September, 2018
```

```
%
% Input: - IP Adress (Server   s IP Adress)
%        - varargin (NMEA classification filters)
%
% Output:  connHandle (NMEA connection)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%% Function %%%%%%%%%%

function connHandle = nmeaConnect(ipAddress, varargin)

% nmeaConnect(ipAddress,filterCell): Connects to an NMEA chat server
% addressed by ipAddress and adds filters to it defined by cell array
% filterCell

%%%% Create de connection Handle %%%%

connHandle = nmea.nmea_client;

%%% Connects to the chat server addres %%%

connHandle.connect(ipAddress);

%%% If there are filters, adds it to the handle %%%

if ~isempty(varargin)
    filterCell = varargin{1};
    for k = 1:length(filterCell)
        connHandle.add_filter(filterCell{k});
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

%%%

6.7 Função das Equações Diferenciais Linearizadas (odefcn.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    Linearized  Differential  Equation  Function
%
% Authors:  Julio  Arantes  and  Pedro  H.  D.  Silva
% Date:  October,  2018
%
% Input:  -  t  (time)
%          -  y  (differential  equation  variable)
%          -  A
%          -  B
%          -  C
%
% Output:   dydt  (solution  of  the  differential  equation)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Function %%%%%

function  dydt  =  odefcn(t,y,A,B,C)

%%%%% Initial  Values %%%%%%

    dydt  =  zeros(2,1);

%%%%% Diferential  Equations %%%%%%

    dydt(1)  =  y(2);
    dydt(2)  =  A*y(2)  +  B*y(1)  +  C;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.8 Função das Equações Diferenciais de β_1 (odefcn_b1.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                 Differential Equation for Beta 1 Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: October, 2018
%
% Input: - t (time)
%        - y (differential equation variable)
%        - m (load mass)
%        - l (cable length)
%        - Fx_wind (wind force on x-axis)
%        - Fy_wind (wind force on y-axis)
%        - Fz_wind (wind force on z-axis)
%        - x_sup_ii (acceleration at the top of the crane on x-axis)
%        - y_sup_ii (acceleration at the top of the crane on y-axis)
%        - z_sup_ii (acceleration at the top of the crane on z-axis)
%        - l_i (cable lifting speed)
%        - g (acceleration of gravity)
%        - l_ii (cabkle lifting acceleration)
%        - beta_22 (Beta 2 value)
%        - beta_22_i (Beta 2 first derivative value)
%        - beta_22_ii (Beta 2 second derivative value)
%
% Output: dydt (solution of the differential equation)
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% Function %%%%%%

function dydt = odefcn_b1(t,y,m,l,Fx_vento,Fy_vento,Fz_vento,x_sup_ii,
y_sup_ii,z_sup_ii,l_i,g,l_ii,beta_22,beta_22_i,beta_22_ii)

%%%% Initial Values %%%%%%

    dydt = zeros(2,1);

%%%% Diferential Equations %%%%%%

    dydt(1) = y(2);
    dydt(2) = (((4*Fx_vento*cos(y(1))*cos(beta_22))/m*l) -
    ((4*Fy_vento*sin(y(1))*sin(beta_22))/m*l) +
    ((4*Fz_vento*sin(y(1))*cos(beta_22))/m*l) -
    ((4*x_sup_ii*cos(y(1))*cos(beta_22))/l) +
    ((4*y_sup_ii*sin(y(1))*sin(beta_22))/l) -
    ((4*z_sup_ii*sin(y(1))*cos(beta_22))/l) -
    (2*l_i*y(2)/l)*(3 - cos(2*y(1)) + cos(2*beta_22) +
    cos(2*y(1))*cos(2*beta_22)) -
    (6*l_i*beta_22_i/l)*sin(2*y(1))*sin(2*beta_22) -
    (y(2)^2)*(sin(2*y(1)) - sin(2*y(1))*cos(2*beta_22)) -
    (beta_22_i^2)*(sin(2*y(1)) -
    sin(2*y(1))*cos(2*beta_22)) +
    2*y(2)*beta_22_i*(sin(2*beta_22) +
    cos(2*y(1))*sin(2*beta_22)) +
    (beta_22_ii^2)*sin(2*y(1))*sin(2*beta_22) -
    (2*(l_i^2)/(l^2))*(sin(2*y(1)) -
    sin(2*y(1))*cos(2*beta_22)) - (2*l_ii/l)*(sin(2*y(1)) -
    sin(2*y(1))*cos(2*beta_22)) -
    (4*g/l)*sin(y(1))*cos(beta_22))/(3 - cos(2*y(1)) +
    cos(2*beta_22) + cos(2*y(1))*cos(2*beta_22));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.9 Função das Equações Diferenciais de β_2 (odefcn_b2.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                Differential Equation for Beta 2 Function
%
%
% Authors: Julio Arantes and Pedro H. D. Silva
% Date: October, 2018
%
% Input: - t (time)
%        - y (differential equation variable)
%        - m (load mass)
%        - l (cable length)
%        - Fx_wind (wind force on x-axis)
%        - Fy_wind (wind force on y-axis)
%        - Fz_wind (wind force on z-axis)
%        - x_sup_ii (acceleration at the top of the crane on x-axis)
%        - y_sup_ii (acceleration at the top of the crane on y-axis)
%        - z_sup_ii (acceleration at the top of the crane on z-axis)
%        - l_i (cable lifting speed)
%        - g (acceleration of gravity)
%        - l_ii (cabkle lifting acceleration)
%        - beta_11 (Beta 1 value)
%        - beta_11_i (Beta 1 first derivative value)
%        - beta_11_ii (Beta 1 second derivative value)
%
% Output:  dydt (solution of the differential equation)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%% Function %%%%%

function dydt = odefcn_b2(t,y,m,l,Fx_vento,Fy_vento,Fz_vento,x_sup_ii,
y_sup_ii,z_sup_ii,l_i,g,l_ii,beta_11,beta_11_i,beta_11_ii)

%%% Initial Values %%%%

    dydt = zeros(2,1);

%%% Differential Equations %%%%%%

    dydt(1) = y(2);
    dydt(2) = (((-4*Fx_vento*sin(beta_11)*sin(y(1)))/m*l) +
    ((4*Fy_vento*cos(beta_11)*cos(y(1)))/m*l) +
    ((4*Fz_vento*cos(beta_11)*sin(y(1)))/m*l) +
    ((4*x_sup_ii*sin(beta_11)*sin(y(1)))/l) -
    ((4*y_sup_ii*cos(beta_11)*cos(y(1)))/l) -
    ((4*z_sup_ii*cos(beta_11)*sin(y(1)))/l) -
    (2*l_i*y(2)/l)*(3 + cos(2*beta_11) - cos(2*y(1)) +
    cos(2*beta_11)*cos(2*y(1))) -
    (6*l_i*beta_11_i/l)*sin(2*beta_11)*sin(2*y(1)) -
    (beta_11_i^2)*(sin(2*y(1)) -
    cos(2*beta_11)*sin(2*y(1))) - (y(2)^2)*(sin(2*y(1)) -
    cos(2*beta_11)*sin(2*y(1))) +
    2*y(2)*beta_11_i*(sin(2*beta_11) +
    sin(2*beta_11)*cos(2*y(1))) +
    (beta_11_ii^2)*sin(2*beta_11)*cos(2*y(1)) -
    (2*(l_i^2)/(l^2))*(sin(2*y(1)) -
    cos(2*beta_11)*sin(2*y(1))) - (2*l_ii/l)*(sin(2*y(1)) -
    cos(2*beta_11)*sin(2*y(1))) -
    (4*g/l)*cos(beta_11)*sin(y(1)))/(3 + cos(2*beta_11) -
    cos(2*y(1)) + cos(2*beta_11)*cos(2*y(1)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

50

%%