MODELAGEM PARAMÉTRICA PARA ANÁLISE ESTRUTURAL DE SEÇÃO MESTRA TÍPICA DE PETROLEIROS COM SOFTWARE ABAQUS CAE

Israel Cubas Pereira
Orientador: Prof. Dr. Diego Felipe Sarzosa Burgos

Dezembro de 2022

Resumo

O uso de simulações em elementos finitos na concepção de arranjos estruturais de embarcações petroleiras têm-se cada vez mais intensificado nos último anos. Devido à importância estrutural da seção mestra do navio na resistência aos esforços externos (1), é fundamental que o projeto desse setor resistente seja auxiliado por computador. O presente estudo visa a concepção de um modelo paramétrico de seção meia-nau típica de Very Large Crude Carriers (VLCC) que possa ser integrado ao *software* Abaqus CAE para posterior simulação estrutural - eliminando assim a necessidade de remodelagem para qualquer alteração do arranjo de reforços escolhidos. Após o modelo estar finalizado, foram realizados ensaios paramétricos com uma seção transversal constante, variando apenas o espaçamento entre reforçadores longitudinais da embarcação. Com os dados obtidos - massa da estrutura, tensão máxima - pôde-se exemplificar a capacidade de aplicação do modelo concebido em auxiliar tomadas de decisão quanto a um arranjo estrutural em projeto.

Palavras-chaves: Simulação Estrutural, Método dos Elementos Finitos, Modelagem Paramétrica, Programação em Python Abaqus, Programação Python, Abaqus CAE.

1 Introdução

1.1 Contextualização

Em projetos de embarcações de grande porte, tais quais petroleiros, a parcela de custos materiais em aço é, segundo (2), ao redor de 10%. Dessa forma, pode-se dizer que é fundamental que o projeto estrutural da embarcação tem uma forte correlação com os custos totais do projeto, implicando que, qualquer planejamento inadequado quanto ao arranjo estrutural, pode colocar o projeto como um todo em risco.

Navios petroleiros - **Figura 1** - tem como avaliação, dentro das iterações de projeto realizadas, a concepção de um arranjo de elementos estruturais na seção mestra do navio capaz de ponderar entre resistência e custos - tais custos sendo tanto de materiais quanto de fabricação. A estrutura de diversos tipos de embarcação sempre são sujeitas a uma ação conjunta de carregamentos, tais quais momentos induzidos pelo oceano na estrutura, pressão hidrostática externa e o peso da carga internamente (3). A concepção de uma seção estrutural otimizada para uma embarcação implicaria na necessidade de iterativamente projetar e simular numericamente diversos arranjos, a

fim de encontrar o que melhor atende os carregamentos impostos ao navio e o fator de segurança esperado para a embarcação.

Figura 1 – Embarcação Petroleira - exemplo.

Fonte: Nipon Yusei Kaisha NYK Line - Arquivo online.

Dada a complexidade do projeto estrutural de uma embarcação petroleira, a modelagem e simulação de diversos arranjos estruturais, a fim de encontrar o mais otimizado, acaba sendo um processo temporalmente muito custoso. Atualmente a concepção de modelos para simulação estrutural via elementos finitos é fundamental para a validação de um arranjo estrutural proposto em um projeto de embarcação. O software de simulação Abaqus CAE, comercializado atualmente pela companhia francesa *Dassault Systemes S.A - (4)*, além de contar com uma interface de modelagem tridimensional, permite aos usuários a construção de modelos diretamente via *Script* em linguagem de programação *Python* (5) .

Com essa versatilidade, a criação de modelos paramétricos para simulação em elementos finitos usando o software mencionado torna-se uma ferramenta poderosa. Usando essa abordagem é possível configurar variáveis no *Script* e executar diversas simulações iterativamente sem a necessidade de qualquer intervenção do usuário em termos de remodelagem. Como tratado acima, fica evidente a importância da concepção de um modelo paramétrico de seção transversal de um navio - **Figura 2**, uma vez que a partir disso pode-se realizar múltiplos ensaios estruturais sem a necessidade de remodelagem constante.

Figura 2 – Embarcação petroleira docada - arranjo estrutural interno visível.

Fonte: Ahads journal - Marine Insight - Arquivo Online.

1.2 Relevância

A resistência longitudinal da seção mestra é a resistência mais importante para garantir um comportamento estrutural seguro da embarcação (1, 6). As normas de sociedades classificadoras - (7, 8) - podem ser utilizadas para estimar a resistência estrutural, entretanto, essas regras são somente aplicáveis a embarcações de casco simplificado e não levam em consideração as tensões provenientes das conexões entre elementos longitudinais e transversais (9).

Para realização de uma avaliação estrutural mais robusta é inevitavelmente necessário recorrer a simulações estruturais, quase sempre utilizando o **Método dos Elementos Finitos (MEF)**. Dessa forma, a criação de uma modelo paramétrico que já está no ambiente dedicado à simulações via MEF é de grande utilidade, uma vez que poupa tanto o tempo de construção da geometria - que é o mais custoso - quanto o tempo de configuração do modelo para a simulação em si.

1.3 Objetivo

O presente trabalho em desenvolvimento objetiva conceber um modelo paramétrico de seção mestra típica de embarcações petroleiras. É esperado que esse modelo seja funcional e possa ser executado parametricamente para diversos casos de variação geométrica. Inicialmente, após avaliação de diversas possíveis geometrias a serem utilizadas - (10, 11), foi escolhido o arranjo comumente usado por VLCCs, mostrado na **Figura 3**.

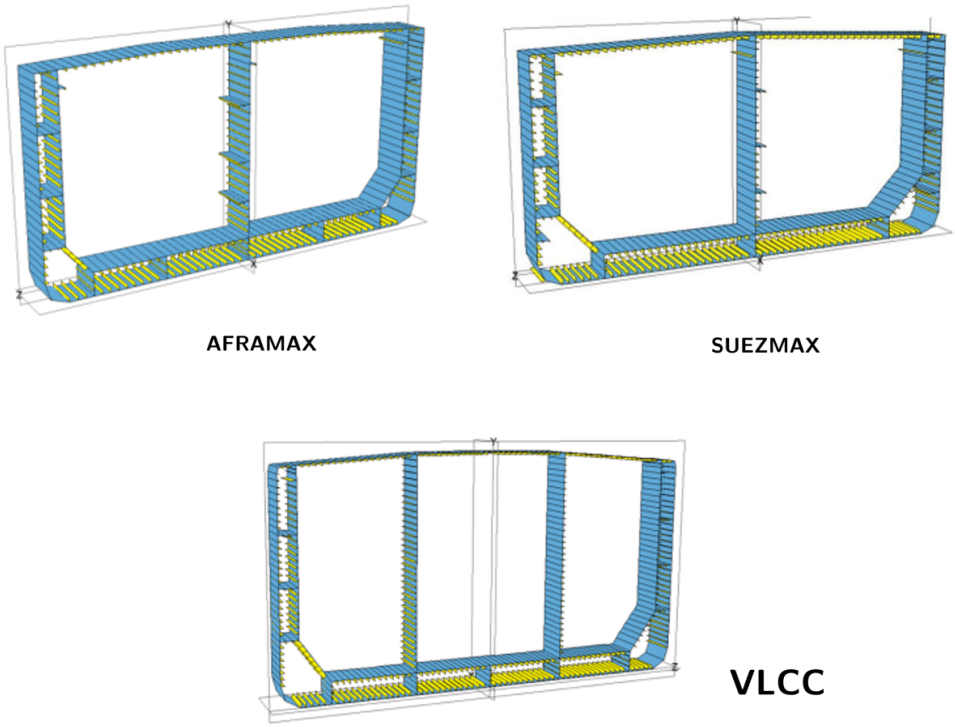


Figura 3 – Possíveis arranjos estruturais de seção mestra de petroleiros - VLCC foi o escolhido.

Fonte: Andric (10).

2 Desenvolvimento do Modelo

2.1 Estratégia de Concepção

No desenvolvimento do modelo final, devido à novas demandas - relacionadas principalmente na possibilidade de alteração do número de elementos reforçadores em cada trecho da geometria - a estratégia de modelagem foi repensada.

Ao invés de modelar individualmente cada um dos elementos longitudinais da estrutura e posteriormente adicioná-los à uma montagem geral, foi escolhida a opção de modelar conjuntamente toda a estrutura longitudinal - já contando com todos os elementos reforçadores - e em seguida realizar uma extrusão desse perfil - como mostra a **Figura 4** abaixo.

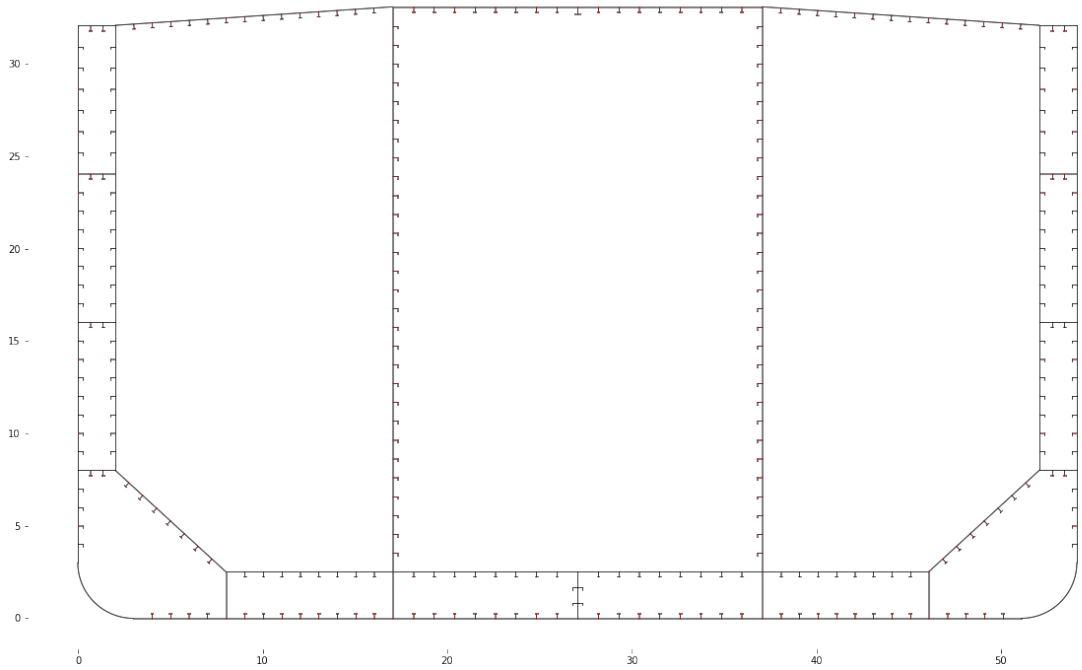


Figura 4 – Exemplo de seção transversal.

Fonte: Elaboração Própria.

Com essa seção longitudinal extrudada e contando com os elementos reforçadores transversais (gigantes (hastilha + caverna + vau) e anteparas) é realizada a montagem do corpo paralelo médio da embarcação - o corpo paralelo médio corresponde ao comprimento do navio cuja seção transversal se mantém constante. A **Figura 5** mostra o esquema de montagem a ser aplicado.

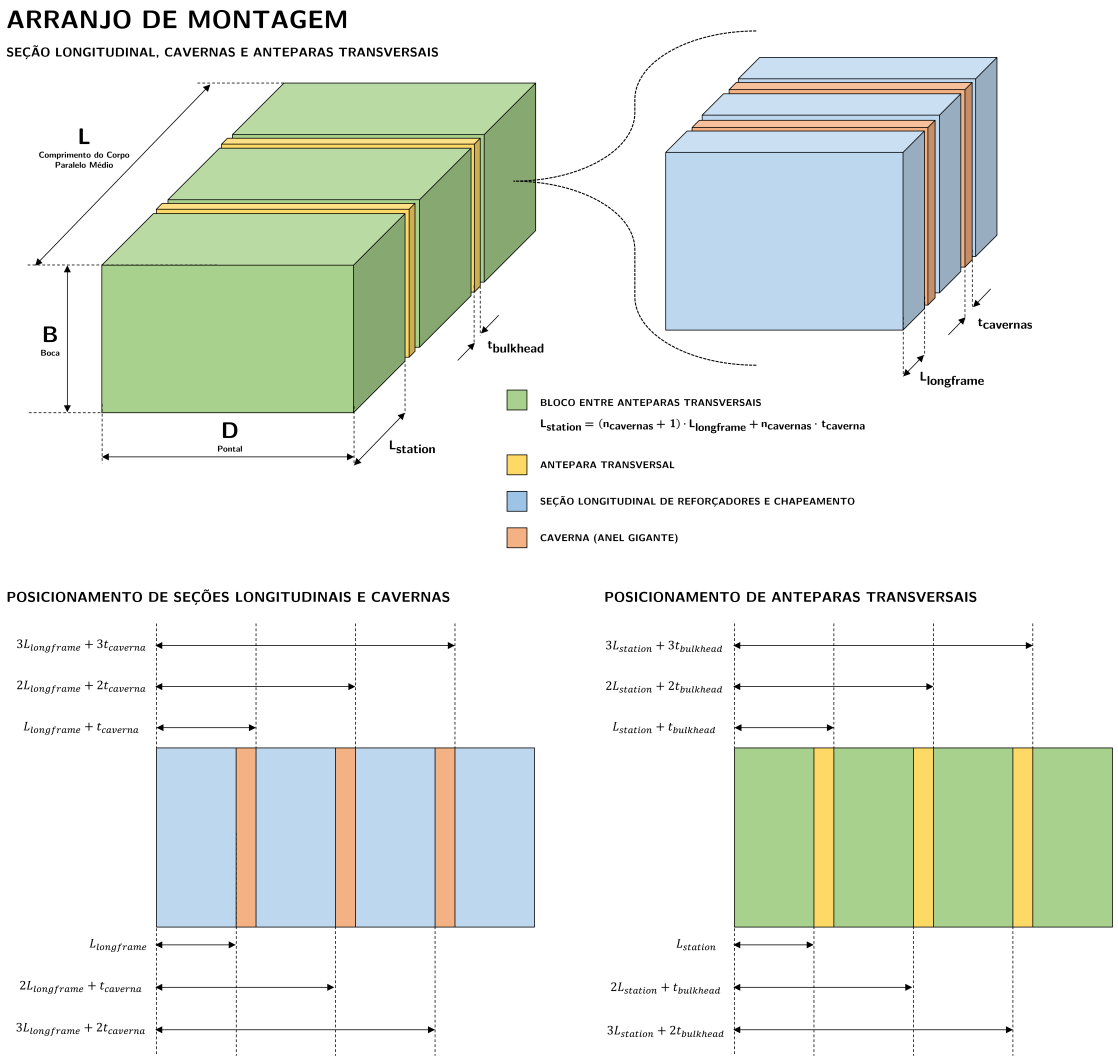


Figura 5 – Esquemático mostrando a montagem do arranjo concebido.

Fonte: Elaboração Própria.

Com essa nova estratégia toda a definição da geometria será concebida diretamente em códigos *Python* e posteriormente, após exportação dos pontos definidores, será desenhada no ambiente de *sketch* do Abaqus fazendo uso de segmentos de reta definidos por dois pontos. A seguir serão detalhados os passos necessários para a concepção da geometria mostrada acima na **Figura 4**.

2.2 Funções para Criação da Geometria - **Apêndice 6**

2.2.1 Space Division

Essa função recebe como parâmetros dois pontos, pA e pB, no formato de tuplas do *Python*, ou seja, pA = (x_A, y_A), sendo x_A e y_A números em ponto flutuante e o parâmetro n se refere à quantidade de elementos reforçadores a serem aplicados entre os dois pontos - pA e pB, sendo n um número inteiro. Ao final, a respectiva função retorna uma lista de pares ordenados x e y para cada um dos reforçadores a ser posicionado, cada par estando no formato de tupla. A **Figura 6** abaixo mostra um diagrama esquemático que foi utilizado para construção da função descrita.

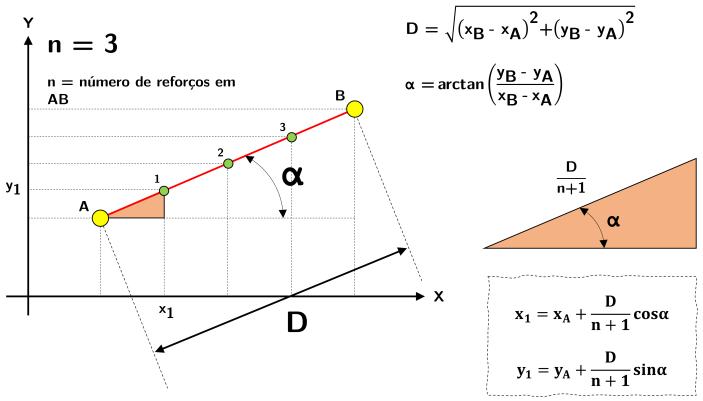


Figura 6 – Diagrama usado na construção da função Space Division.

Fonte: Elaboração Própria.

2.2.2 T Reinforcement Points

Função responsável por retornar tanto os pontos definidores da geometria do reforçador T - definido pelos parâmetros de entrada a, b, t e h, respectivamente alma, flange, espessura da alma e espessura do flange - quanto os segmentos de reta que serão posteriormente utilizados na partição do reforçador para construção da malha em elementos finitos do modelo. Particionar um sólido trata-se de realizar cortes nesse corpo para alcançar poliedros regulares.

Além dos parâmetros já mencionados que são entradas para a respectiva função, temos ainda que informar a espessura da base onde ele será posicionado - t_{base}, o par ordenado (x,y) referente ao local onde o elemento será posicionado. Ainda é necessário informar o ângulo de inclinação do elemento - parâmetro *theta1*. Para melhor entendimento da dedução geométrica realizada, é mostrado na **Figura 7** abaixo um diagrama detalhando a sequência de pontos definidos para concepção da respectiva função.

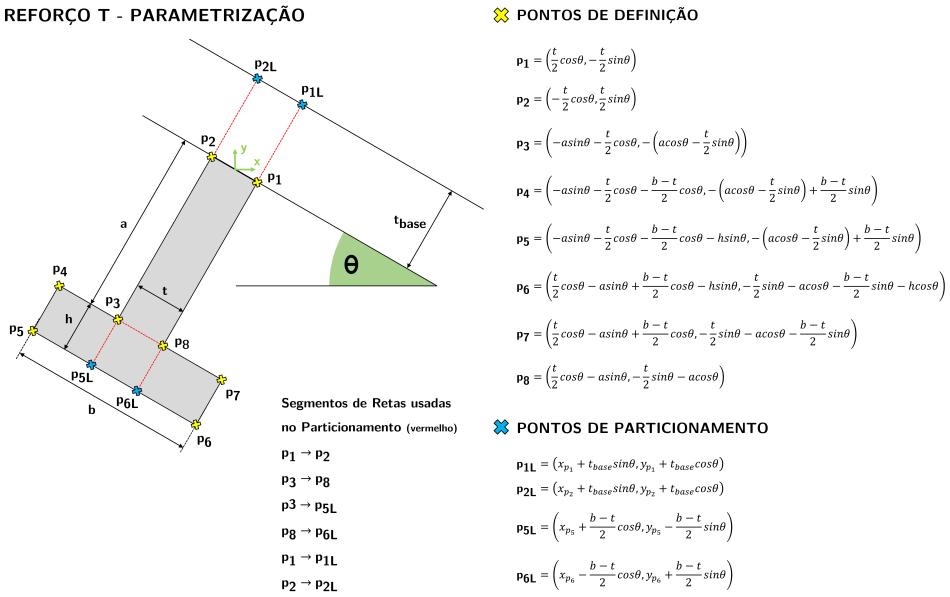


Figura 7 – Diagrama usado na construção da função T Reinforcement Points.

Fonte: Elaboração Própria.

Na **Figura 8** abaixo é mostrado um exemplo de uso, plotando um reforçador T com os seguintes parâmetros: a = 10, b = 10, t = 2, h = 2, t_{base} = 2 e inclinado a -45º com origem configurada no ponto (1,1) - ponto azul na respectiva figura. Nessa figura podemos ver em vermelho os segmentos de reta que serão posteriormente usados para particionamento do reforçador para melhor adequação da malha de elementos finitos.

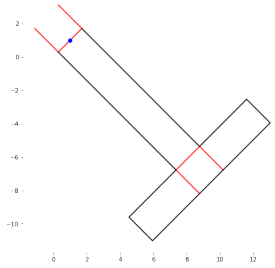


Figura 8 – Exemplo de Aplicação da função T Reinforcement Points.

Fonte: Elaboração Própria.

2.2.3 L1 e L2 Reinforcement Points

As funções L1 e L2 Reinforcement Points são análogas à função para o reforçamento T apresentada acima, bem como a explicação para seus parâmetros de entrada. Foram construídas funções separadas para tal formato de reforçador devido ao possível uso em lados opostos na região do duplo costado da embarcação.

Os diagramas de concepção para tais funções são mostrados nas **Figuras 9** e **10** a seguir. Abaixo na **Figura 11** temos um demonstrativo de uso das funções - à esquerda temos um reforçador do tipo L1 com os parâmetros a = b = 10, t = h = t_{base} = 2, origem em (1,1) e theta1 = 90º e à direita temos um reforçador do tipo L2 com os mesmos parâmetros geométricos que o anterior, com origem em (15,1) e theta1 = 270º - nessa figura, em azul, temos a origem, as linhas em preto definem a geometria do reforçador e, em vermelho, temos as linhas de particionamento.

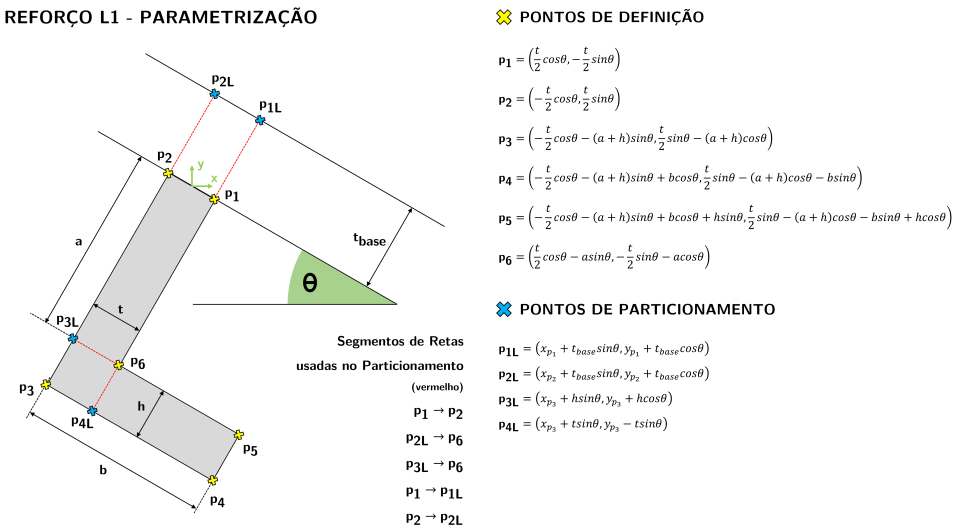


Figura 9 – Diagrama usado na construção da função L1 Reinforcement Points.

Fonte: Elaboração Própria.

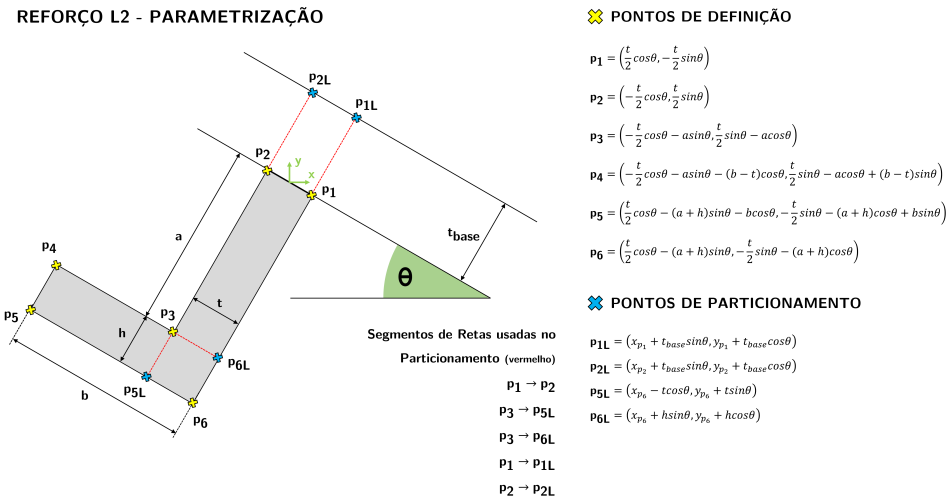


Figura 10 – Diagrama usado na construção da função L2 Reinforcement Points.

Fonte: Elaboração Própria.

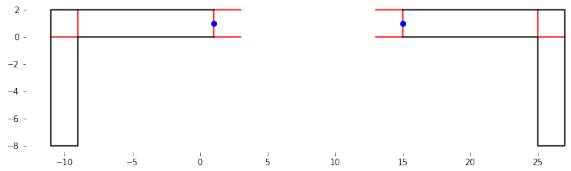


Figura 11 – Exemplo de Aplicação das funções L1 e L2 Reinforcement Points.

Fonte: Elaboração Própria.

2.2.4 Reinforcement Main

A função Reinforcement Main é responsável por fazer o acoplamento de todas as funções já descritas acima. Tal função recebe dois pontos, pA e pB - no formato (x,y), o número n de reforçadores a serem aplicados entre tais dois pontos, o tipo de reforçador a ser aplicado (T, L1 ou L2), as dimensões geométricas do reforçador (a,b,t e h), espessura da base de aplicação do reforço e o ângulo responsável pela ortogonalidade do elemento reforçador em relação à sua base. A **Figura 12** mostra um exemplo de uso da função Reinforcement Main, aplicando entre os pontos (0,0) e (1500,300), cinco reforçadores do tipo T com dimensões a = b = 100 e t = h = t_{base} = 25 e com variação angular (angle variation) de $-\pi/2$.

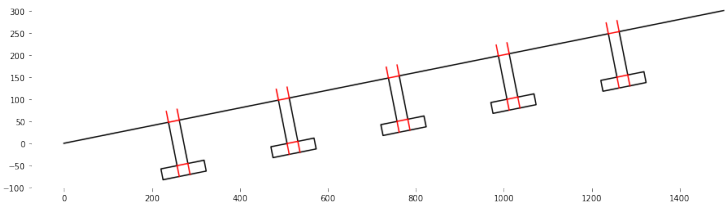


Figura 12 – Exemplo de Aplicação da função Reinforcement Main.

Fonte: Elaboração Própria.

2.2.5 Double Reinforcement Main

A função Double Reinforcement Main foi concebida devido à necessidade de alocar dois tipos diferentes de reforçadores entre dois pontos. Em algumas embarcações modeladas em trabalhos consultados na bibliografia - tal qual mostrado em (12), especialmente na antepara longitudinal de petroleiros, é comum observar o uso de alguns reforçadores maiores e, entre eles, outros reforçadores, geralmente com outro formato.

Essa função recebe dois pontos, pA e pB, no formato de pares ordenados (x,y), tipo, número, dimensões e variação de inclinação para cada um dos reforçadores a serem alocados. O primeiro tipo de reforçador, com parâmetros indexados em RI, será primeiramente alocado no comprimento entre A e B - em seguida, entre cada par dos reforçadores primários, serão alocados n_{RII} reforçadores do segundo tipo. Para melhor entendimento da natureza dos parâmetros recebidos pelas funções descritas na presente seção, é recomendado observar as tabelas presentes da **Seção 2.3**.

A **Figura 13** mostra um exemplo de uso da função descrita acima. Para o exemplo foram aplicados, entre os pontos (0,0) e (1500,300) (respectivas coordenadas x e y), três reforçadores do tipo T com dimensões de alma, flange, espessura de alma, espessura de flange e espessura da base respectivamente iguais a 75, 75, 15, 15 e 15 e tal reforçador é alocado a 90º da inclinação formada pelo pontos A e B. Entre os reforçadores T, são aplicados três reforçadores do tipo L2 com dimensões (análogas às comentadas anteriormente) iguais a 25, 25, 10, 10 e 10 - também ortogonal e com ordem de pontos AB.

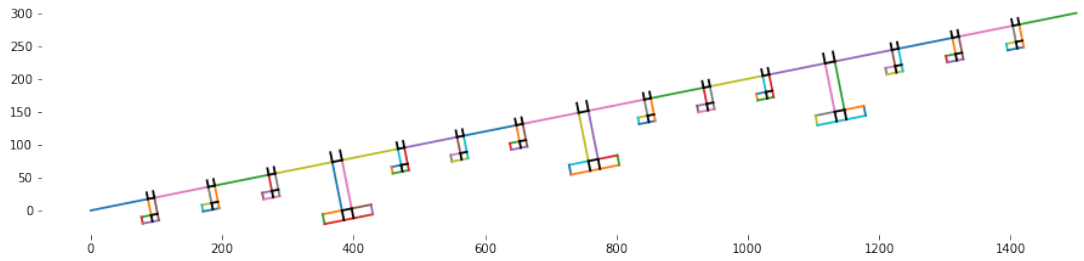


Figura 13 – Exemplo de Aplicação da função Double Reinforcement Main.

Fonte: Elaboração Própria.

2.2.6 Funções de Apoio

Nessa subseção serão apresentadas funções que desempenham um papel pontual na concepção do modelo, porém são fundamentais para a construção da geometria final da seção longitudinal.

- **Cartesian Distance:** Função para devolver a distância cartesiana entre dois pontos no plano. Criada apenas por simplicidade e utilizada na conversão do espaçamento de elementos em uma região no número de elementos a serem aplicados. O espaçamento entre elementos é definido de centro a centro.
- **Return Middle:** Função para devolver o ponto médio entre dois pontos do plano. Usada especificamente para seleção de retas no elemento reforçador transversal.
- **Y from X Parallel Line:** Função que recebe um ponto x, dois pontos pA e pB - pares ordenados (x,y), uma distância t e o ângulo de inclinação formado entre os pontos A e B e o eixo das abscissas e retorna o valor y de uma reta paralela a formada pelos pontos pA e pB e distante t.

- **X from Y Parallel Line:** Função análoga a apresentada anteriormente, porém ao invés de retornar o valor de y em função de x ela retorna o inverso, ou seja, o valor de x em função de y.
- **Sided Points:** Função fundamental para construção da já descrita Double Reinforcement Main. É responsável por devolver os pontos laterais a origem dos reforçadores - usada no desenvolvimento da já citada função responsável por posicionar dois tipo diferentes de reforçadores entre dois pontos. A **Figura 14** mostra a geometria que foi avaliada para a construção da respectiva função.
- **Grouper:** Função usada para agrupar elementos de uma lista dois a dois. Usada internamente da função Double Reinforcement Main - responsável por criar todos os pontos definidores da geometria do modelo.
- Mirror Image: Função responsável por espelhar um par ordenado (x,y) em função de um plano. Essa função foi usada após a construção de toda metade da geometria. Devido à limitações do *software* Abaqus CAE, optou-se por realizar o espelhamento ainda no ambiente Python e construir a geometria completa dentro do *software CAE* citado.

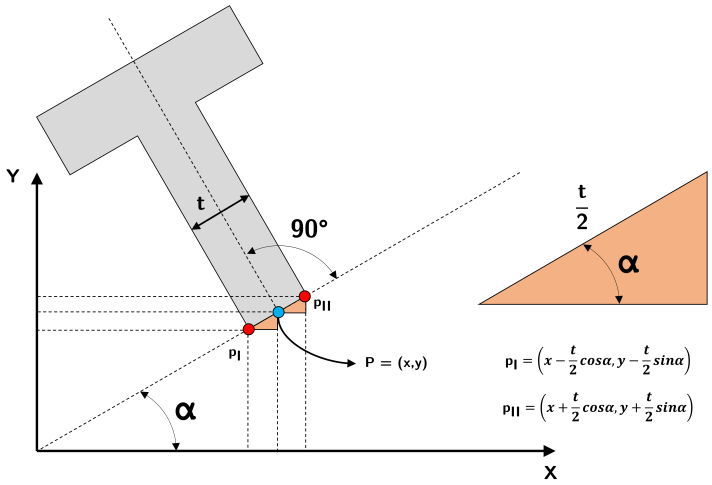


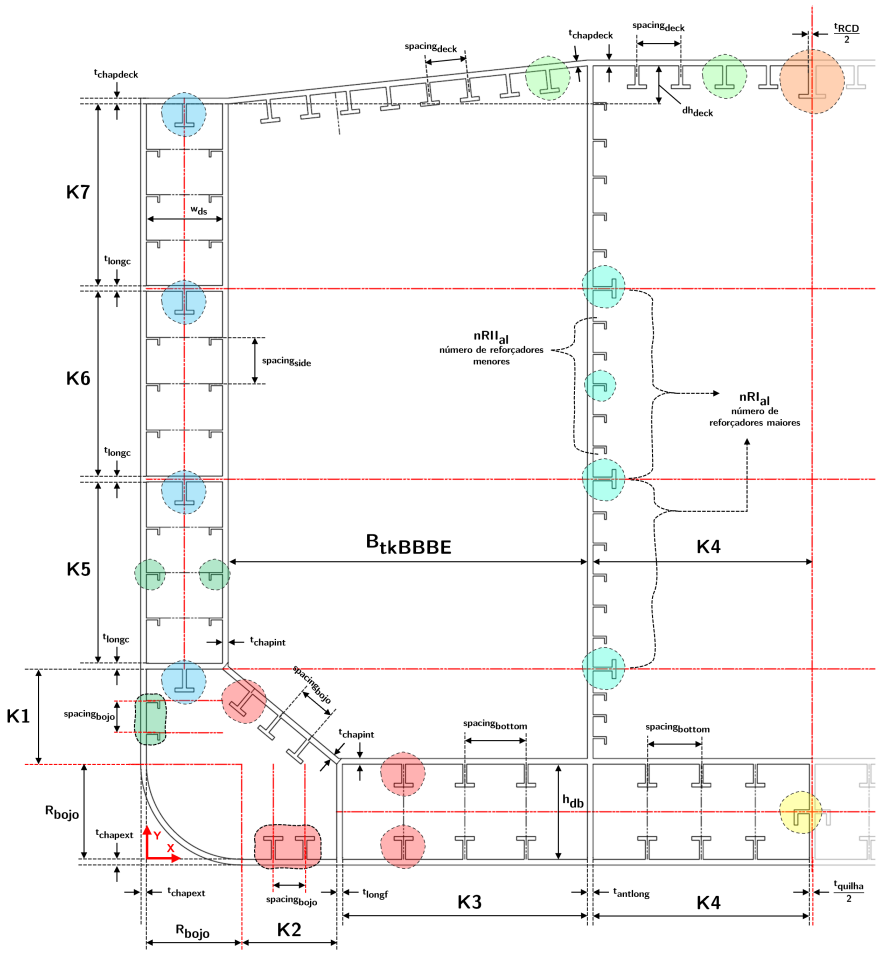
Figura 14 – Esquemático mostrando a geometria para construção da função Sided Points.

Fonte: Elaboração Própria.

2.3 Apresentação e Descrição dos Parâmetros Finais

Para continuar com a descrição completa da parametrização que foi aplicada ao modelo final, é preciso apresentar e descrever todos os parâmetros que foram utilizados. As tabelas mostradas a seguir apresentam e descrevem todos os parâmetros do modelo final, bem como mostra qual classe é esperada para tal (inteiro, ponto flutuante (*float*), *string*).

Para entendimento completo da natureza de aplicação dos parâmetros que serão apresentados a seguir, é recomendado consultar de forma concomitante o esquemático principal de parâmetros dado pela **Figura 15**.



REFORÇADORES E RESPECTIVOS PARÂMETROS

REFORÇADOR DO FUNDO
Parâmetros: Ref_bottom_type, a_bottom, b_bottom, t_bottom, h_bottom

REFORÇADOR DO COSTADO
Parâmetros: Ref_Lside_type, Ref_Rside_type, a_side, b_side, t_side, h_side

REFORÇADOR DA LONGITUDINAL DO COSTADO
Parâmetros: nRef_cost, Ref_cost_type, aRef_cost, bRef_cost, tRef_cost, hRef_cost

REFORÇADOR DO CONVÉS (DECK)
Parâmetros: Ref_deck_type, a_deck, b_deck, t_deck, h_deck

REFORÇADOR DA QUILHA
Parâmetros: nRef_q, Ref_q_type, a_q, b_q, t_q, h_q

REFORÇADORES DA ANTEPARA LONGITUDINAL
Parâmetros – Ref. Maior: nRI_al, RI_al_type, aRI_al, bRI_al, tRI_al, hRI_al
Parâmetros – Ref. Menor: nRII_al, RII_al_type, aRII_al, bRII_al, tRII_al, hRII_al

REFORÇADOR CENTRAL DO CONVÉS (DECK) – RCD (Reforçador Central do Deck)
Parâmetros: a_RCD, b_RCD, t_RCD, h_RCD

PARÂMETROS FIXADOS (simplificação de uso)

$K1 = K2 = B_{tkBBBE} + w_{ds} + t_{chapint} - (R_{bojo} + K3 + t_{longf})$

$B = 2 \cdot (t_{chapext} + w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4)$

$D = R_{bojo} + K1 + t_{longc} + K5 + t_{longc} + K6 + t_{longc} + K7 + dh_{deck} + t_{chapdeck}$

Input do usuário

Definido por outros parâmetros

Figura 15 – Esquemático mostrando a localidade de cada um dos parâmetros gerais - **Tabela 1**.

Fonte: Elaboração Própria.

2.3.1 Parâmetros Gerais

Os parâmetros apresentados na **Tabela 1** envolvem a definição indireta dos escantilhões do modelo, espessuras de chapeamentos e chapas em geral, número de entidades reforçadoras transversais e espaçamentos entre reforçadores longitudinais de fundo, costado, convés e bojo.

Tabela 1 – Parâmetros Gerais de Geometria e Montagem - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
$n_{anteparas}$	número de anteparas transversais	inteiro ≥ 0
$n_{cavernas}$	número de cavernas (gigantes) alocadas entre anteparas	inteiro ≥ 0
$L_{longframe}$	comprimento entre cavernas	*float* > 0
B_{tkBBBE}	largura dos tanques de BB e BE	*float* > 0
R_{bojo}	raio do bojo da embarcação	*float* > 0
h_{db}	altura do duplo fundo	*float* > 0
w_{ds}	largura do duplo costado	*float* > 0
dh_{deck}	variação de altura convés central e BB BE	*float* > 0
t_{longc}	espessura da longitudinal maior do costado	*float* > 0
t_{longf}	espessura da longitudinal maior do fundo	*float* > 0
t_{quilha}	espessura da quilha	*float* > 0
$t_{antlong}$	espessura da antepara longitudinal	*float* > 0
$t_{caverna}$	espessura da caverna (anel gigante)	*float* > 0
$t_{bulkhead}$	espessura da antepara transversal	*float* > 0
$t_{chapint}$	espessura do chapeamento interno	*float* > 0
$t_{chapext}$	espessura do chapeamento externo	*float* > 0
$t_{chapdeck}$	espessura do chapeamento do convés	*float* > 0
$spacing_{bottom}$	espaçamento dos reforçadores do fundo	*float* > 0
$spacing_{side}$	espaçamento dos reforçadores do costado	*float* > 0
$spacing_{deck}$	espaçamento dos reforçadores do convés	*float* > 0
$spacing_{bojo}$	espaçamento dos reforçadores do bojo	*float* > 0
K3	largura do trecho horizontal de fundo do tanque BB e BE	*float* > 0
K4	largura de metade do tanque central	*float* > 0
K5	altura do primeiro trecho de duplo costado	*float* > 0
K6	altura do segundo trecho de duplo costado	*float* > 0
K7	altura do terceiro trecho de duplo costado	*float* > 0

Fonte: Elaboração própria.

2.3.2 Reforçadores Longitudinais

As **Tabelas 2, 3, 4, 5, 6, 7** e **8** apresentam todos os parâmetros relativos à reforçadores longitudinais maiores e menores.

Tabela 2 – Parâmetros - Reforçador Central do Convés (*Deck*) - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
a_{RCD}	altura da alma do reforçador T	*float* > 0
b_{RCD}	largura do flange do reforçador T	*float* > 0
t_{RCD}	espessura da alma do reforçador T	*float* > 0
h_{RCD}	espessura da flange do reforçador T	*float* > 0

Fonte: Elaboração própria.

Tabela 3 – Parâmetros - Reforçador da Longitudinal do Costado - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
$nRef_{cost}$	número de reforçadores da long. do costado	inteiro ≥ 0
$Ref_{cost_{type}}$	tipo de reforçador da long. do costado	*string* $\in \{T, L1, L2\}$
$aRef_{cost}$	altura da alma do reforçador	*float* > 0
$bRef_{cost}$	largura do flange do reforçador	*float* > 0
$tRef_{cost}$	espessura da alma do reforçador	*float* > 0
$hRef_{cost}$	espessura do flange do reforçador	*float* > 0

Fonte: Elaboração própria.

Tabela 4 – Parâmetros - Reforçador do Fundo - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
$Ref_{bottom_{type}}$	tipo de reforçador do fundo	*string* $\in \{T, L1, L2\}$
a_{bottom}	altura da alma do reforçador	*float* > 0
b_{bottom}	largura do flange do reforçador	*float* > 0
t_{bottom}	espessura da alma do reforçador	*float* > 0
h_{bottom}	espessura do flange do reforçador	*float* > 0

Fonte: Elaboração própria.

Tabela 5 – Parâmetros - Reforçador da Quilha - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
$nRef_q$	número de reforçadores da quilha	inteiro ≥ 0
$Ref_{q_{type}}$	tipo de reforçador da quilha	*string* $\in \{T, L1, L2\}$
a_q	altura da alma do reforçador	*float* > 0
b_q	largura do flange do reforçador	*float* > 0
t_q	espessura da alma do reforçador	*float* > 0
h_q	espessura do flange do reforçador	*float* > 0

Fonte: Elaboração própria.

Tabela 6 – Parâmetros - Reforçador do Costado - Descrição e Classe

Parâmetro	Descrição	Classe
$\text{Ref}_{\text{Lside}_{\text{type}}}$	tipo de reforçador do costado (chap. externo)	*string* $\in \{\text{T}, \text{L1}, \text{L2}\}$
$\text{Ref}_{\text{Rside}_{\text{type}}}$	tipo de reforçador do costado (chap. interno)	*string* $\in \{\text{T}, \text{L1}, \text{L2}\}$
a_{side}	altura da alma do reforçador	*float* > 0
b_{side}	largura do flange do reforçador	*float* > 0
t_{side}	espessura da alma do reforçador	*float* > 0
h_{side}	espessura do flange do reforçador	*float* > 0

Fonte: Elaboração própria.

Tabela 7 – Parâmetros - Reforçador do Convés - Descrição e Classe

Parâmetro	Descrição	Classe
$\text{Ref}_{\text{deck}_{\text{type}}}$	tipo de reforçador do convés	*string* $\in \{\text{T}, \text{L1}, \text{L2}\}$
a_{deck}	altura da alma do reforçador	*float* > 0
b_{deck}	largura do flange do reforçador	*float* > 0
t_{deck}	espessura da alma do reforçador	*float* > 0
h_{deck}	espessura do flange do reforçador	*float* > 0

Fonte: Elaboração própria.

Tabela 8 – Parâmetros - Reforçadores da Antepara Longitudinal - Descrição e Classe

Parâmetro	Descrição	Classe
nRI_{al}	número de reforçadores maiores da ant. longitudinal	inteiro ≥ 0
$\text{RI}_{\text{al}_{\text{type}}}$	tipo de reforçador maior	*string* $\in \{\text{T}, \text{L1}, \text{L2}\}$
aRI_{al}	altura da alma do reforçador maior	*float* > 0
bRI_{al}	largura do flange do reforçador maior	*float* > 0
tRI_{al}	espessura da alma do reforçador maior	*float* > 0
hRI_{al}	espessura do flange do reforçador maior	*float* > 0
nRII_{al}	número de reforçadores menores da ant. longitudinal	inteiro ≥ 0
$\text{RII}_{\text{al}_{\text{type}}}$	tipo de reforçador menor	*string* $\in \{\text{T}, \text{L1}, \text{L2}\}$
aRII_{al}	altura da alma do reforçador menor	*float* > 0
bRII_{al}	largura do flange do reforçador menor	*float* > 0
tRII_{al}	espessura da alma do reforçador menor	*float* > 0
hRII_{al}	espessura do flange do reforçador menor	*float* > 0

Fonte: Elaboração própria.

2.3.3 Gigantes (cavernas) e outros

A **Tabela 9** descreve os parâmetros que definem a geometria das cavernas (gigantes) do modelo - para maior esclarecimento é sugerido consultar a **Figura 22** para observar o que realmente

significam os parâmetros descritos. Os parâmetros materiais, de condição de contorno e de controle do tamanho dos elementos na malha são apresentados na **Tabela 10**

Tabela 9 – Parâmetros Cavernas (anéis gigantes) - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
dh_{deckC}	altura do vau que compõe a gigante	*float* ≥ 0
dh_{sideC}	variação de largura relativa ao costado	*float* ≥ 0
$dh_{bottomC}$	variação de largura relativa ao fundo	*float* ≥ 0
$dh_{antlongC}$	variação de largura relativa a antepara longitudinal	*float* ≥ 0
$R_{caverna}$	raio de arredondamento dos cantos da caverna (gigante)	*float* ≥ 0

Fonte: Elaboração própria.

Tabela 10 – Parâmetros Materiais, de Contorno e Malha - Descrição e Classe

Parâmetro	**Descrição**	**Classe**
$Material_{Name}$	nome do material a ser aplicado	*string*
E_{Young}	módulo de elasticidade do material	*float* > 0
$v_{poisson}$	coeficiente de Poisson do material	*float* > 0
density	densidade do material	*float* > 0
$applied_{rotation}$	rotação em radianos a ser aplicada ao modelo	*float* > 0
$Average_{element_{size}}$	tamanho médio de elemento a ser aplicado às malhas	*float* > 0

Fonte: Elaboração própria.

2.3.4 Arquivo de Parâmetros - **Apêndice 5**

O arquivo *PARAMETERS.py* contém a função responsável por exportar um simples arquivo texto que faz a comunicação dos parâmetros configurados. Os dados exportados são lidos tando pelo arquivo responsável para criação da geometria (arquivo *GEOMETRY.py* - **Apêndice 7**) quanto pelo arquivo *MIDSHIP SECTION FINAL.py* - **Apêndice 8**, que é o modelo final executado pelo Abaqus CAE.

A função mencionada, chamada *Return Parameters File* recebe como entrada um dicionário do Python contendo todos os 81 parâmetros descritos anteriormente. Após ler o dicionário, tal função escreve um arquivo texto em três colunas, contendo o nome do parâmetro na primeiro coluna, valor do parâmetro na segunda e a classe que a variável pertence na última - isso é utilizado para reconstrução das variáveis em outro ambiente.

Esse procedimento é realizado devido à ruídos de comunicação entre **Python 2** (usado pelo Abaqus CAE) e o **Python 3** (utilizado na construção da geometria e execução sequencial dos arquivos). Apesar de não ser uma forma muito elegante de integração dos arquivos, se mostrou funcional o suficiente. A **Figura 16** mostra o arquivo texto exportado para melhor visualização do seu formato.

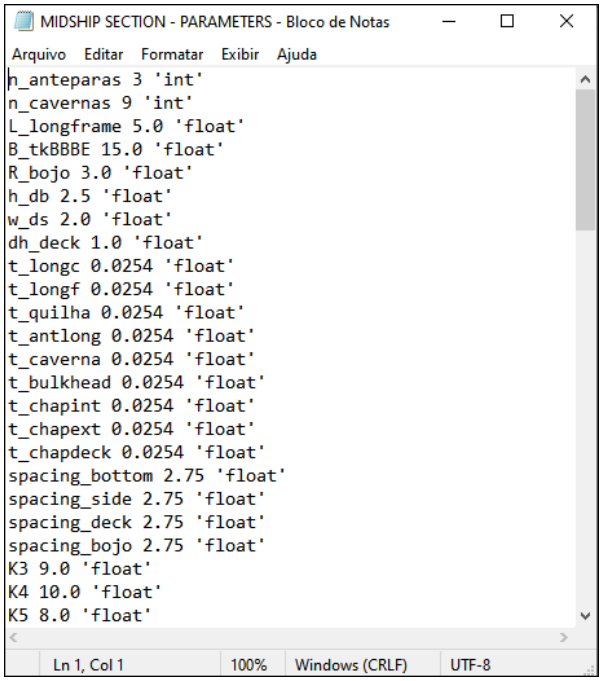


Figura 16 – Formato do arquivo de parâmetros exportado pela função *Return Parameters File*.

Fonte: Elaboração Própria.

2.4 Parametrização dos Pontos Principais

A seguir serão apresentados os pontos principais que definem a geometria do modelo final, tanto no arranjo longitudinal, quanto no arranjo transversal. Os pontos principais são definidos pelos parâmetros que foram descritos na **Seção 2.3**.

2.4.1 Arranjo Longitudinal

Os pontos principais que definem o arranjo geométrico longitudinal do modelo final são apresentados pela **Tabela 11**. Para melhor entendimento dos pontos que são apresentados abaixo é recomendado consultar as **Figuras 17**, **18**, **19** e **20**.

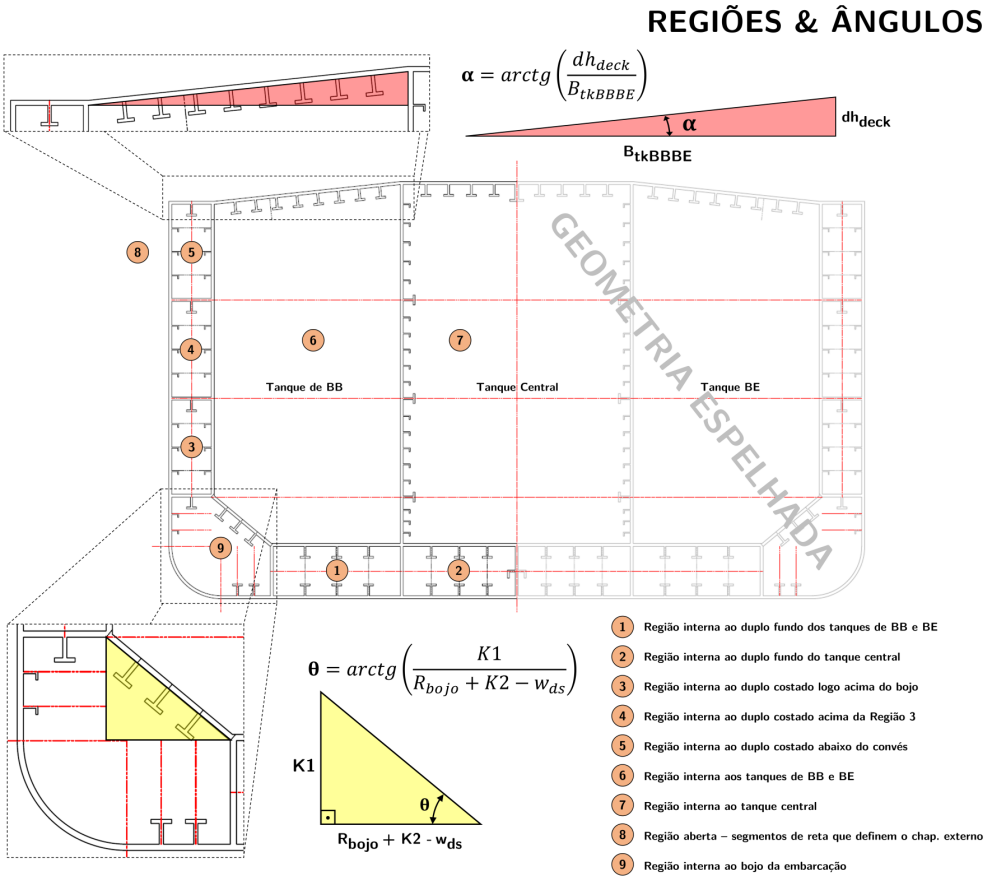


Figura 17 – Definição das regiões tratadas na geometria longitudinal.

Fonte: Elaboração Própria.

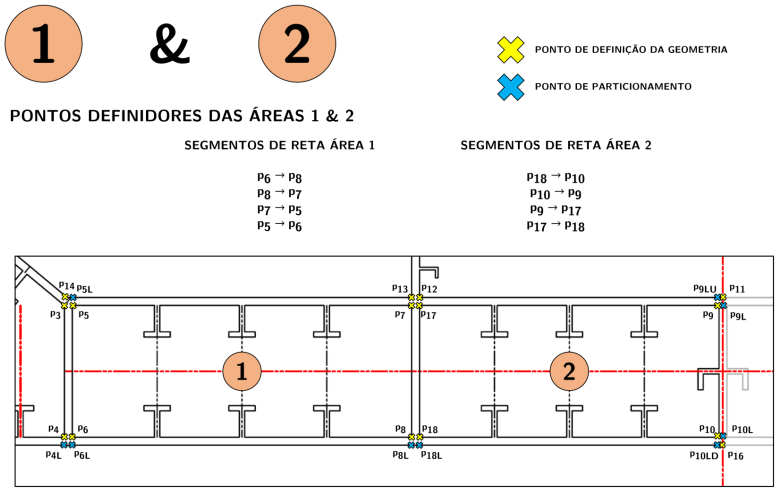


Figura 18 – Detalhamento dos pontos das regiões 1 e 2.

Fonte: Elaboração Própria.

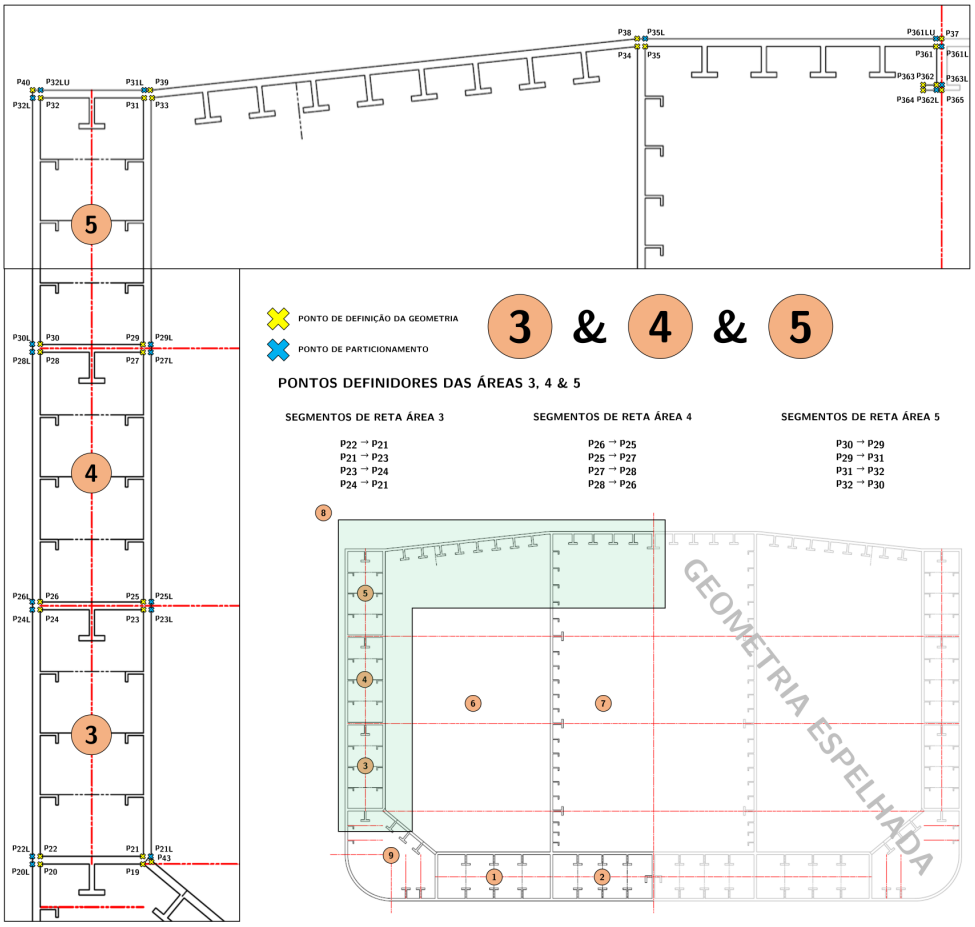


Figura 19 – Detalhamento dos pontos das regiões 3, 4 e 5.

Fonte: Elaboração Própria.

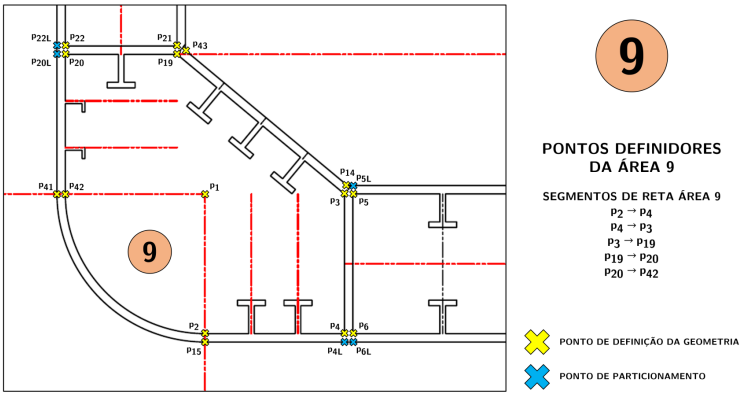


Figura 20 – Detalhamento dos pontos da região 9.

Fonte: Elaboração Própria.

Tabela 11 – Pontos Principais - Arranjo Longitudinal - Coordenadas X e Y

Ponto	Coordenada X	Coordenada Y
p_1	R_{bojo}	R_{bojo}
p_2	R_{bojo}	0
p_3	$K2 + R_{bojo}$	h_{db}
p_4	$K2 + R_{bojo}$	0
p_5	$K2 + R_{bojo} + t_{longf}$	h_{db}
p_6	$K2 + R_{bojo} + t_{longf}$	0
p_7	$K2 + R_{bojo} + t_{longf} + K3$	h_{db}
p_8	$K2 + R_{bojo} + t_{longf} + K3$	0
p_9	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong} + K4 - \frac{t_{quilha}}{2}$	h_{db}
p_{10}	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong} + K4 - \frac{t_{quilha}}{2}$	0
p_{11}	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong} + K4$	$h_{db} + t_{chapint}$
p_{12}	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong}$	$h_{db} + t_{chapint}$
p_{13}	$K2 + R_{bojo} + t_{longf} + K3$	$h_{db} + t_{chapint}$
p_{15}	R_{bojo}	$-t_{chapext}$
p_{16}	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong} + K4$	$-t_{chapext}$
p_{17}	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong}$	h_{db}
p_{18}	$K2 + R_{bojo} + t_{longf} + K3 + t_{antlong}$	0
p_{19}	w_{ds}	$R_{bojo} + K1$
p_{141}	$K2 + R_{bojo} + \frac{1-\cos\theta}{\sin\theta} \cdot t_{chapint}$	$h_{db} + t_{chapint}$
p_{142}	x from y Parallel Line $\left(h_{db} + t_{chapint}, p_3, p_{19}, t_{chapint}, \theta\right)$	$h_{db} + t_{chapint}$
p_{14}	$\frac{X_{p141}+X_{p142}}{2}$	$\frac{Y_{p141}+Y_{p142}}{2}$
p_{20}	0	$R_{bojo+K1}$
p_{21}	w_{ds}	$R_{bojo+K1} + t_{longc}$
p_{22}	0	$R_{bojo+K1} + t_{longc}$
p_{23}	w_{ds}	$R_{bojo+K1} + +K5 + t_{longc}$
p_{24}	0	$R_{bojo+K1} + K5 + t_{longc}$
p_{25}	w_{ds}	$R_{bojo+K1} + K5 + 2t_{longc}$
p_{26}	0	$R_{bojo+K1} + K5 + 2t_{longc}$
p_{27}	w_{ds}	$R_{bojo+K1} + K5 + K6 + 2t_{longc}$
p_{28}	0	$R_{bojo+K1} + K5 + K6 + 2t_{longc}$
p_{29}	w_{ds}	$R_{bojo+K1} + K5 + K6 + 3t_{longc}$
p_{30}	0	$R_{bojo+K1} + K5 + K6 + 3t_{longc}$
p_{31}	w_{ds}	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc}$
p_{32}	0	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc}$
p_{33}	$w_{ds} + t_{chapint}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc}$
p_{34}	$w_{ds} + t_{chapint} + B_{tkBBBE}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck}$
p_{35}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck}$
p_{361}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4 - \frac{t_{RCD}}{2}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck}$
p_{362}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4 - \frac{t_{RCD}}{2}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} - a_{RCD}$
p_{363}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4 - \frac{t_{RCD}}{2} - \frac{b_{RCD}}{2}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} - a_{RCD}$
p_{364}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4 - \frac{t_{RCD}}{2} - \frac{b_{RCD}}{2}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} - a_{RCD} - h_{RCD}$
p_{365}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} - a_{RCD} - h_{RCD}$
p_{37}	$w_{ds} + t_{chapint} + B_{tkBBBE} + t_{antlong} + K4$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} + t_{chapdeck}$
p_{38} para $\alpha \neq 0$	$w_{ds} + t_{chapint} + B_{tkBBBE} - \frac{(1/\cos\alpha)-1}{\tan\alpha} \cdot t_{chapdeck}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} + t_{chapdeck}$
p_{39} para $\alpha \neq 0$	$w_{ds} + t_{chapint} - \frac{(1/\cos\alpha)-1}{\tan\alpha} \cdot t_{chapdeck}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} + t_{chapdeck}$
p_{38} para $\alpha = 0$	$w_{ds} + t_{chapint} + B_{tkBBBE}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} + t_{chapdeck}$
p_{39} para $\alpha = 0$	$w_{ds} + t_{chapint}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} + t_{chapdeck}$
p_{40}	$-t_{chapext}$	$R_{bojo+K1} + K5 + K6 + K7 + 3t_{longc} + dh_{deck} + t_{chapdeck}$
p_{41}	$-t_{chapext}$	R_{bojo}
p_{42}	0	R_{bojo}
p_{431}	$w_{ds} + t_{chapint}$	$R_{bojo} + K1 + \frac{1-\sin\theta}{\cos\theta} \cdot t_{chapint}$
p_{432}	$w_{ds} + t_{chapint}$	y from x Parallel Line $\left(w_{ds} + t_{chapint}, p_3, p_{19}, t_{chapint}, \theta\right)$
p_{43}	$\frac{X_{p431}+X_{p432}}{2}$	$\frac{Y_{p431}+Y_{p432}}{2}$

Fonte: Elaboração própria.

2.4.2 Arranjo Transversal

A seguir serão apresentados os esquemáticos e pontos que definem a geometria tanto dos anéis gigantes, quanto das anteparas transversais. Em muitas variáveis dos códigos desenvolvidos, as gigantes estão sendo mencionadas como cavernas, sendo apenas uma espécie de metonímia relativa a realidade, uma vez que a caverna é uma parte que compõe um anel gigante. Antes de apresentar a tabela contendo as coordenadas x e y dos ponto que definem a geometria da gigante, é importante apresentar a localidade espacial desses pontos e o significado dos parâmetros que foram apresentados pela **Tabela 9** na **Seção 2.3**. As **Figuras 21** e **22** apresentam a geometria de uma gigante, esclarecendo os parâmetros e a localidade dos pontos a serem em seguida descritos.

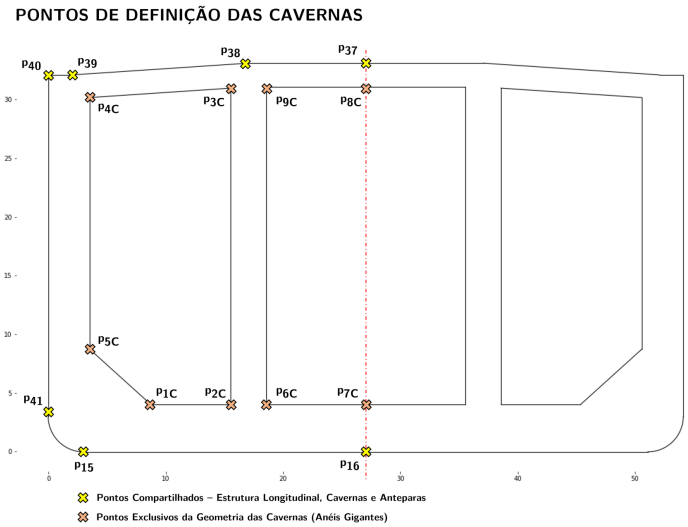


Figura 21 – Pontos de definição da geometria das Gigantes.

Fonte: Elaboração Própria.

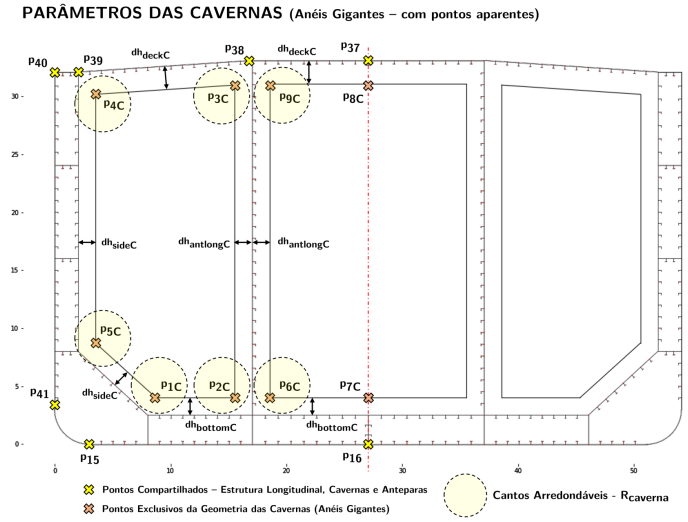


Figura 22 – Localidade dos parâmetros definidores da geometria das gigantes.

Fonte: Elaboração Própria.

Tabela 12 – Pontos de Definição da Geometria - Gigantes (cavernas)

Ponto	Coordenada X	Coordenada Y
p_{1C}	$x_{p_{14}} - \frac{dh_{sideC} - \frac{dh_{bottomC}}{\sin(\pi/2-\theta)}}{\sin\theta}$	$y_{p_{14}} + dh_{bottomC}$
p_{2C}	$x_{p_{13}} - dh_{antlongC}$	$y_{p_{13}} + dh_{bottomC}$
p_{3C}	$x_{p_{34}} - dh_{antlongC}$	$y_{p_{34}} - \frac{dh_{deckC}}{\cos\alpha} - dh_{antlongC} \cdot \tan\alpha$
p_{4C}	$x_{p_{33}} + dh_{sideC}$	$y_{p_{33}} - \frac{dh_{deckC} - df_{sideC \cdot \sin\alpha}}{\cos\alpha}$
p_{5C}	$x_{p_{43}} + dh_{sideC}$	$y_{p_{43}} + dh_{sideC} \cdot \frac{1-\sin\theta}{\cos\theta}$
p_{6C}	$x_{p_{12}} + dh_{antlongC}$	$y_{p_{12}} + dh_{bottomC}$
p_{7C}	$x_{p_{11}}$	$y_{p_{11}} + dh_{bottomC}$
p_{8C}	$x_{p_{37}}$	$y_{p_{361}} - df_{deckC}$
p_{9C}	$x_{p_{35}} + dh_{antlongC}$	$y_{p_{35}} - dh_{deckC}$

Fonte: Elaboração própria.

As anteparas transversais não possuem pontos particulares, ou seja, a geometria que constitui ela utiliza pontos provenientes do arranjo longitudinal. A **Figura 23** apresenta graficamente a correspondência dos pontos de definição da antepara e aqueles já descritos anteriormente na **Subseção 2.4.1**.

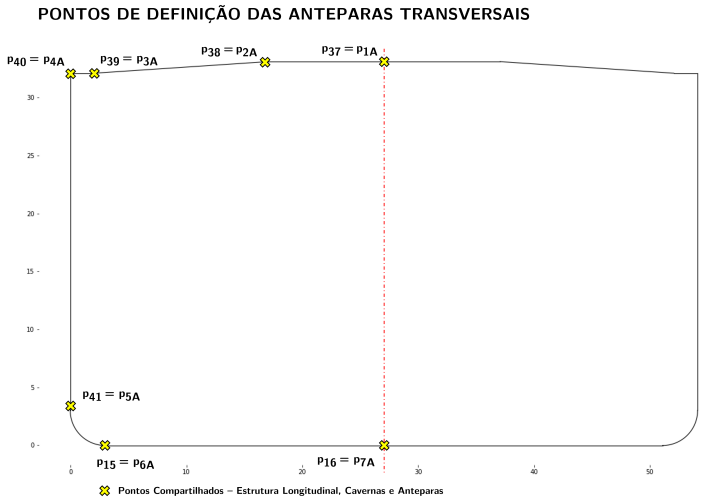


Figura 23 – Pontos de definição da geometria das anteparas transversais.

Fonte: Elaboração Própria.

2.5 Criação da Geometria - **Apêndice 7**

A seguir serão apresentados os trechos de código, pertencentes ao arquivo *GEOMETRY.py* - **Apêndice 7** - que são responsáveis pela criação da geometria do modelo, fazendo uso das funções descritas na **Seção 2.2**, parâmetros apresentados na **Seção 2.3** e pontos definidos na **Seção 2.4**. O arquivo mencionado, *GEOMETRY.py*, faz a leitura do arquivo texto exportado pelo arquivo *PARAMETERS.py* para recebimento dos parâmetros configurados para o modelo e exportação dos dados da geometria - que posteriormente será construída no Abaqus CAE.

A leitura do arquivo - realizada entre as linhas 7 e 21 do respectivo código mostrado no **Apêndice 7** - é bem simples. É executado um *loop* para cada linha pertencente ao arquivo texto e tal linha é quebrada em uma lista com três valores, usando como separador o espaço. Em seguida são recriadas as variáveis globais em função do nome do parâmetro, valor e classe escritos no arquivo texto.

Abaixo segue a listagem das linhas de código que são responsáveis pela criação da geometria do atual modelo - o código se encontra anexado em **Apêndice 7**.

- **Arranjo Longitudinal:** a partir da linha 149 até a linha 240
- **Arranjo Transversal:** a partir da linha 242 até a linha 264

Devido à dificuldades com relação a criação da malha em elementos finitos, em especial para o arranjo longitudinal, foi necessário realizar o particionamento dessa estrutura, ou seja, dividir a geometria em seções menores e com formas mais básicas. As linhas do código que realizam tal ações vão da 267 até a linha de número 293.

A gigante - mostrada nos esquemáticos das **Figuras 21 e 22** - não apresenta arredondamento nos cantos que circundam os tanques. No modelo isso pode sim ser alterado, uma vez que existe o parâmetro $R_{caverna}$. Para que isso fosse implementado foi necessário pegar em sequência, de duas a duas, as retas que deveriam ser selecionadas para aplicar um *Fillet by Radius* dentro do ambiente de esboço do Abaqus CAE.

Para recolher esses dados foi implementada a rotina que é apresentada entre as linhas 295 e 326 do código presente no **Apêndice 7**, onde são montadas tuplas contendo dois pares (x,y) cada - indicando assim quais retas devem ser selecionadas para aplicação do raio de arredondamento. É importante mencionar que isso só é ativado no modelo quando o parâmetro $R_{caverna}$ é diferente de zero.

Após a criação de todas as listas de pontos para todas as partes que o modelo será composto (arranjo longitudinal, gigantes, anteparas transversais, partições e retas para arredondamento dos cantos das gigantes), o arquivo *GEOMETRY.py* escreve cinco arquivos texto. Esses arquivos contém os dados para construção do modelo pelo arquivo *MIDSHIP SECTION FINAL.py* (que é executado pelo Python do Abaqus CAE) - presente no **Apêndice 8**.

As linhas do código 328 até a 400 apresentado no **Apêndice 7** apresentam a rotina responsável por exportar respectivamente os pontos de geração da seção longitudinal, pontos das gigantes, pontos das anteparas transversais, pontos para o particionamento e os pontos para a seleção de retas para aplicação de arredondamento nas gigantes. Todos os arquivos exportados têm o mesmo formato de dados, sendo cada linha um segmento de reta formado por dois pontos, na seguinte sequência: xA yA xB yB - separados por um espaço simples.

A forma mais eficiente de desenhar dentro do Abaqus - após um certo tempo de avaliação prática - foi o uso de linhas, porém isso acaba implicando que é necessário passar como entrada o desenho completo segmento a segmento - assim como fica visível na demonstração de uso da função Double Reinforcement Main na **Figura 13** - na figura citada cada segmento possui uma cor diferente, é exatamente dessa forma que todas as geometrias são construídas internamente no software mencionado. Para não deixar dúvida quanto ao formato dos arquivos, a **Figura 24** mostra uma parte da saída relativa aos pontos principais.

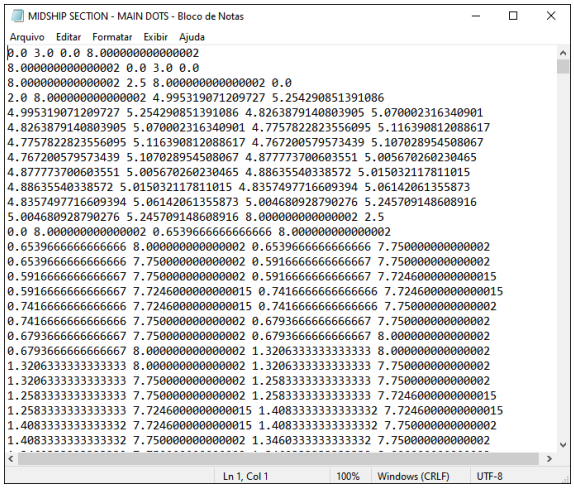


Figura 24 – Formato dos arquivos texto de saída para construção da geometria.

Fonte: Elaboração Própria.

2.6 Implementação Abaqus CAE - **Apêndice 8**

Após a finalização da exportação dos dados referentes à construção da geometria, será tratado da criação do modelo dentro do *software* Abaqus CAE. Não será detalhado linha a linha dos código apresentados, só será comentado a função global que ele está realizando - para maiores detalhes com relação à modelagem Abaqus Python verificar (4, 5).

2.6.1 Leitura dos Parâmetros e Criação do Modelo

Antes de iniciar qualquer atividade dentro do Abaqus CAE, e algo que é realizado automaticamente na inicialização da aplicação, é a importação das bibliotecas que serão utilizadas na criação dos modelos. São importadas ainda outras três bibliotecas para outras funcionalidades.

A primeira é o pacote **os** - que fica responsável por relacionar o Python com o sistema operacional do computador, usada na criação de pastas, manipulação de arquivos em geral. A segunda é a biblioteca **numpy**, usada para operações numéricas em geral e a última é a **datetime**. Tal *package* foi usado especificamente para obter o exato instante de início de execução do código. Esse instante é armazenado como uma variável *string* para ser usada na nomeação do diretório criado para armazenamento da simulação, evitando assim, que ocorram sobreposições de arquivos. As linhas 1 a 10 do código apresentado no **Apêndice 8** são responsáveis pela importação das bibliotecas a serem utilizadas.

Após a importação é realizada a leitura do arquivo texto de parâmetros - linhas 20 a 35 do código do **Apêndice 8**. Ao final da leitura do arquivo, os parâmetros estão armazenados na memória como variáveis globais. Antes de fazer a leitura do arquivo, foram realizadas duas ações importantes: criação de um diretório para a simulação - os.mkdir() - e alteração da pasta do Abaqus para o diretório criado - os.chdir().

Feita a leitura dos parâmetros, faz-se agora a criação do modelo - linhas 36 à 40 do código citado anteriormente. As linhas *backwardCompatibility* e *session.journalOptions* são fundamentais para a concepção de qualquer modelo, uma vez que elas forçam o Abaqus a devolver, em seu arquivo de histórico de operações - chamado de abaqus.rpy, as coordenadas das ações realizadas dentro da interface gráfica - o usual é o *software* devolver variáveis indexadas que dificultam o entendimento da construção do código. Agora está tudo adequado para a construção dos esboços que vão definir as partes do modelo.

2.6.2 Criação das Geometrias

A criação dos *sketchs* para cada uma das entidade é de certa forma equivalente. O primeiro ponto é criar um esboço e acessar suas *features* - geometria, vértices, dimensões, *contraints* - em seguida já possível iniciar a construção. Para todas as partes construídas, sempre foi iniciado com o desenho dos raios do bojo da embarcação, porém isso não é regra - poderia ter sido feito ao final sem qualquer complicação.

Para cada parte é feita a leitura do respectivo arquivo de pontos e em seguida é feito um *loop* para a construção de cada segmento de reta dado pelos dois pontos. Ao final é feita a extrusão do perfil com o comando **BaseSolidExtrude** - atendendo à dimensão relativa de cada parte, ou seja, o perfil longitudinal é extrudado em $L_{longframe}$ e assim por diante.

O raio de arredondamento das cavernas só é aplicado quando for diferente de zero e a antepara transversal só é construída se o número de anteparas for também diferente de zero. As linhas entre 42 e 70, 72 e 109 e 111 e 137 do **Apêndice 8** contém os códigos relativos aos *sketchs* e extrusões dos perfis respectivamente do arranjo longitudinal, gigantes (cavernas) e anteparas. A **Figura 25** mostra um exemplo de *sketch* para o arranjo longitudinal - pode-se observar o grande número de reforçadores menores que foram alocados à geometria.

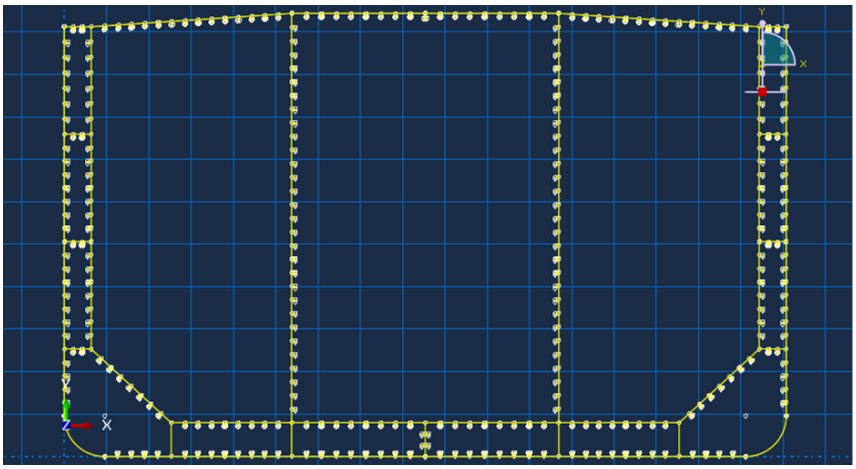


Figura 25 – Abaqus CAE - exemplo de *sketch* para o modelo final.

Fonte: Elaboração Própria.

2.6.3 Montagem

A montagem dos elementos foi concebida em função do arranjo que está apresentado na **Figura 5** e foi realizada uma generalização em função de $n_{cavernas}$ e $n_{anteparas}$. As linhas entre 139 e 165 do **Apêndice 8** tem-se o código que foi construído para realizar a montagem da estrutura - sendo ele dividido para dois casos - um com número de anteparas diferente de zero e outro para o caso de não estar alocando anteparas transversais na estrutura. A **Figura 26** apresenta um modelo montado na interface do software Abaqus CAE.

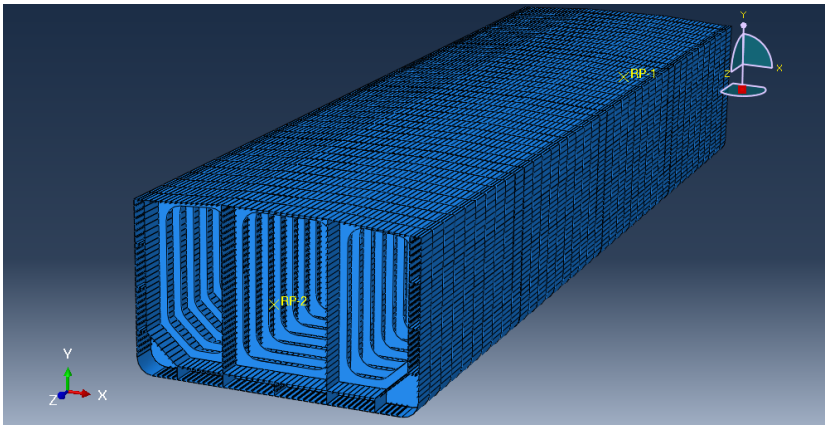


Figura 26 – Exemplo de montagem do modelo final.

Fonte: Elaboração Própria.

2.6.4 Interações e *Constraints*

Fundamentais para o sucesso do modelo, as interações - do tipo *Tie-Constraint* - que vão garantir que ocorrerá transmissão de esforços entre os elementos que compõem a seção da embarcação (arranjo longitudinal, gigantes e anteparas). Os contatos para a formação de um *Tie-Constraint* são feitos superfície a superfície - dessa forma foi necessário selecionar parametricamente os pares de superfícies que estivessem em contato para a criação das corretas interações.

O esquema utilizado para montagem - **Figura 5** - também foi útil para desenvolver a generalização no caso das interações. Também para essa configuração foram avaliados casos separados se o número de anteparas transversais é ou não diferente de zero. As linhas entre 167 e 196 do **Apêndice 8** são a rotina construída para aplicar os *Tie-Constraints*.

Ainda no campo das interações, foi criado para o carregamento de demonstração ser aplicado de melhor forma, dois **MPC constraint (Multi-point Constrain)** relacionando um ponto de referência criado um no centro da face frontal e outro no centro da face de trás do modelo (9). Os nós de cada uma das respectivas faces foram amarrados a esse ponto. As linhas entre 198 e 223 do código apresentado no **Apêndice 8** apresentam a criação dessas restrições e **Figura 27** mostra um exemplo de modelo com as interações aparentes - destaque para o **MPC Contraint** na face frontal.

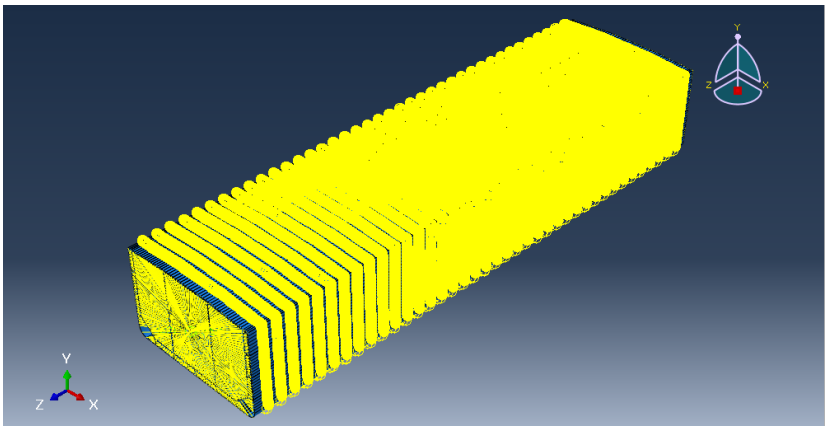


Figura 27 – Modelo com interações aparentes - exemplo.

Fonte: Elaboração Própria.

2.6.5 Material, Seção e Step

Todos os petroleiros modernos são construídos com juntas soldadas de aço. A tensão de escoamento dos aços utilizados vão de 235 MPa até 355 MPa (12). Para o presente estudo foi programado que apenas um material será assinado à todos os elementos que compõe a embarcação. É possível trabalhar e incrementar essa maior flexibilidade, porém devido à limitações de tempo isso ficará como recomendações futuras.

O material é rapidamente configurado em termos elásticos - estamos trabalhando sem a possibilidade de adicionar uma curva de tensão deformação para definição do material - apenas com coeficiente de Poisson e módulo de Elasticidade. É importante adicionar a densidade do material para ser possível realizar avaliações com relação à redução da massa estrutural em detrimento da tensão observada em um ponto crítico do modelo. Importante mencionar a necessidade de manter consistência no sistema de unidades adotado.

O Step (passo no tempo para avaliação estrutura estática) foi adicionado de forma simples e diretamente não foi deixado em aberto nenhum parâmetro para alteração paramétrica desse aspecto. As linhas entre 225 e 247 do **Apêndice 8** mostram a rotina relacionada à definição de material, seção e assinatura de material e Step.

2.6.6 Particionamento e Malha

O particionamento foi construído de forma parecida com a própria construção dos elementos do modelo. Primeiro foi realizada a leitura do arquivo texto contendo os segmentos de reta para a partição e montou-se o esboço. Particularmente nesse caso foi necessário selecionar todas as linhas do *sketch* criado para a realização da extrusão, a fim de cortar a estrutura longitudinal. Para realizar isso, foi realizada novamente a leitura do arquivo texto e foram sendo selecionados os pontos médios de cada segmento (usando a função Return Middle) e adicionando em uma lista.

Finalizada a seleção, bastou selecionar a entidade a ser particionada e uma aresta para ser utilizada como vetor da extrusão. Para construção da malha bastou selecionar os elementos e informar um tamanho médio de elemento. As linhas entre 305 e 367 do **Apêndice 8** apresentam os códigos construídos para realização da ações aqui descritas.

2.6.7 Job e Exportação dos Parâmetros na Pasta da Simulação

As linhas de código entre 369 e 382 do **Apêndice 8** são responsáveis pela criação do Job (simulação propriamente dita) e sequente submissão para avaliação. Logo após a execução é realizada a exportação dos parâmetros do modelo em um arquivo texto dentro da pasta da simulação que estava sendo executada - isso é importante para casos de múltiplas execuções e a possível necessidade de reensaiar um modelo especificamente.

Não serão tratadas das condições de contorno na presente seção porque elas serão tratadas como um caso particular de aplicação do modelo desenvolvido - os detalhes serão dados todos na seção a seguir. Serão apresentadas as configurações atuais do modelo para esse caso, configurações do código e exploração da saída, além de trazer resultados de uma avaliação paramétrica realizada.

3 Resultados

Como amostra de funcionalidade do modelo concebido, nessa seção serão apresentados alguns resultados com relação à uma definição de carregamento específica. A fim de realizar um ensaio que tenha ligação com as cargas que uma embarcação pode realmente enfrentar, foi definido a aplicação das adequadas condições de contorno a atender uma situação de tosamento. O tosamento (*sagging* em inglês) ocorre quando popa e proa da embarcação se encontram em cristas de onda - como mostra o esquemático da **Figura 28** - implicando em tensões compressivas no convés e tensões trativas no fundo do navio.

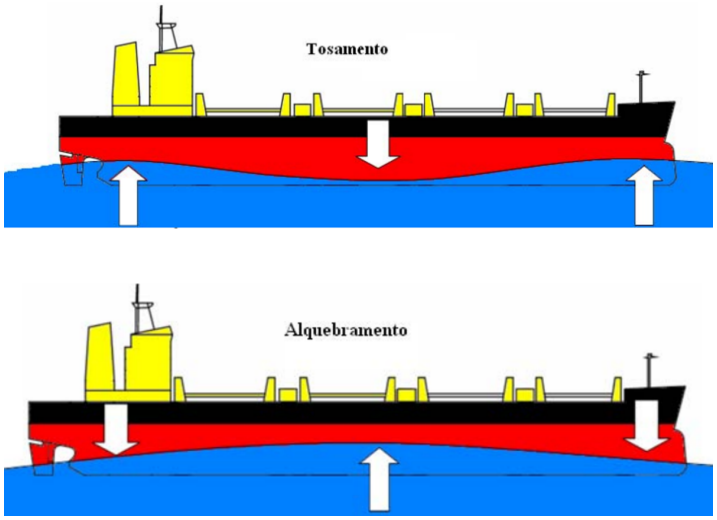


Figura 28 – Esquemático - Tosamento e Alquebramento.

Fonte: Hernández(13).

Para reproduzir esse carregamento no modelo foi escolhido aplicar momentos fletores nos **MPC constraints** definidos nos extremos do modelo. A **Figura 29** mostra o modelo montado com tais condições de contorno aparentes. Inicialmente o modelo está com todos os graus de liberdade de translação anulados e a rotação na direção z também é anulada (z corresponde ao vetor que acompanha o sentido popa proa, y aponta ao convés e x a bombordo - como mostra a **Figura 29**). O esquemático apresentado pela **Figura 30** mostra com mais claridade a descrição das condições de contorno aplicadas a fim de reproduzir um efeito de tosamento na estrutura do corpo paralelo médio modelado. O valor de momento fletor utilizado está sendo calculado a partir do equacionamento dado pela ABS (*American Bureau of Shipbuilding*) (7, 14) com relação ao momento de ondas à meia-nau. O equacionamento específico para tosamento (*sagging*) é apresentado a seguir.

$$M_{WS} = -k_1 \cdot C_1 \cdot L^2 \cdot B \cdot (C_b + 0.7) \times 10^{-3} \tag{1}$$

onde

$k_1 = 110$

$C_1 = 0.044 \cdot L + 3.75$ para $61 \leq L \leq 90$ m

$C_1 = 10.75 - \left(\frac{300-L}{100}\right)^{1.5}$ para $90 < L < 300$ m

$C_1 = 10.75$ para $61 < L < 90$ m

$C_1 = 10.75 - \left(\frac{L-350}{150}\right)^{1.5}$ para $300 \leq L \leq 500$ m

L = comprimento entre perpendiculares

B = boca da embarcação

C_b = coeficiente de bloco

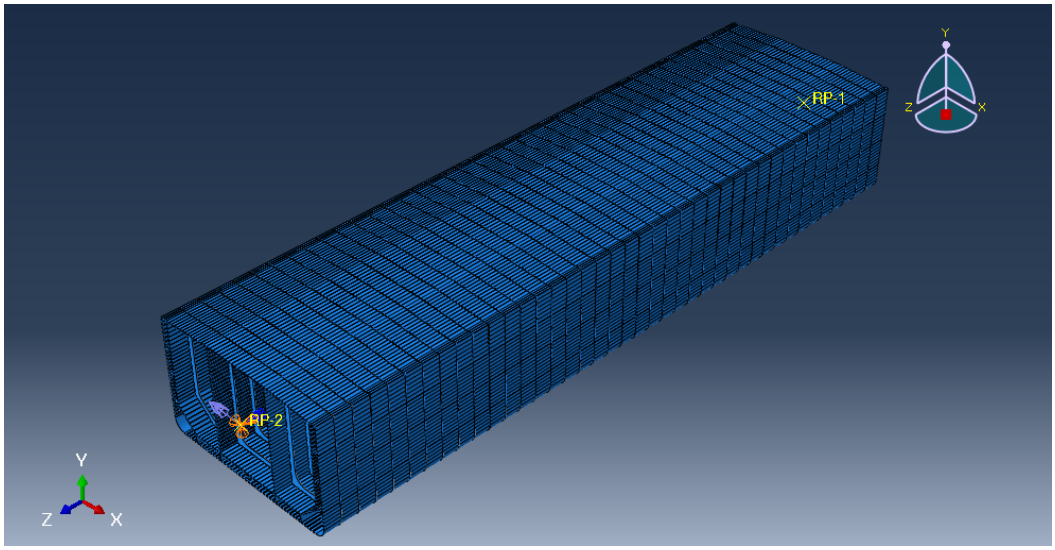


Figura 29 – Montagem - com condição de contorno visível.

Fonte: Elaboração Própria.

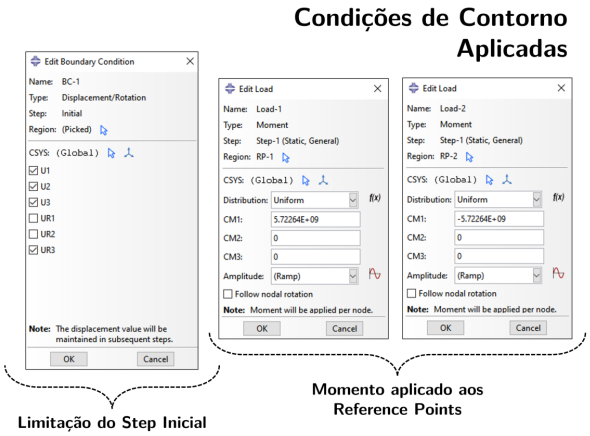


Figura 30 – Condições de Contorno - exemplo de funcionalidade.

Fonte: Elaboração Própria.

Existe um porém para trabalhar com a **Equação 1** mostrada acima com relação ao modelo concebido. O atual modelo retorna o corpo paralelo médio de uma embarcação e não o comprimento entre perpendiculares. Para contornar esse aspecto obteve-se uma proporção entre o comprimento do modelo e o LBP (*Length Between Perpendiculars)*).

Adotando uma hipótese de que o comprimento útil de carga - chamado de LC_{tk} é equivalente ao comprimento do corpo paralelo médio de uma embarcação, podemos usar proporções obtidas a partir de embarcações presentes na revista *Significant Ships* (15). A **Figura 31** fornece um diagrama mostrando a natureza das medições realizadas com as 24 embarcações avaliados da *Significant Ships*.

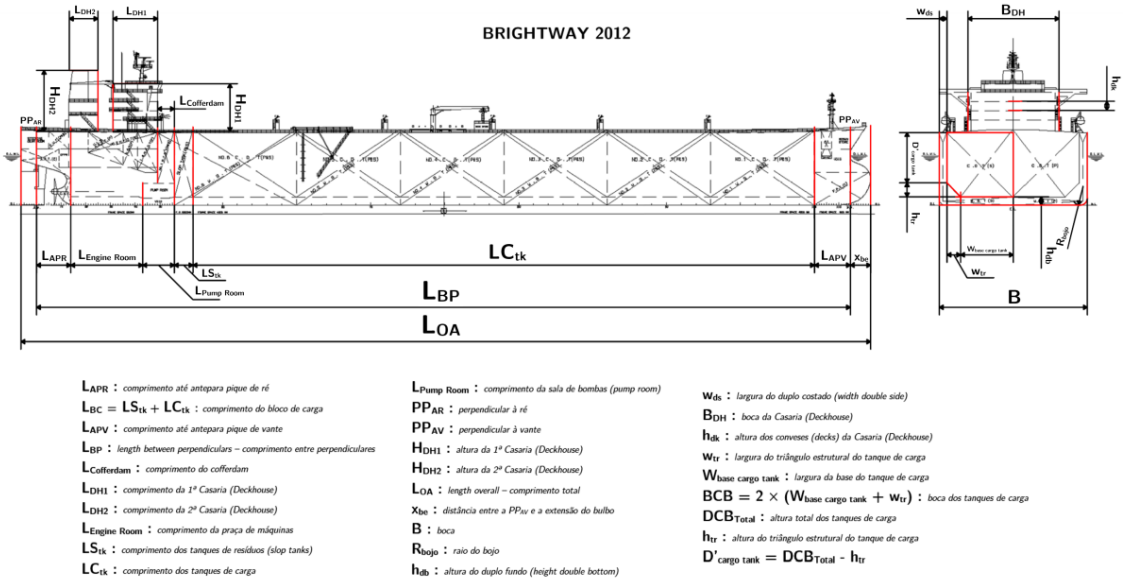


Figura 31 – Significant Ships - medições realizadas.

Fonte: Elaboração Própria.

Ao calcular a média dos valores obtidos nas medições, obtemos um coeficiente de aproximadamente 0.756, esse foi o valor utilizado no atual código para realizar a conversão entre comprimento do corpo paralelo médio para comprimento entre perpendiculares. As condições de contorno ficaram programadas entre as linhas 249 a 297 do código no **Apêndice 8**.

Até um certo limite inferior de comprimento do corpo paralelo médio a avaliação é feita via aplicação do momento da pela **Equação 1**, se o comprimento for inferior é aplicado uma rotação definida pelo usuário (parâmetro $\text{Applied}_{\text{rotation}}$) (linhas 295 a 297 do código no **Apêndice 8**).

Foram configurados também os *outputs* do modelo para poder avaliar algumas forças específicas em pontos de reação e principalmente requisitar a massa do modelo após a execução. As configurações aplicadas são entre as linhas 299 e 303 do código no **Apêndice 8**.

3.1 Exploração da Saída do Modelo

Para extrair dados de pontos específicos do modelo automaticamente e exportar vistas dos *field Outputs* dos modelos sem a necessidade de utilizar a interface gráfica do Abaqus, foi concebida uma função. A função MIDSHIP SECTION OUTPUT faz a leitura do arquivo de saída da simulação - formato .odb - e após sua execução, exporta um arquivo texto com resultados relativos à massa da estrutura, tensão máxima de von Mises ao redor da meia-nau e outros dados relevantes do modelo. Além de extrair esses dados pontuais, tal função - linhas 391 a 531 do código **Apêndice 8** - faz a exportação de todas as vistas do modelo não deformado e os *Field Outputs* relativos à tensão e deformação do modelo.

A **Figura 32** apresenta um exemplo de saída do arquivo texto comentado anteriormente e as **Figuras 33** e **34** apresentam respectivamente - para um modelo executado - a vista isométrica da estrutura e o campo de tensões de von Mises com a geometria deformada.

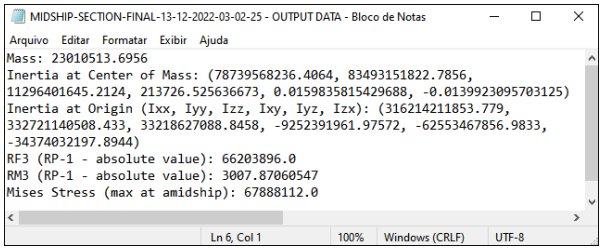


Figura 32 – Arquivo texto de saída da função exportações de dados.

Fonte: Elaboração Própria.

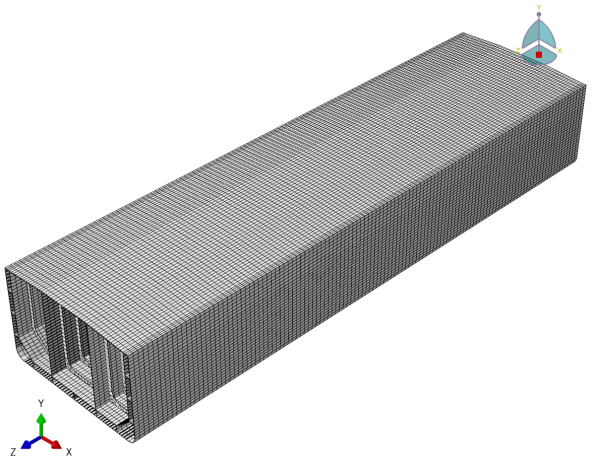


Figura 33 – Vista isométrica do modelo não deformado.

Fonte: Elaboração Própria.

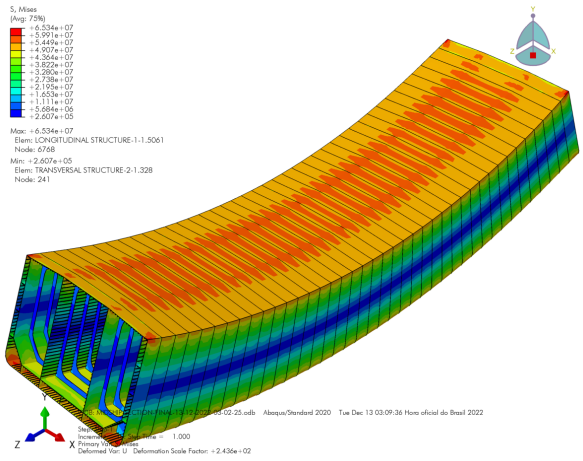


Figura 34 – Modelo deformado - campo de tensões de von Mises.

Fonte: Elaboração Própria.

3.2 Ensaios Múltiplos - **Apêndice 9**

Após a concepção do geral do modelo, foi possível realizar um ensaio avaliando aspectos estruturais em função da variação de apenas um parâmetro. Com os arquivos prontos - *PARAMETERS.py* (exportação de arquivo texto com os parâmetros - **Apêndice 5**), *GEOMETRY.py* (exportação dos pontos da geometria - **Apêndice 7**), *FUNCTIONS.py* (contém todas as funções descritas na **Seção 2.2** - **Apêndice 6**) e *MIDSHIP SECTION FINAL.py* (arquivo de construção do modelo no Abaqus - **Apêndice 8**) - foi criado um arquivo *MAIN.py* - **Apêndice 9** - responsável por ordenadamente executar arquivo a arquivo listado acima para a execução dos modelos. O arquivo *MAIN.py* realiza as seguintes ações ordenadamente - executando de forma múltipla, variando o espaçamento entre reforçadores:

- 1 - Executa (no Python 3) o arquivo *PARAMETERS.py* para garantir a existência de um arquivo texto com parâmetros
- 2 - Importa a função Return Parameters File do arquivo *PARAMETERS.py*
- 3 - Importa tudo do arquivo *GEOMETRY.py*
- 4 - Importa a biblioteca time - monitoramento do tempo de execução
- 5 - Cria uma lista contendo valores os quais o modelo será redefinido a cada iteração
- 6 - Dentro do *loop*:
  - 6.1 - Marca o início do tempo de processamento
  - 6.2 - Redefine o dicionário parameters com o valor proveniente do *loop*
  - 6.3 - Executa a função Return Parameters File com o dicionário modificado para refazer o arquivo de parâmetros
  - 6.4 - Executa o arquivo *GEOMETRY.py* que exporta os pontos em função dos parâmetros ajustados
  - 6.5 - Importa a biblioteca os (*operational system*)
  - 6.6 - Executa o arquivo *MIDSHIP SECTION FINAL.py* no Abaqus CAE
  - 6.7 - Marca o fim do tempo de processamento
  - 6.8 - Imprime o tempo de execução

Utilizando a configuração do arquivo *MAIN.py* (**Apêndice 9**) foram realizadas as execuções dos modelos variando o espaçamento geral - em todas as áreas de forma igual (fundo, bojo, costado, convés e antepara longitudinal). Ao todo **foram executados 17 modelos**, mantendo configurações gerais de dimensão e chapeamento e variando apenas o espaçamento entre os reforçadores - os resultados são apresentados a seguir. É importante mencionar, para fins de comparação, a máquina em que esses ensaios foram realizados: **Notebook Samsung 300E5M - HDD 1Tb - Processador Intel Core i5 7th Generation e placa de vídeo dedicada NVIDIA GeForce 920MX**.

Os parâmetros usados usados na definição da geometria usada para a realização dos ensaios paramétricos são apresentados no código presente no **Apêndice 5** e a **Figura 4** - apresentada anteriormente no presente estudo - é justamente a geometria adotada para o ensaio, em um caso particular onde os espaçamentos entre reforçadores longitudinais são iguais a 1 metro.

3.3 Exemplo de Execução Paramétrica

A seguir serão apresentados quatro gráficos demonstrando uma aplicação do modelo em avaliar a variação da tensão máxima da estrutura em função de outras variáveis. Como mencionado na seção anterior, a execução paramétrica foi realizada em função do espaçamento (em metros - todas unidade no Sistema Internacional) dos reforçadores - tal espaçamento foi mantido o mesmo

para todas as áreas que contém reforçadores longitudinais no modelo, ou seja, fundo, costado, bojo, antepara longitudinal e convés.

O tempo de execução dos modelos apresentou um comportamento que se ajustou muito bem a uma função do tipo potência - **Figura 35**. Esse decrescimento potencial do tempo de execução, com possíveis vistas de convergência para maiores valores entre reforçadores faz sentido, uma vez que em algum momento o espaçamento será superior ao tamanho da região em que tais reforços são aplicados. Dessa forma, o número de reforçadores tenderá a zero, porém a complexidade de execução do modelo, mesmo sem reforçadores menores, permanece - mostrando assim o porque do valor não decrescer continuamente.

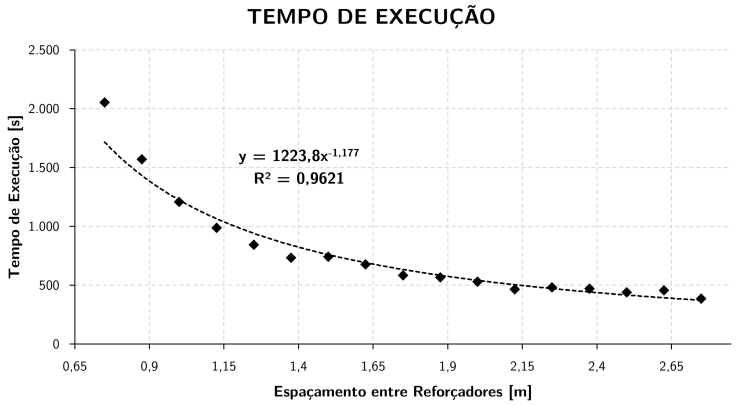


Figura 35 – Tempo de Execução em função do espaçamento de reforçadores.

Fonte: Elaboração Própria.

A mesma avaliação de convergência para um modelo sem reforçadores pode ser aplicada para o caso da massa da estrutura - **Figura 36**. Mesmo que não tenha apresentado um comportamento assintótico tão evidente quanto o tempo de execução, a massa da estrutura - na presente forma de execução paramétrica dos modelos - tenderá inevitavelmente ao valor da estrutura livre de reforçadores menores, composta apenas de chapeamentos, antepara longitudinal e longitudinais maiores do fundo e costado.

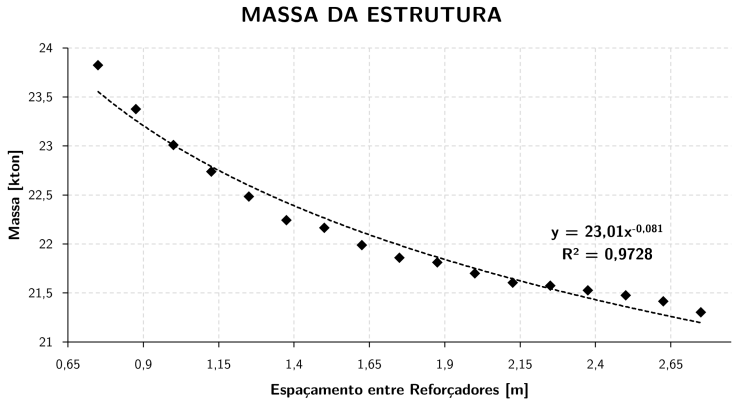


Figura 36 – Massa da estrutura em função do espaçamento de reforçadores.

Fonte: Elaboração Própria.

Também pode-se avaliar a evolução da tensão máxima a meia-nau de forma análoga - **Figura 37**, uma vez que para um espaçamento tendendo ao infinito, o número de reforçadores entre um intervalo finito tende a zero. Dessa forma - mesmo que a função tenha sido muito bem recuperada por uma expressão quadrática - os valores finais do gráfico evidenciam claramente a formação de um patamar, confirmando assim os limites propostos - um espaçamento tendendo a zero implica na construção de um arranjo maciço com espessura geral igual a soma das dimensões de altura dos reforçadores e, um espaçamento tendendo ao infinito implica em uma estrutura completamente livre de reforçadores menores.

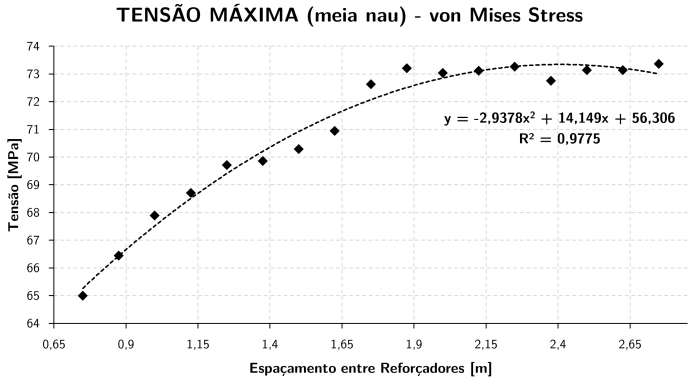


Figura 37 – Tensão máxima (von Mises) a meia-nau em função do espaçamento de reforçadores.

Fonte: Elaboração Própria.

O último gráfico construído - dado na Figura 38 - aponta um comportamento linear entre aumento da massa da estrutura e tensão máxima à meia-nau. Isso pode ser interessante em termos de avaliação financeira, uma vez que um aumento linear na tensão máxima - dependendo do fator de segurança - pode ser relevado em detrimento de uma queda, talvez exponencial, de custos de projeto.

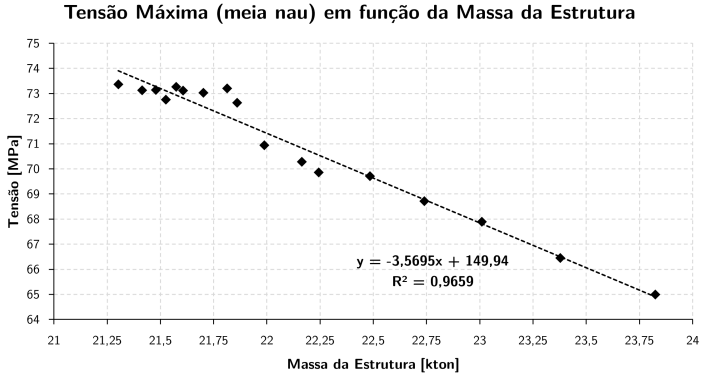


Figura 38 – Tensão máxima (von Mises) a meia-nau em função da massa da estrutura.

Fonte: Elaboração Própria.

Esses gráficos apresentados e também a presente seção, como um todo, são apenas um demonstrativo de aplicação do modelo de seção mestra concebido. O presente estudo já atingiu os objetivos planejados, uma vez que um modelo funcional foi concebido e se mostrou confiável em termos de execução.

4 Conclusões

O presente estudo teve êxito com relação a seu objetivo inicial, ou seja, foi capaz de obter com sucesso um modelo completamente funcional de um corpo paralelo médio de uma embarcação de transporte de petróleo. **O modelo obtido possui**, ao final de sua concepção, **81 parâmetros** - dentre eles os escantilhões da embarcação, dimensões dos reforçadores do fundo, costado, bojo, convés e antepara longitudinal, número de elementos transversais a serem aplicados e definições do material e malha a serem aplicados.

A condição de contorno proposta nos resultados se mostrou adequada para exemplificação da capacidade de aplicação do modelo que o presente estudo concebeu. A avaliação paramétrica apresentada na seção anterior, mostrando a variação de massa e tensão em função do espaçamento dos elementos reforçadores longitudinais do fundo, costado, convés, bojo e antepara longitudinal, apresentou com sucesso um ponto de vista que pode ser usado na decisão de arranjo estrutural de uma embarcação - avaliando sempre a redução de custos materiais e de fabricação relativo à segurança estrutural.

Mesmo sendo um modelo pesado computacionalmente, seu tempo de execução não excede o tempo esperado de um modelo em elementos finitos com tal complexidade - como pôde ser observado no gráfico plotado na seção passada - **Figura 35**. A estratégia de modelagem usada, tanto quanto as funções desenvolvidas para a construção da geometria e técnicas aplicadas ao particionamento da estrutura podem muito bem ser aplicadas em outros estudos futuros, ou ainda, um modelo de seção de embarcação ainda mais completo e próximo à realidade observada nos projetos navais da atualidade.

O modelo construído têm diversas possíveis aplicações, desde apenas um auxílio a decisão de projeto, até uma integração completa de execução com algum algoritmo de otimização para escolha de uma geometria em função de parâmetros estruturais de saída (tensão máxima na quilha por exemplo).

Com um modelo paramétrico é possível facilmente executar toda uma população de corpos paralelos médios - variando n parâmetros - e no final realizar algum tipo de estudo de aprendizado de máquina e inteligência artificial relativo à previsão de aspectos estruturais sem a necessidade de utilizar propriamente o método dos elementos finitos.

Por fim é importante mencionar que o presente estudo deixa mais claro a utilização do software Abaqus CAE através da programação em linguagem Python. Essa possibilidade é fundamental para execução paramétrica de qualquer modelo concebido, deixando a cargo do computador a realização de todo o pré e pós processamento de dados - deixando apenas a cargo do pesquisador a tarefa de avaliar criticamente os resultados obtidos.

Tratando do tópico de Simulação Estrutural, a fase de modelagem dos arranjos estruturais costumam ocupar boa parte do tempo dedicado a esse tipo de estudo. Com a possibilidade e conceber um modelo que é flexível, as avaliações críticas das simulações acabam sendo o foco principal, implicando diretamente em projetos mais bem otimizados e eficientes em termos estruturais.

Referências

1 YAO, T. Hull girder strength. *Marine Structures*, v. 16, p. 1–13, 01 2003.

2 BASTOS, I. V. *ANÁLISE ECONÔMICA E OPERACIONAL DE UM PETROLEIRO SUEZMAX*. Tese (Doutorado), 02 2017. Disponível em: <http://repositorio.poli.ufrj.br/monografias/monopoli10020570.pdf>.

3 GORDO, J. Compressive strength of double-bottom under alternate hold loading condition. *Progress in the Analysis and Design of Marine Structures*, 04 2017.

4 149.89.49. *Abaqus Scripting User's Guide (6.14)*. Dassault Sustèmes, 2022. Disponível em: <http://130.149.89.49:2080/v6.14/books/cmd/default.htm>.

5 PURI, G. M. *Python scripts for Abaqus : learn by example*. [S.l.]: Puri, 2011.

6 TAYYAR, G. T. Overall hull girder nonlinear strength monitoring based on inclinometer sensor data. *International Journal of Naval Architecture and Ocean Engineering*, v. 12, p. 902–909, 2020.

7 ABS, A. B. o. S. *Rules for Building and Classing Marine Vessels Part 3 Hull Construction and Equipment*. 2022. Disponível em: <https://ww2.eagle.org/content/dam/eagle/rules-and-guides/current/other/1_marinevesselrules_2022/mvr-part-3-july22.pdf>.

8 , L. R. . *Rules and Regulations for the Classification of Ships*. 2020. Disponível em: <https://www.lr.org/en/rules-and-regulations-for-the-classification-of-ships/>.

9 SALAZAR-DOMíNGUEZ, C. M. et al. Structural analysis of a barge midship section considering the still water and wave load effects. *Journal of Marine Science and Engineering*, v. 9, p. 99, 01 2021.

10 ANDRIć, J.; KITAROVIC, S.; PIRIC, K. Residual hull girder ultimate strength of double hull oil tankers. *Brodogradnja*, v. 66, p. 1–13, 09 2015.

11 KIM, D. K. et al. Ultimate strength performance of tankers associated with industry corrosion addition practices. *International Journal of Naval Architecture and Ocean Engineering*, v. 6, p. 507–528, 09 2014.

12 , E. N. G. P. . *Design and Construction of Oil Tankers Enbridge Northern Gateway Project Design and Construction of Oil Tankers*. 2012. Disponível em: <https://ceaa-acee.gc.ca/050/documents_staticpost/cearref_21799/4234/Attachment_12.pdf>.

13 *FLAMBAGEM TORCIONAL DE ENRIJECEDORES EM PAINÉIS DE NAVIOS TANQUES*. Disponível em: <https://www.researchgate.net/publication/301691173_FLAMBAGEM_TORCIONAL_DE_ENRIJECEDORES_EM_PAINEIS_DE_NAVIOS_TANQUES>.

14 HERMANN, M. P. *UNIVERSIDADE FEDERAL DE SANTA CATARINA CENTRO TECNOLÓGICO DE JOINVILLE CURSO DE ENGENHARIA DE NAVAL*. 2016. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/171658/Otimiza\%C3\%A7\%C3\%A3o\%20de\%20projeto\%20estrutural\%20para\%20embarca\%C3\%A7\%C3\%B5es\%20de\%20grande\%20porte.pdf?sequence=1&isAllowed=y>.

15 WWW.RINA.ORG.UK. *Significant Ships*. Disponível em: <https://www.rina.org.uk/sigships.html>.

5 Apêndice - arquivo *PARAMETERS.py*

```python
# PARAMETROS DEFINIDORES DA GEOMETRIA
# OBS.: escolher um sistema de unidades e manter sobre todo modelo (ex.: SI Standard (m, kg, Pa, ...))

spacing = 1.00

parameters = {'n_anteparas'         :  3    ,
              'n_cavernas'          :  9    ,
              'L_longframe'         :  5.0  ,
              'B_tkBBBE'            : 15.0  ,
              'R_bojo'              :  3.0  ,
              'h_db'                :  2.5  ,
              'w_ds'                :  2.0  ,
              'dh_deck'             :  1.0  ,
              't_longc'             :  0.0254,
              't_longf'             :  0.0254,
              't_quilha'            :  0.0254,
              't_antlong'           :  0.0254,
              't_caverna'           :  0.0254,
              't_bulkhead'          :  0.0254,
              't_chapint'           :  0.0254,
              't_chapext'           :  0.0254,
              't_chapdeck'          :  0.0254,
              'spacing_bottom'      :  spacing ,
              'spacing_side'        :  spacing ,
              'spacing_deck'        :  spacing ,
              'spacing_bojo'        :  spacing ,
              'K3'                  :  9.0  ,
              'K4'                  : 10.0  ,
              'K5'                  :  8.0  ,
              'K6'                  :  8.0  ,
              'K7'                  :  8.0  ,
              'a_RCD'               :  0.35 ,
              'b_RCD'               :  0.25 ,
              't_RCD'               :  0.0254,
              'h_RCD'               :  0.0254,
              'nRef_cost'           :  2    ,
              'Ref_cost_type'       : 'T'   ,
              'aRef_cost'           :  0.25 ,
              'bRef_cost'           :  0.15 ,
              'tRef_cost'           :  0.0254,
              'hRef_cost'           :  0.0254,
              'Ref_bottom_type'     : 'T'   ,
              'a_bottom'            :  0.25 ,
              'b_bottom'            :  0.15 ,
              't_bottom'            :  0.0127,
              'h_bottom'            :  0.0127,
              'nRef_q'              :  2    ,
              'Ref_q_type'          :  'L1' ,
              'a_q'                 :  0.25 ,
              'b_q'                 :  0.15 ,
              't_q'                 :  0.0127,
              'h_q'                 :  0.0127,
              'Ref_Lside_type'      :  'L2' ,
              'Ref_Rside_type'      :  'L1' ,
              'a_side'              :  0.25 ,
              'b_side'              :  0.15 ,
              't_side'              :  0.0127,
              'h_side'              :  0.0127,
              'Ref_deck_type'       :  'T'  ,
              'a_deck'              :  0.25 ,
              'b_deck'              :  0.15 ,
              't_deck'              :  0.0127,
              'h_deck'              :  0.0127,
              'nRI_al'              :  6    ,
              'RI_al_type'          :  'T'  ,
              'aRI_al'              :  0.25 ,
              'bRI_al'              :  0.15 ,
              'tRI_al'              :  0.0127,
              'hRI_al'              :  0.0127,
              'nRII_al'             :  4    ,
              'RII_al_type'         :  'L2' ,
              'aRII_al'             :  0.25 ,
              'bRII_al'             :  0.15 ,
              'tRII_al'             :  0.0127,
              'hRII_al'             :  0.0127,
              'dh_deckC'            :  2.0  ,
              'dh_sideC'            :  1.5  ,
              'dh_bottomC'          :  1.5  ,
              'dh_antlongC'         :  1.5  ,
              'R_caverna'           :  3.0  ,
              'Material_Name'       :  'STEEL',
              'E_Young'             :  210e9  ,
              'v_poisson'           :  0.33   ,
              'density'             :  7850.0 ,
              'applied_rotation'    :  0.045  ,
              'Average_element_size':  1.50   }

def Return_Parameters_File(dic_parameters):

    n_anteparas = dic_parameters['n_anteparas']   # numero de anteparas transversais a serem aplicadas ao modelo (int >= 0)
    n_cavernas  = dic_parameters['n_cavernas']    # numero de aneis (gigantes) a serem aplicados em cada trecho entre anteparas (int >= 0)

    L_longframe = dic_parameters['L_longframe']   # comprimento da secao de reforcos longitudinais entre os aneis transversais (gigantes)
    B_tkBBBE    = dic_parameters['B_tkBBBE']      # largura dos tanques de BB e BE
    R_bojo      = dic_parameters['R_bojo']        # raio do bojo da embarcacao
    h_db        = dic_parameters['h_db']          # altura do duplo fundo
    w_ds        = dic_parameters['w_ds']          # largura do duplo costado
    dh_deck     = dic_parameters['dh_deck']       # altura da inclinacao do conves central

    t_longc   = dic_parameters['t_longc']         # espessura da chapa das longitudinais maiores do costado
    t_longf   = dic_parameters['t_longf']         # espessura da chapa das longitudinais maiores do fundo
    t_quilha  = dic_parameters['t_quilha']        # espessura da chapa da quilha

    t_antlong  = dic_parameters['t_antlong']      # espessura da chapa das anteparas longitudinais
    t_caverna  = dic_parameters['t_caverna']      # espessura da chapa dos aneis transversais (gigantes)
    t_bulkhead = dic_parameters['t_bulkhead']     # espessura da chapa das anteparas transversais

    t_chapint  = dic_parameters['t_chapint']      # espessura do chapeamento interno (forro dos tanques)
    t_chapext  = dic_parameters['t_chapext']      # espessura do chapeamento externo da embarcacao
    t_chapdeck = dic_parameters['t_chapdeck']     # espessura do chapeamento do conves da embarcacao

    spacing_bottom = dic_parameters['spacing_bottom']   # espacamento dos reforcadores longitudinais menores do fundo
    spacing_side   = dic_parameters['spacing_side']     # espacamento dos reforcadores longitudinais menores do costado
    spacing_deck   = dic_parameters['spacing_deck']     # espacamento dos reforcadores longitudinais menores do conves
    spacing_bojo   = dic_parameters['spacing_bojo']     # espacamento dos reforcadores longitudinais menores do bojo e borboleta

    K3 = dic_parameters['K3']    # comprimento de fundo dos tanques BB e BE (resistido por longitudinais menores) - consultar figura
    K4 = dic_parameters['K4']    # comprimento de fundo do tanque central entre quilha (resistido por longitudinais menores) - consultar figura

    K5 = dic_parameters['K5']    # altura entre longitudinais maiores do costado (primeira secao - de baixo para cima) - consultar figura
    K6 = dic_parameters['K6']    # altura entre longitudinais maiores do costado (segunda secao - de baixo para cima) - consultar figura
    K7 = dic_parameters['K7']    # altura entre longitudinais maiores do costado (terceira secao - de baixo para cima) - consultar figura

    K1 = B_tkBBBE + w_ds + t_chapint - (R_bojo + K3 + t_longf) # altura entre a face inferior da primeira longitudinal maior do costado e o fim do raio do bojo - consultar
    K2 = B_tkBBBE + w_ds + t_chapint - (R_bojo + K3 + t_longf) # comprimento entre a face esquerda da primeira longitudinal maior do fundo e o fim do raio do bojo - consult

    B = 2*(t_chapext + w_ds + t_chapint + B_tkBBBE + t_antlong + K4)                    # Boca da embarcao modelada - formada pelos parametros definidos acima
```

```
    D = R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck + t_chapdeck      # Pontal da embarcacao modelada - formado pelos parametros definidos acima
    L_station  = ((n_cavernas + 1) * L_longframe) + (n_cavernas * t_caverna)                 # Comprimento da secao resistente entre anteparas transversais
    L = ((n_anteparas + 1) * L_station) + (n_anteparas * t_bulkhead)                         # Comprimento total do corpo paralelo medio definido

    # PARAMETROS DEFINIDORES DA GEOMETRIA DOS REFORCADORES MENORES

    # REFORCADOR LONGITUDINAL CENTRAL DO CONVES - FORMATO T FIXADO
    # RCD: Reforcador Central Deck
    a_RCD = dic_parameters['a_RCD']    # alma do RCD
    b_RCD = dic_parameters['b_RCD']    # flange do RCD
    t_RCD = dic_parameters['t_RCD']    # espessura da alma
    h_RCD = dic_parameters['h_RCD']    # espessura do flange

    # REFORCADOR DAS LONGITUDINAIS MAIORES DO COSTADO
    nRef_cost     = dic_parameters['nRef_cost']        # numero de reforcadores
    Ref_cost_type = dic_parameters['Ref_cost_type']    # tipo de reforcador a ser aplicado (T, L1 e L2 - consultar relatorio)
    aRef_cost     = dic_parameters['aRef_cost']        # alma
    bRef_cost     = dic_parameters['bRef_cost']        # flange
    tRef_cost     = dic_parameters['tRef_cost']        # espessura da alma
    hRef_cost     = dic_parameters['hRef_cost']        # espessura do flange

    # REFORCADORES LONGITUDINAIS MENORES DO FUNDO
    Ref_bottom_type = dic_parameters['Ref_bottom_type']   # tipo de reforcador a ser aplicado (T, L1 e L2 - consultar relatorio)
    a_bottom        = dic_parameters['a_bottom']          # alma
    b_bottom        = dic_parameters['b_bottom']          # flange
    t_bottom        = dic_parameters['t_bottom']          # espessura da alma
    h_bottom        = dic_parameters['h_bottom']          # espessura do flange

    # REFORCADORES MENORES ACOPLADOS A CHAPA DA QUILHA
    nRef_q     = dic_parameters['nRef_q']        # numero de reforcadores
    Ref_q_type = dic_parameters['Ref_q_type']    # tipo de reforcador a ser aplicado (T, L1 e L2 - consultar relatorio)
    a_q        = dic_parameters['a_q']           # alma
    b_q        = dic_parameters['b_q']           # flange
    t_q        = dic_parameters['t_q']           # espessura da alma
    h_q        = dic_parameters['h_q']           # espessura do flange

    # REFORCADORES LONGITUDINAIS MENORES DO COSTADO
    Ref_Lside_type = dic_parameters['Ref_Lside_type']   # tipo de reforcador a ser aplicado no lado esquerdo (T, L1 e L2 - consultar relatorio)
    Ref_Rside_type = dic_parameters['Ref_Rside_type']   # tipo de reforcador a ser aplicado no lado direito (T, L1 e L2 - consultar relatorio)
    a_side = dic_parameters['a_side']                   # alma
    b_side = dic_parameters['b_side']                   # flange
    t_side = dic_parameters['t_side']                   # espessura da alma
    h_side = dic_parameters['h_side']                   # espessura do flange

    # REFORCADORES LONGITUDINAIS DO CONVES
    Ref_deck_type = dic_parameters['Ref_deck_type']   # tipo de reforcador a ser aplicado (T, L1 e L2 - consultar relatorio)
    a_deck = dic_parameters['a_deck']                 # alma
    b_deck = dic_parameters['b_deck']                 # flange
    t_deck = dic_parameters['t_deck']                 # espessura da alma
    h_deck = dic_parameters['h_deck']                 # espessura do flange

    # REFORCADORES DA ANTEPARA LONGITUDINAL

    # REFORCADOR MAIOR DA ANTEPARA LONGITUDINAL
    nRI_al     = dic_parameters['nRI_al']        # numero de reforcadores maiores
    RI_al_type = dic_parameters['RI_al_type']    # tipo de reforcador a ser aplicado (T, L1 e L2 - consultar relatorio)
    aRI_al     = dic_parameters['aRI_al']        # alma
    bRI_al     = dic_parameters['bRI_al']        # flange
    tRI_al     = dic_parameters['tRI_al']        # espessura da alma
    hRI_al     = dic_parameters['hRI_al']        # espessura do flange

    # REFORCADOR MENOR DA ANTEPARA LONGITUDINAL
    nRII_al     = dic_parameters['nRII_al']       # numero de reforcadores menores
    RII_al_type = dic_parameters['RII_al_type']   # tipo de reforcador a ser aplicado (T, L1 e L2 - consultar relatorio)
    aRII_al     = dic_parameters['aRII_al']       # alma
    bRII_al     = dic_parameters['bRII_al']       # flange
    tRII_al     = dic_parameters['tRII_al']       # espessura da alma
    hRII_al     = dic_parameters['hRII_al']       # espessura do flange

    # GIGANTES (ANEIS TRANSVERSAIS - hastilha + caverna + vau)
    dh_deckC    = dic_parameters['dh_deckC']      # altura do vau (variacao de altura relativa ao conves - consultar relatorio)
    dh_sideC    = dic_parameters['dh_sideC']      # variacao de largura relativa ao costado - consultar relatorio
    dh_bottomC  = dic_parameters['dh_bottomC']    # variacao de largura relativa ao fundo - consultar relatorio
    dh_antlongC = dic_parameters['dh_antlongC']   # variacao de largura relativa a antepara longitudinal
    R_caverna   = dic_parameters['R_caverna']     # raio de arredondamento dos cantos presentes no reforcador transversal (gigante)

    # DEFINICAO DO MATERIAL
    Material_Name = dic_parameters['Material_Name']   # nome do material (formalismo apenas)
    E_Young       = dic_parameters['E_Young']         # modulo de elasticidade do material
    v_poisson     = dic_parameters['v_poisson']       # coeficiente de Poisson do material
    density       = dic_parameters['density']         # densidade do material (adequado as demais unidades usadas)

    # ROTACAO APLICADA PARA A SIMULACAO EXEMPLO DEFINIDA
    applied_rotation = dic_parameters['applied_rotation']       # rotacao (radianos) - consultar relatÃ³rio para mais detalhes

    # MALHA DO MODELO
    Average_element_size = dic_parameters['Average_element_size']     # tamanho medio dos elementos para os componentes estruturais definidos

    # 87 PARAMETROS
    list_locals = list(locals())

    with open('MIDSHIP SECTION - PARAMETERS.txt', 'w') as archive:
        for item in list_locals:
            if '__' not in item and item != 'dic_parameters':
                archive.write(f"{item} {locals()[item]} {str(type(locals()[item])).replace('<class ','').replace('>','')}\n")

Return_Parameters_File(parameters)
```

6 Apêndice - arquivo *FUNCTIONS.py*

```python
def Space_Division(pA, pB, n):

    import numpy as np

    xA, yA = pA
    xB, yB = pB

    D     = ((xB - xA)**2 + (yB - yA)**2)**0.5
    alpha = np.arctan2((yB - yA),(xB - xA))

    list_points = []

    if yB >= yA:
        for i in range(n):
            list_points.append((xA + (i+1)*(D/(n+1))*np.cos(alpha),
                                yA + (i+1)*(D/(n+1))*np.sin(alpha)))

    elif yA > yB:

        alpha = alpha - np.pi

        for i in range(n):
            list_points.append((xA - (i+1)*(D/(n+1))*np.cos(alpha),
                                yA - (i+1)*(D/(n+1))*np.sin(alpha)))

    return list_points

def T_Reinforcement_Points(a, b, t, h, t_base, origin_tuple, theta1):

    import numpy as np

    k1, k2 = origin_tuple

    p0 = (k1,k2)
    p1 = ((t/2)*np.cos(theta1) + k1, (-t/2)*np.sin(theta1) + k2)
    p2 = ((-t/2)*np.cos(theta1) + k1, (t/2)*np.sin(theta1) + k2)
    p3 = (-a*np.sin(theta1) - (t/2)*np.cos(theta1) + k1, -(a*np.cos(theta1) -(t/2)*np.sin(theta1)) + k2)
    p4 = (-((b-t)/2)*np.cos(theta1) -a*np.sin(theta1) - (t/2)*np.cos(theta1) + k1,
           ((b-t)/2)*np.sin(theta1) -(a*np.cos(theta1) -(t/2)*np.sin(theta1)) + k2)
    p5 = (-h*np.sin(theta1) -((b-t)/2)*np.cos(theta1) -a*np.sin(theta1) - (t/2)*np.cos(theta1) + k1,
          -h*np.cos(theta1) + ((b-t)/2)*np.sin(theta1) -(a*np.cos(theta1) -(t/2)*np.sin(theta1)) + k2)
    p6 = (-h*np.sin(theta1) +((b-t)/2)*np.cos(theta1) -a*np.sin(theta1) + (t/2)*np.cos(theta1) + k1,
          -h*np.cos(theta1) - ((b-t)/2)*np.sin(theta1) -a*np.cos(theta1) -(t/2)*np.sin(theta1) + k2)
    p7 = (((b-t)/2)*np.cos(theta1) -a*np.sin(theta1) + (t/2)*np.cos(theta1) + k1,
         -((b-t)/2)*np.sin(theta1) -a*np.cos(theta1) -(t/2)*np.sin(theta1) + k2)
    p8 = (-a*np.sin(theta1) + (t/2)*np.cos(theta1) + k1, -a*np.cos(theta1) -(t/2)*np.sin(theta1) + k2)

    p1L = (p1[0] + t_base*np.sin(theta1), p1[1] + t_base*np.cos(theta1))
    p2L = (p2[0] + t_base*np.sin(theta1), p2[1] + t_base*np.cos(theta1))
    p5L = (p5[0] + ((b-t)/2)*np.cos(theta1), p5[1] - ((b-t)/2)*np.sin(theta1))
    p6L = (p6[0] - ((b-t)/2)*np.cos(theta1), p6[1] + ((b-t)/2)*np.sin(theta1))

    list_points = [p2, p3, p4, p5, p6, p7, p8, p1]

    x = [x_value for x_value,y_value in list_points]
    y = [y_value for x_value,y_value in list_points]

    list_partition = [(p1, p2), (p3, p8), (p3, p5L), (p8, p6L), (p1, p1L), (p2, p2L)]

    return x, y, list_partition

def L1_Reinforcement_Points(a, b, t, h, t_base, origin_tuple, theta1):

    import numpy as np

    k1, k2 = origin_tuple

    p0 = (k1,k2)
    p1 = ((t/2)*np.cos(theta1) + k1, (-t/2)*np.sin(theta1) + k2)
    p2 = ((-t/2)*np.cos(theta1) + k1, (t/2)*np.sin(theta1) + k2)
    p3 = ((-t/2)*np.cos(theta1)  - (a + h)*np.sin(theta1) + k1,
          (t/2)*np.sin(theta1) - (a + h)*np.cos(theta1) + k2)
    p4 = ((-t/2)*np.cos(theta1)  - (a + h)*np.sin(theta1) + b*np.cos(theta1) + k1,
          (t/2)*np.sin(theta1) - (a + h)*np.cos(theta1) - b*np.sin(theta1) + k2)
    p5 = ((-t/2)*np.cos(theta1)  - (a + h)*np.sin(theta1) + b*np.cos(theta1) + h*np.sin(theta1) + k1,
          (t/2)*np.sin(theta1) - (a + h)*np.cos(theta1) - b*np.sin(theta1) + h*np.cos(theta1) + k2)
    p6 = ((t/2)*np.cos(theta1) - a*np.sin(theta1) + k1, (-t/2)*np.sin(theta1) - a*np.cos(theta1) + k2)

    p1L = (p1[0] + t_base*np.sin(theta1), p1[1] + t_base*np.cos(theta1))
    p2L = (p2[0] + t_base*np.sin(theta1), p2[1] + t_base*np.cos(theta1))
    p3L = (p3[0] + h*np.sin(theta1), p3[1] + h*np.cos(theta1))
    p4L = (p3[0] + t*np.cos(theta1), p3[1] - t*np.sin(theta1))

    list_points = [p2, p3, p4, p5, p6, p1]

    x = [x_value for x_value,y_value in list_points]
    y = [y_value for x_value,y_value in list_points]

    list_partition = [(p1, p2), (p3L, p6), (p4L, p6), (p1, p1L), (p2, p2L)]

    return x, y, list_partition

def L2_Reinforcement_Points(a, b, t, h, t_base, origin_tuple, theta1):
```

```
94
95      import numpy as np
96
97      k1, k2 = origin_tuple
98
99      p0 = (k1,k2)
100     p1 = ((t/2)*np.cos(theta1) + k1, (-t/2)*np.sin(theta1) + k2)
101     p2 = ((-t/2)*np.cos(theta1) + k1, (t/2)*np.sin(theta1) + k2)
102     p3 = ((-t/2)*np.cos(theta1)  - a*np.sin(theta1) + k1,
103           (t/2)*np.sin(theta1) - a*np.cos(theta1) + k2)
104     p4 = ((-t/2)*np.cos(theta1)  - a*np.sin(theta1) - (b - t)*np.cos(theta1) + k1,
105           (t/2)*np.sin(theta1) - a*np.cos(theta1) + (b - t)*np.sin(theta1) + k2)
106     p5 = ((t/2)*np.cos(theta1) - (a + h)*np.sin(theta1) - b*np.cos(theta1) + k1,
107           (-t/2)*np.sin(theta1) - (a + h)*np.cos(theta1) + b*np.sin(theta1) + k2)
108     p6 = ((t/2)*np.cos(theta1) - (a + h)*np.sin(theta1) + k1,
109           (-t/2)*np.sin(theta1) - (a + h)*np.cos(theta1) + k2)
110
111     p1L = (p1[0] + t_base*np.sin(theta1), p1[1] + t_base*np.cos(theta1))
112     p2L = (p2[0] + t_base*np.sin(theta1), p2[1] + t_base*np.cos(theta1))
113     p5L = (p6[0] - t*np.cos(theta1), p6[1] + t*np.sin(theta1))
114     p6L = (p6[0] + h*np.sin(theta1), p6[1] + h*np.cos(theta1))
115
116     list_points = [p2, p3, p4, p5, p6, p1]
117
118     x = [x_value for x_value,y_value in list_points]
119     y = [y_value for x_value,y_value in list_points]
120
121     list_partition = [(p1, p2), (p3, p6L), (p5L, p3), (p1, p1L), (p2, p2L)]
122
123     return x, y, list_partition
124
125 def Reinforcement_Main(pA, pB, n, reinforcement_type, a, b, t, h, t_base, angle_variation):
126
127     import numpy as np
128
129     xA, yA = pA
130     xB, yB = pB
131
132     alpha = np.arctan2((xB - xA),(yB - yA))
133
134     if round(yA,2) > round(yB,2):
135         alpha = alpha - np.pi
136
137     centers_list = Space_Division(pA, pB, n)
138
139     x_list = [xA]
140     y_list = [yA]
141     all_partitions = []
142
143     for x,y in centers_list:
144
145         if reinforcement_type == 'T':
146             x_values, y_values, list_partition = T_Reinforcement_Points(a, b, t, h, t_base, (x,y), alpha + angle_variation)
147             x_list += x_values
148             y_list += y_values
149             all_partitions += list_partition
150
151         elif reinforcement_type == 'L1':
152             x_values, y_values, list_partition = L1_Reinforcement_Points(a, b, t, h, t_base, (x,y), alpha + angle_variation)
153             x_list += x_values
154             y_list += y_values
155             all_partitions += list_partition
156
157         elif reinforcement_type == 'L2':
158             x_values, y_values, list_partition = L2_Reinforcement_Points(a, b, t, h, t_base, (x,y), alpha + angle_variation)
159             x_list += x_values
160             y_list += y_values
161             all_partitions += list_partition
162
163         else:
164             break
165
166     x_list.append(xB)
167     y_list.append(yB)
168
169     return x_list, y_list, all_partitions
170
171 def Cartesian_Distance(pA, pB):
172
173     xA, yA = pA
174     xB, yB = pB
175
176     return ((xB - xA)**2 + (yB - yA)**2)**0.5
177
178 def Return_Middle(pA, pB):
179
180     xA, yA = pA
181     xB, yB = pB
182
183     return ((xA + xB)/2, (yA + yB)/2)
184
185 def y_from_x_Parallel_Line(x, pA, pB, t, theta):
186
187     import numpy as np
```

```
188
189     xA, yA = pA
190     xB, yB = pB
191
192     a = (yA - yB) / (xA - xB)
193     b = yA - ((yA - yB) / (xA - xB))*xA
194
195     y = (a*x) + b + (t/np.cos(theta))
196
197     return y
198
199 def x_from_y_Parallel_Line(y, pA, pB, t, theta):
200
201     import numpy as np
202
203     xA, yA = pA
204     xB, yB = pB
205
206     a = (yA - yB) / (xA - xB)
207     b = yA - ((yA - yB) / (xA - xB))*xA
208
209     x = (y - b - (t/np.cos(theta)))/a
210
211     return x
212
213 def Sided_Points(pA, pB, P, t):
214
215     import numpy as np
216
217     xA, yA = pA
218     xB, yB = pB
219     x, y = P
220
221     alpha = np.arctan2((yB - yA),(xB - xA))
222
223     list_points = []
224
225     if yB >= yA:
226         pI  = (x - (t/2)*np.cos(alpha), y - (t/2)*np.sin(alpha))
227         pII = (x + (t/2)*np.cos(alpha), y + (t/2)*np.sin(alpha))
228         list_points.append(pI)
229         list_points.append(pII)
230
231     elif yA > yB:
232
233         alpha = alpha - np.pi
234
235         pI  = (x - (t/2)*np.cos(alpha), y - (t/2)*np.sin(alpha))
236         pII = (x + (t/2)*np.cos(alpha), y + (t/2)*np.sin(alpha))
237         list_points.append(pI)
238         list_points.append(pII)
239
240     return list_points
241
242 def grouper(iterable, n, fillvalue=None):
243
244     from itertools import zip_longest
245
246     args = [iter(iterable)] * n
247
248     return zip_longest(*args, fillvalue=fillvalue)
249
250 def mirrorImage( a, b, c, x1, y1):
251     temp = -2 * (a * x1 + b * y1 + c) /(a * a + b * b)
252     x = temp * a + x1
253     y = temp * b + y1
254     return (x, y)
255
256 def Double_Reinforcement_Main(pA, pB,
257                               RefI_type , n_RI , a_RI , b_RI , t_RI , h_RI , t_base_RI , RI_angle_var,
258                               RefII_type, n_RII, a_RII, b_RII, t_RII, h_RII, t_base_RII, RII_angle_var, RII_ab_order):
259
260     import numpy as np
261
262     list_gen = [pA]
263     for P in Space_Division(pA, pB, n_RI):
264         list_gen += Sided_Points(pA, pB, P, t_RI)
265     list_gen.append(pB)
266
267     x_gen_list = []
268     y_gen_list = []
269     all_partitions = []
270
271     for i, item in enumerate(list(grouper(list_gen, 2, None))):
272         pa, pb = item
273
274         if RII_ab_order == 'AB':
275             x_list, y_list, list_partition = Reinforcement_Main(pa, pb, n_RII, RefII_type,
276                                                                 a_RII, b_RII, t_RII, h_RII,
277                                                                 t_base_RII, RII_angle_var)
278
279         elif RII_ab_order == 'BA':
280             x_list, y_list, list_partition = Reinforcement_Main(pb, pa, n_RII, RefII_type,
281                                                                 a_RII, b_RII, t_RII, h_RII,
```

```
                                                                t_base_RII, RII_angle_var)

        x_gen_list += x_list
        y_gen_list += y_list
        all_partitions += list_partition

        if i < n_RI:
            xA, yA = pA
            xB, yB = pB

            alpha = np.arctan2((xB - xA),(yB - yA))

            if round(yA,2) > round(yB,2): alpha = alpha - np.pi

            if RefI_type == 'T':
                x, y, list_partition = T_Reinforcement_Points(a_RI, b_RI, t_RI, h_RI, t_base_RI,
                                                              Space_Division(pA, pB, n_RI)[i], alpha + RI_angle_var)
                x_gen_list += x
                y_gen_list += y
                all_partitions += list_partition

            elif RefI_type == 'L1':
                x, y, list_partition = L1_Reinforcement_Points(a_RI, b_RI, t_RI, h_RI, t_base_RI,
                                                               Space_Division(pA, pB, n_RI)[i], alpha + RI_angle_var)
                x_gen_list += x
                y_gen_list += y
                all_partitions += list_partition

            elif RefI_type == 'L2':
                x, y, list_partition = L2_Reinforcement_Points(a_RI, b_RI, t_RI, h_RI, t_base_RI,
                                                               Space_Division(pA, pB, n_RI)[i], alpha + RI_angle_var)
                x_gen_list += x
                y_gen_list += y
                all_partitions += list_partition

    return x_gen_list, y_gen_list, all_partitions
```

7 Apêndice - arquivo *GEOMETRY.py*

```python
from FUNCTIONS import *
import numpy as np
import math

user_created_folder = 'C:\\Users\\israe\\Desktop\\MIDSHIP SECTION MODEL'

with open(f'{user_created_folder}\\MIDSHIP SECTION - PARAMETERS.txt', 'r') as archive:
    lista = archive.readlines()
    for line in lista:
        variable_name  = line.split(' ')[0]
        variable_value = line.split(' ')[1]
        variable_class = line.split(' ')[2].replace('\n','').replace("'", '').replace("'", '')

        if variable_class == 'int':
            globals()[variable_name] = int(variable_value)

        elif variable_class == 'float':
            globals()[variable_name] = float(variable_value)

        elif variable_class == 'str':
            globals()[variable_name] = str(variable_value)

theta = np.arctan2(K1, R_bojo + K2 - w_ds)
alpha = np.arctan2(dh_deck, B_tkBBBE)

# DEFINIÃ O DOS PONTOS GERAIS DA GEOMETRIA
# Arranjos Longitudinal, Transversal e Anteparas

p1  = (R_bojo, R_bojo)
p2  = (R_bojo, 0.0)
p3  = (K2 + R_bojo, h_db)
p4  = (K2 + R_bojo, 0.0)
p5  = (K2 + R_bojo + t_longf, h_db)
p6  = (K2 + R_bojo + t_longf, 0.0)
p7  = (K2 + R_bojo + t_longf + K3, h_db)
p8  = (K2 + R_bojo + t_longf + K3, 0.0)
p9  = (K2 + R_bojo + t_longf + K3 + t_antlong + K4 - t_quilha/2, h_db)
p10 = (K2 + R_bojo + t_longf + K3 + t_antlong + K4 - t_quilha/2, 0.0)
p11 = (K2 + R_bojo + t_longf + K3 + t_antlong + K4, h_db + t_chapint)
p12 = (K2 + R_bojo + t_longf + K3 + t_antlong, h_db + t_chapint)
p13 = (K2 + R_bojo + t_longf + K3, h_db + t_chapint)

p15 = (R_bojo, - t_chapext)
p16 = (K2 + R_bojo + t_longf + K3 + t_antlong + K4, - t_chapint)
p17 = (K2 + R_bojo + t_longf + K3 + t_antlong, h_db)
p18 = (K2 + R_bojo + t_longf + K3 + t_antlong, 0.0)
p19 = (w_ds, R_bojo + K1)

p141 = (K2 + R_bojo + ((1 - np.cos(theta))/(np.sin(theta)))*t_chapint, h_db + t_chapint)
p142 = (x_from_y_Parallel_Line(h_db + t_chapint, p3, p19, t_chapint, theta), h_db + t_chapint)

p14  = ((p141[0] + p142[0])/2, (p141[1] + p142[1])/2)

p20 = (0.0 , R_bojo + K1)
p21 = (w_ds, R_bojo + K1 + t_longc)
p22 = (0.0 , R_bojo + K1 + t_longc)
p23 = (w_ds, R_bojo + K1 + t_longc + K5)
p24 = (0.0 , R_bojo + K1 + t_longc + K5)
p25 = (w_ds, R_bojo + K1 + t_longc + K5 + t_longc)
p26 = (0.0 , R_bojo + K1 + t_longc + K5 + t_longc)
p27 = (w_ds, R_bojo + K1 + t_longc + K5 + t_longc + K6)
p28 = (0.0 , R_bojo + K1 + t_longc + K5 + t_longc + K6)
p29 = (w_ds, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc)
p30 = (0.0 , R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc)
p31 = (w_ds, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7)
p32 = (0.0 , R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7)
p33 = (w_ds + t_chapint, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7)
p34 = (w_ds + t_chapint + B_tkBBBE, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck)
p35 = (w_ds + t_chapint + B_tkBBBE + t_antlong, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck)

p361 = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4 - t_RCD/2, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck)
p362 = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4 - t_RCD/2, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck - a_RCD)
p363 = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4 - t_RCD/2 - b_RCD/2, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck - a_RCD)
p364 = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4 - t_RCD/2 - b_RCD/2, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck - a_RCD - h_RCD)
p365 = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck - a_RCD - h_RCD)

p37 = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck + t_chapdeck)

if alpha != 0:
    p38 = (w_ds + t_chapint + B_tkBBBE - (t_chapdeck*((1 / np.cos(alpha)) - 1))/(np.tan(alpha)), R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck + t_cha
    p39 = (w_ds + t_chapint - (t_chapdeck*((1 / np.cos(alpha)) - 1))/(np.tan(alpha)), R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + t_chapdeck)
else:
    p38 = (w_ds + t_chapint + B_tkBBBE, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck + t_chapdeck)
    p39 = (w_ds + t_chapint, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + t_chapdeck)

p40 = (- t_chapext, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + t_chapdeck)
p41 = (- t_chapext, R_bojo)
p42 = (0.0, R_bojo)

p431 = (w_ds + t_chapint, R_bojo + K1 + ((1 - np.sin(theta))/(np.cos(theta)))*t_chapint)
p432 = (w_ds + t_chapint, y_from_x_Parallel_Line(w_ds + t_chapint, p3, p19, t_chapint, theta))

p43  = ((p431[0] + p432[0])/2, (p431[1] + p432[1])/2)

p4L    = (K2 + R_bojo, -t_chapext)
p5L    = (K2 + R_bojo + t_longf, h_db + t_chapint)
p6L    = (K2 + R_bojo + t_longf, -t_chapext)
p8L    = (K2 + R_bojo + t_longf + K3, -t_chapext)
p9L    = (K2 + R_bojo + t_longf + K3 + t_antlong + K4, h_db)
p9LU   = (K2 + R_bojo + t_longf + K3 + t_antlong + K4 - t_quilha/2, h_db + t_chapint)
p10L   = (K2 + R_bojo + t_longf + K3 + t_antlong + K4, 0.0)
p10LD  = (K2 + R_bojo + t_longf + K3 + t_antlong + K4 - t_quilha/2, -t_chapext)
p18L   = (K2 + R_bojo + t_longf + K3 + t_antlong, -t_chapext)
p20L   = (-t_chapext, R_bojo + K1)
p21L   = (w_ds + t_chapint, R_bojo + K1 + t_longc)
p22L   = (-t_chapext, R_bojo + K1 + t_longc)
p23L   = (w_ds + t_chapint, R_bojo + K1 + t_longc + K5)
p24L   = (-t_chapext, R_bojo + K1 + t_longc + K5)
p25L   = (w_ds + t_chapint, R_bojo + K1 + t_longc + K5 + t_longc)
p26L   = (-t_chapext, R_bojo + K1 + t_longc + K5 + t_longc)
p27L   = (w_ds + t_chapint, R_bojo + K1 + t_longc + K5 + t_longc + K6)
p28L   = (-t_chapext, R_bojo + K1 + t_longc + K5 + t_longc + K6)
p29L   = (w_ds + t_chapint, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc)
p30L   = (-t_chapext, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc)
p31L   = (w_ds, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + t_chapdeck)
p32L   = (- t_chapext, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7)
p32LU  = (0.0, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + t_chapdeck)
p35L   = (w_ds + t_chapint + B_tkBBBE + t_antlong, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck + t_chapdeck)
p361L  = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck)
p361LU = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4 - t_RCD/2, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck + t_chapdeck)
p362L  = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4 - t_RCD/2, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck - a_RCD - h_RCD)
p363L  = (w_ds + t_chapint + B_tkBBBE + t_antlong + K4, R_bojo + K1 + t_longc + K5 + t_longc + K6 + t_longc + K7 + dh_deck - a_RCD)

# Pontos de DefiniÃ§Ã£o das Cavernas (Gigantes)

p1C = (p14[0] - (dh_sideC - (dh_bottomC / np.sin(np.pi/2 - theta)))/np.sin(theta), p14[1] + dh_bottomC)
p2C = (p13[0] - dh_antlongC, p13[1] + dh_bottomC)
```

```
p3C = (p34[0] - dh_antlongC, p34[1] - dh_deckC/np.cos(alpha) - dh_antlongC*np.tan(alpha))
p4C = (p33[0] + dh_sideC, p33[1] - ((dh_deckC - dh_sideC*np.sin(alpha))/np.cos(alpha)))
p5C = (p43[0] + dh_sideC, p43[1] + dh_sideC*((1 - np.sin(theta))/np.cos(theta)))
p6C = (p12[0] + dh_antlongC, p12[1] + dh_bottomC)
p7C = (p11[0], p11[1] + dh_bottomC)
p8C = (p37[0], p361[1] - dh_deckC)
p9C = (p35[0] + dh_antlongC, p35[1] - dh_deckC)

# Ponto de DefiniÃ§Ã£o das Anteparas Transversais

p1A = p37
p2A = p38
p3A = p39
p4A = p40
p5A = p41
p6A = p15
p7A = p16

#-----------------------------------------------------------------------------------------------------------------------
# CONSTRUÃ Ã O GEOMÃ TRICA GERAL

# LONGITUDINAL STRUCTURE
# ReforÃ§adores do Bojo

# Bojo REGIAO 9

N8  = int(math.floor(Cartesian_Distance(p42, p20)/spacing_bojo) - 1)
N9  = int(math.floor(Cartesian_Distance( p4,  p2)/spacing_bojo) - 1)
N10 = int(math.floor(Cartesian_Distance(p19,  p3)/spacing_bojo) - 1)

x1, y1, PT1 = Double_Reinforcement_Main(p42, p20, Ref_Lside_type , N8, a_side  , b_side  , t_side  , h_side  , t_chapext, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x2, y2, PT2 = Double_Reinforcement_Main( p4,  p2, Ref_bottom_type, N9, a_bottom, b_bottom, t_bottom, h_bottom, t_chapext, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x3, y3, PT3 = Double_Reinforcement_Main( p3,  p4, 'T',  0, 0, 0, 0, 0, 0, 'T',   0, 0, 0, 0, 0, 0, 'AB')
x4, y4, PT4 = Double_Reinforcement_Main( p3, p19, 'T',  0, 0, 0, 0, 0, 0, 0, Ref_bottom_type, N10, a_bottom, b_bottom, t_bottom, h_bottom, t_chapint, np.pi/2, 'BA')

x5, y5, PT5 = Double_Reinforcement_Main(p19, p20, 'T',  0, 0, 0, 0, 0, 0, 0, Ref_cost_type, nRef_cost, aRef_cost, bRef_cost, tRef_cost, hRef_cost, t_longc, -np.pi/2, 'BA')

# ReforÃ§adores do Fundo

N1 = int(math.floor(Cartesian_Distance(p8, p6)/spacing_bottom) - 1)
N2 = int(math.floor(Cartesian_Distance(p10, p18)/spacing_bottom) - 1)

# Fundo REGIAO 1
x6 , y6, PT6 = Double_Reinforcement_Main(p8, p6, Ref_bottom_type, N1, a_bottom, b_bottom, t_bottom, h_bottom, t_chapext, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x7 , y7, PT7 = Double_Reinforcement_Main(p8, p7, Ref_bottom_type,  0, a_bottom, b_bottom, t_bottom, h_bottom, t_longf  , -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x8 , y8, PT8 = Double_Reinforcement_Main(p5, p7, Ref_bottom_type, N1, a_bottom, b_bottom, t_bottom, h_bottom, t_chapint, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x9 , y9, PT9 = Double_Reinforcement_Main(p5, p6, Ref_bottom_type,  0, a_bottom, b_bottom, t_bottom, h_bottom, t_longf  , -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')

# Fundo REGIAO 2
x10, y10, PT10 = Double_Reinforcement_Main(p10, p18, Ref_bottom_type, N2, a_bottom, b_bottom, t_bottom, h_bottom, t_chapext, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x11, y11, PT11 = Double_Reinforcement_Main(p17,  p9, Ref_bottom_type, N2, a_bottom, b_bottom, t_bottom, h_bottom, t_chapint, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x12, y12, PT12 = Double_Reinforcement_Main(p17, p18, Ref_bottom_type,  0, a_bottom, b_bottom, t_bottom, h_bottom, t_longf  , np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')

x13, y13, PT13 = Double_Reinforcement_Main(p10,  p9, 'T',  0, 0, 0, 0, 0, 0, np.pi/2, Ref_q_type, nRef_q, a_q, b_q, t_q, h_q, t_longf/2, np.pi/2, 'BA')

# ReforÃ§adores do Costado

N3 = int(math.floor(Cartesian_Distance(p24, p22)/spacing_side) - 1)
N4 = int(math.floor(Cartesian_Distance(p28, p26)/spacing_side) - 1)
N5 = int(math.floor(Cartesian_Distance(p32, p30)/spacing_side) - 1)

# Costado REGIAO 3
x14, y14, PT14 = Double_Reinforcement_Main(p22, p21, 'T', 0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x15, y15, PT15 = Double_Reinforcement_Main(p21, p23, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_Rside_type, N3, a_side, b_side, t_side, h_side, t_chapint,  np.pi/2, 'BA')
x16, y16, PT16 = Double_Reinforcement_Main(p22, p24, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_Lside_type, N3, a_side, b_side, t_side, h_side, t_chapext, -np.pi/2, 'AB')

x17, y17, PT17 = Double_Reinforcement_Main(p23, p24, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_cost_type, nRef_cost, aRef_cost, bRef_cost, tRef_cost, hRef_cost, t_longc, -np.pi/2, 'BA'

# Costado REGIAO 4
x18, y18, PT18 = Double_Reinforcement_Main(p26, p25, 'T', 0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x19, y19, PT19 = Double_Reinforcement_Main(p25, p27, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_Rside_type, N4, a_side, b_side, t_side, h_side, t_chapint,  np.pi/2, 'BA')
x20, y20, PT20 = Double_Reinforcement_Main(p26, p28, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_Lside_type, N4, a_side, b_side, t_side, h_side, t_chapext, -np.pi/2, 'AB')

x21, y21, PT21 = Double_Reinforcement_Main(p27, p28, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_cost_type, nRef_cost, aRef_cost, bRef_cost, tRef_cost, hRef_cost, t_longc, -np.pi/2, 'BA'

# Costado REGIAO 5
x22, y22, PT22 = Double_Reinforcement_Main(p30, p29, 'T', 0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x23, y23, PT23 = Double_Reinforcement_Main(p29, p31, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_Rside_type, N5, a_side, b_side, t_side, h_side, t_chapint,  np.pi/2, 'BA')
x24, y24, PT24 = Double_Reinforcement_Main(p30, p32, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_Lside_type, N5, a_side, b_side, t_side, h_side, t_chapext, -np.pi/2, 'AB')

x25, y25, PT25 = Double_Reinforcement_Main(p31, p32, 'T', 0, 0, 0, 0, 0, 0, 0, Ref_cost_type, nRef_cost, aRef_cost, bRef_cost, tRef_cost, hRef_cost, t_longc, -np.pi/2, 'BA'

# ReforÃ§adores do ConvÃ©s

N6 = int(math.floor(Cartesian_Distance(p33, p34)/spacing_side) - 1)
N7 = int(math.floor(Cartesian_Distance(p35, p361)/spacing_side) - 1)

# Tanque REGIAO 6
x26, y26, PT26 = Double_Reinforcement_Main(p14, p13, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x27, y27, PT27 = Double_Reinforcement_Main(p13, p34, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x28, y28, PT28 = Double_Reinforcement_Main(p33, p34, Ref_deck_type, N6, a_deck, b_deck, t_deck, h_deck, t_chapdeck, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x29, y29, PT29 = Double_Reinforcement_Main(p33, p43, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x30, y30, PT30 = Double_Reinforcement_Main(p43, p14, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')

# Tanque REGIAO 7
x31, y31, PT31 = Double_Reinforcement_Main(p365, p364, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x32, y32, PT32 = Double_Reinforcement_Main(p364, p363, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x33, y33, PT33 = Double_Reinforcement_Main(p363, p362, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x34, y34, PT34 = Double_Reinforcement_Main(p362, p361, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x35, y35, PT35 = Double_Reinforcement_Main( p35, p361, Ref_deck_type, N7, a_deck, b_deck, t_deck, h_deck, t_chapdeck, -np.pi/2, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x36, y36, PT36 = Double_Reinforcement_Main( p12,  p11, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')

# Antepara Longitudinal
x37, y37, PT37 = Double_Reinforcement_Main(p12,  p35,
                                           RI_al_type , nRI_al , aRI_al , bRI_al , tRI_al , hRI_al , t_antlong, -np.pi/2,
                                           RII_al_type, nRII_al, aRII_al, bRII_al, tRII_al, hRII_al, t_antlong, -np.pi/2, 'AB')

# Chapeamento Externo REGIAO 8
x38, y38, PT38 = Double_Reinforcement_Main(p37, p38, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x39, y39, PT39 = Double_Reinforcement_Main(p38, p39, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x40, y40, PT40 = Double_Reinforcement_Main(p39, p40, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x41, y41, PT41 = Double_Reinforcement_Main(p40, p41, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x42, y42, PT42 = Double_Reinforcement_Main(p15, p16, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')

# TRANVERSAL STRUCTURE
x43, y43, PT43 = Double_Reinforcement_Main(p1C, p2C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x44, y44, PT44 = Double_Reinforcement_Main(p2C, p3C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x45, y45, PT45 = Double_Reinforcement_Main(p3C, p4C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x46, y46, PT46 = Double_Reinforcement_Main(p4C, p5C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x47, y47, PT47 = Double_Reinforcement_Main(p5C, p1C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')

x48, y48, PT48 = Double_Reinforcement_Main(p8C, p9C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x49, y49, PT49 = Double_Reinforcement_Main(p9C, p6C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x50, y50, PT50 = Double_Reinforcement_Main(p6C, p7C, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')

x51, y51, PT51 = Double_Reinforcement_Main(p37, p38, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x52, y52, PT52 = Double_Reinforcement_Main(p38, p39, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
x53, y53, PT53 = Double_Reinforcement_Main(p39, p40, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 'AB')
```

```
x54, y54, PT54 = Double_Reinforcement_Main(p40, p41, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')
x55, y55, PT55 = Double_Reinforcement_Main(p15, p16, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')

# BULKHEAD STRUCTURE
x56, y56, PT56 = Double_Reinforcement_Main(p1A, p2A, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')
x57, y57, PT57 = Double_Reinforcement_Main(p2A, p3A, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')
x58, y58, PT58 = Double_Reinforcement_Main(p3A, p4A, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')
x59, y59, PT59 = Double_Reinforcement_Main(p4A, p5A, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')
x60, y60, PT60 = Double_Reinforcement_Main(p6A, p7A, 'T',  0, 0, 0, 0, 0, 0, 0, 'T', 0, 0, 0, 0, 0, 0, 0, 'AB')

# PARTICIONAMENTO DA ESTRUTURA LONGITUDINAL
all_partitions = [(p2 , p15)  , (p4  , p6)   , (p3  , p5)   , (p8  , p18)  , (p7 , p17) ,
                  (p13, p12)  , (p3  , p14)  , (p19 , p21)  , (p20 , p22)  , (p41, p42) ,
                  (p23, p25)  , (p24 , p26)  , (p27 , p29)  , (p28 , p30)  , (p9 , p9L) ,
                  (p10, p10L) , (p32 , p32L) , (p31 , p33)  , (p33 , p39)  , (p34, p38) ,
                  (p34, p35)  , (p361, p361L), (p362, p363L), (p362, p362L), (p19, p43) ,
                  (p4 , p4L)  , (p6  , p6L)  , (p5  , p5L)  , (p13 , p7)   , (p12, p17) ,
                  (p8 , p8L)  , (p18 , p18L) , (p9  , p9LU) , (p10 , p10LD), (p21, p21L),
                  (p20, p20L) , (p22 , p22L) , (p23 , p23L) , (p25 , p25L) , (p24, p24L),
                  (p26, p26L) , (p27 , p27L) , (p29 , p29L) , (p28 , p28L) , (p30, p30L),
                  (p32, p32LU), (p31 , p31L) , (p35 , p35L) , (p361, p361LU)]

for i in range(1,43,1):
    all_partitions += globals()[f'PT{i}']

all_partitions_mirror = []

for pair in all_partitions:
    pA, pB = pair
    xA, yA = pA
    xB, yB = pB

    pA_mirror = mirrorImage(1.0, 0.0, -1 * p16[0], xA, yA)
    pB_mirror = mirrorImage(1.0, 0.0, -1 * p16[0], xB, yB)

    all_partitions_mirror.append((pA_mirror, pB_mirror))

all_partitions += all_partitions_mirror

# DADOS PARA ARREDONDAMENTO DE CANTOS DAS CAVERNAS
R1_tuple = (Return_Middle(p5C,p1C), Return_Middle(p1C,p2C))
R2_tuple = (Return_Middle(p1C,p2C), Return_Middle(p2C,p3C))
R3_tuple = (Return_Middle(p2C,p3C), Return_Middle(p3C,p4C))
R4_tuple = (Return_Middle(p3C,p4C), Return_Middle(p4C,p5C))
R5_tuple = (Return_Middle(p4C,p5C), Return_Middle(p5C,p1C))

R6_tuple = (Return_Middle(p9C,p6C), Return_Middle(p6C,p7C))
R7_tuple = (Return_Middle(p8C,p9C), Return_Middle(p9C,p6C))

# Espelhados

R8_tuple  = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p5C,p1C)[0], Return_Middle(p5C,p1C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p1C,p2C)[0], Return_Middle(p1C,p2C)[1]))

R9_tuple  = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p1C,p2C)[0], Return_Middle(p1C,p2C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p2C,p3C)[0], Return_Middle(p2C,p3C)[1]))

R10_tuple = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p2C,p3C)[0], Return_Middle(p2C,p3C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p3C,p4C)[0], Return_Middle(p3C,p4C)[1]))

R11_tuple = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p3C,p4C)[0], Return_Middle(p3C,p4C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p4C,p5C)[0], Return_Middle(p4C,p5C)[1]))

R12_tuple = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p4C,p5C)[0], Return_Middle(p4C,p5C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p5C,p1C)[0], Return_Middle(p5C,p1C)[1]))

R13_tuple = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p9C,p6C)[0], Return_Middle(p9C,p6C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p6C,p7C)[0], Return_Middle(p6C,p7C)[1]))

R14_tuple = (mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p8C,p9C)[0], Return_Middle(p8C,p9C)[1]),
             mirrorImage(1.0, 0.0, -1 * p16[0], Return_Middle(p9C,p6C)[0], Return_Middle(p9C,p6C)[1]))

with open('MIDSHIP SECTION - MAIN DOTS.txt', 'w') as archive:
    for i in range(1,43,1):
        for j in range(len((globals()[f'x{i}'])) - 1):

            xA = (globals()[f'x{i}'])[j]
            yA = (globals()[f'y{i}'])[j]
            xB = (globals()[f'x{i}'])[j+1]
            yB = (globals()[f'y{i}'])[j+1]

            archive.write(f'{xA} {yA} {xB} {yB}\n')

    for i in range(1,43,1):
        for j in range(len((globals()[f'x{i}'])) - 1):

            xA, yA = mirrorImage(1.0, 0.0, -1 * p16[0], (globals()[f'x{i}'])[j], (globals()[f'y{i}'])[j])
            xB, yB = mirrorImage(1.0, 0.0, -1 * p16[0], (globals()[f'x{i}'])[j+1], (globals()[f'y{i}'])[j+1])

            archive.write(f'{xA} {yA} {xB} {yB}\n')

with open('MIDSHIP SECTION - PARTITION DOTS.txt', 'w') as archive:
    for pair in all_partitions:

        pA, pB = pair
        xA, yA = pA
        xB, yB = pB

        archive.write(f'{xA} {yA} {xB} {yB}\n')

with open('MIDSHIP SECTION - TRANSVERSAL STRUCTURE.txt', 'w') as archive:
    for i in range(43,56,1):
        for j in range(len((globals()[f'x{i}'])) - 1):
            xA = (globals()[f'x{i}'])[j]
            yA = (globals()[f'y{i}'])[j]
            xB = (globals()[f'x{i}'])[j+1]
            yB = (globals()[f'y{i}'])[j+1]

            archive.write(f'{xA} {yA} {xB} {yB}\n')

    for i in range(43,56,1):
        for j in range(len((globals()[f'x{i}'])) - 1):

            xA, yA = mirrorImage(1.0, 0.0, -1 * p16[0], (globals()[f'x{i}'])[j]  , (globals()[f'y{i}'])[j])
            xB, yB = mirrorImage(1.0, 0.0, -1 * p16[0], (globals()[f'x{i}'])[j+1], (globals()[f'y{i}'])[j+1])

            archive.write(f'{xA} {yA} {xB} {yB}\n')

with open('MIDSHIP SECTION - BULKHEAD STRUCTURE.txt', 'w') as archive:
    for i in range(56,61,1):
        for j in range(len((globals()[f'x{i}'])) - 1):
            xA = (globals()[f'x{i}'])[j]
            yA = (globals()[f'y{i}'])[j]
            xB = (globals()[f'x{i}'])[j+1]
            yB = (globals()[f'y{i}'])[j+1]

            archive.write(f'{xA} {yA} {xB} {yB}\n')

    for i in range(56,61,1):
```

```
    for i in range(56,61,1):
        for j in range(len((globals()[f'x{i}'])) - 1):

            xA, yA = mirrorImage(1.0, 0.0, -1 * p16[0], (globals()[f'x{i}'])[j]  , (globals()[f'y{i}'])[j])
            xB, yB = mirrorImage(1.0, 0.0, -1 * p16[0], (globals()[f'x{i}'])[j+1], (globals()[f'y{i}'])[j+1])

            archive.write(f'{xA} {yA} {xB} {yB}\n')

with open('MIDSHIP SECTION - TRANSVERSAL STRUCTURE RADIUS.txt', 'w') as archive:
    for i in range(1,15,1):

        pA, pB = globals()[f'R{i}_tuple']

        xA, yA = pA
        xB, yB = pB

        archive.write(f'{xA} {yA} {xB} {yB}\n')
```

8 Apêndice - arquivo *MIDSHIP SECTION FINAL.py*

```python
from abaqus import *
from abaqusConstants import *
from caeModules import *
from driverUtils import executeOnCaeStartup

from odbAccess import *

import os
import numpy as np
from datetime import datetime

string_date = datetime.today().strftime("%d-%m-%Y-%H-%M-%S")

user_created_folder = 'C:\\Users\\israe\\Desktop\\MIDSHIP SECTION MODEL'
model_folder = '{}\\MIDSHIP-SECTION-FINAL-{}'.format(user_created_folder, string_date)

os.mkdir(model_folder)
os.chdir(model_folder)

with open('{}\\MIDSHIP SECTION - PARAMETERS.txt'.format(user_created_folder), 'r') as archive:
    lista = archive.readlines()
    for line in lista:
        variable_name  = line.split(' ')[0]
        variable_value = line.split(' ')[1]
        variable_class = line.split(' ')[2].replace('\n','').replace("'", '').replace("'", '')

        if variable_class == 'int':
            globals()[variable_name] = int(variable_value)

        elif variable_class == 'float':
            globals()[variable_name] = float(variable_value)

        elif variable_class == 'str':
            globals()[variable_name] = str(variable_value)

executeOnCaeStartup()
Mdb()
mdb.models.changeKey(fromName='Model-1', toName='MIDSHIP SECTION MODEL')
backwardCompatibility.setValues(includeDeprecated=True, reportDeprecated=False)
session.journalOptions.setValues(replayGeometry=COORDINATE, recoverGeometry=COORDINATE)

# ARRANJO LONGITUDINAL

sketch = mdb.models['MIDSHIP SECTION MODEL'].ConstrainedSketch(name='SKETCH - LONGITUDINAL FRAME', sheetSize=200.0)
g, v, d, c = sketch.geometry, sketch.vertices, sketch.dimensions, sketch.constraints
sketch.setPrimaryObject(option=STANDALONE)

sketch.ArcByCenterEnds(center=(R_bojo, R_bojo)                  , point1=(R_bojo, 0.0)            , point2=(0.0, R_bojo)                  , direction=CLOCKWISE)
sketch.ArcByCenterEnds(center=(B - 2*t_chapext - R_bojo, R_bojo), point1=(B - 2*t_chapext, R_bojo), point2=(B - 2*t_chapext - R_bojo, 0.0), direction=CLOCKWISE)

sketch.ArcByCenterEnds(center=(R_bojo, R_bojo)                  , point1=(R_bojo, - t_chapext)  , point2=(-t_chapext, R_bojo)                  , direction=CLOCKWISE)
sketch.ArcByCenterEnds(center=(B - 2*t_chapext - R_bojo, R_bojo), point1=(B - t_chapext, R_bojo), point2=(B - 2*t_chapext - R_bojo, -t_chapext), direction=CLOCKWISE)

with open('{}\\MIDSHIP SECTION - MAIN DOTS.txt'.format(user_created_folder), 'r') as archive:
    lista = archive.readlines()
    for line in lista:
        xA = float(line.split(' ')[0])
        yA = float(line.split(' ')[1])
        xB = float(line.split(' ')[2])
        yB = float(line.split(' ')[3])

        pA = (xA, yA)
        pB = (xB, yB)

        sketch.Line(point1=pA, point2=pB)

part = mdb.models['MIDSHIP SECTION MODEL'].Part(name='LONGITUDINAL STRUCTURE', dimensionality=THREE_D, type=DEFORMABLE_BODY)
part = mdb.models['MIDSHIP SECTION MODEL'].parts['LONGITUDINAL STRUCTURE']
part.BaseSolidExtrude(sketch=sketch, depth=L_longframe)
sketch.unsetPrimaryObject()

# ARRANJO TRANSVERSAL

sketch = mdb.models['MIDSHIP SECTION MODEL'].ConstrainedSketch(name='SKETCH - TRANSVERSAL FRAME', sheetSize=200.0)
g, v, d, c = sketch.geometry, sketch.vertices, sketch.dimensions, sketch.constraints
sketch.setPrimaryObject(option=STANDALONE)

sketch.ArcByCenterEnds(center=(R_bojo, R_bojo)                  , point1=(R_bojo, -t_chapext)   , point2=(-t_chapext, R_bojo)                  , direction=CLOCKWISE)
sketch.ArcByCenterEnds(center=(B - 2*t_chapext - R_bojo, R_bojo), point1=(B - t_chapext, R_bojo), point2=(B - 2*t_chapext - R_bojo, -t_chapext), direction=CLOCKWISE)

with open('{}\\MIDSHIP SECTION - TRANSVERSAL STRUCTURE.txt'.format(user_created_folder), 'r') as archive:
    lista = archive.readlines()
    for line in lista:
        xA = float(line.split(' ')[0])
        yA = float(line.split(' ')[1])
        xB = float(line.split(' ')[2])
        yB = float(line.split(' ')[3])

        pA = (xA, yA)
        pB = (xB, yB)

        sketch.Line(point1=pA, point2=pB)

if R_caverna != 0:
    with open('{}\\MIDSHIP SECTION - TRANSVERSAL STRUCTURE RADIUS.txt'.format(user_created_folder), 'r') as archive:
        lista = archive.readlines()
        for line in lista:
            xA = float(line.split(' ')[0])
            yA = float(line.split(' ')[1])
            xB = float(line.split(' ')[2])
            yB = float(line.split(' ')[3])

            sketch.FilletByRadius(radius=R_caverna, curve1=g.findAt((xA, yA)), nearPoint1=((xA, yA)),
                                                    curve2=g.findAt((xB, yB)), nearPoint2=((xB, yB)))

part = mdb.models['MIDSHIP SECTION MODEL'].Part(name='TRANSVERSAL STRUCTURE', dimensionality=THREE_D, type=DEFORMABLE_BODY)
part = mdb.models['MIDSHIP SECTION MODEL'].parts['TRANSVERSAL STRUCTURE']
part.BaseSolidExtrude(sketch=sketch, depth=t_caverna)
sketch.unsetPrimaryObject()

# ANTEPARAS TRANSVERSAIS - BULKHEADS

if n_anteparas != 0:
    sketch = mdb.models['MIDSHIP SECTION MODEL'].ConstrainedSketch(name='SKETCH - BULKHEAD', sheetSize=200.0)
    g, v, d, c = sketch.geometry, sketch.vertices, sketch.dimensions, sketch.constraints
    sketch.setPrimaryObject(option=STANDALONE)

    sketch.ArcByCenterEnds(center=(R_bojo, R_bojo)                  , point1=(R_bojo, -t_chapext)   , point2=(-t_chapext, R_bojo)                  , direction=CLOCKWISE)
    sketch.ArcByCenterEnds(center=(B - 2*t_chapext - R_bojo, R_bojo), point1=(B - t_chapext, R_bojo), point2=(B - 2*t_chapext - R_bojo, -t_chapext), direction=CLOCKWISE)

    with open('{}\\MIDSHIP SECTION - BULKHEAD STRUCTURE.txt'.format(user_created_folder), 'r') as archive:
        lista = archive.readlines()
        for line in lista:
            xA = float(line.split(' ')[0])
            yA = float(line.split(' ')[1])
            xB = float(line.split(' ')[2])
            yB = float(line.split(' ')[3])
```

```
            pA = (xA, yA)
            pB = (xB, yB)

            sketch.Line(point1=pA, point2=pB)

    part = mdb.models['MIDSHIP SECTION MODEL'].Part(name='BULKHEAD STRUCTURE', dimensionality=THREE_D, type=DEFORMABLE_BODY)
    part = mdb.models['MIDSHIP SECTION MODEL'].parts['BULKHEAD STRUCTURE']
    part.BaseSolidExtrude(sketch=sketch, depth=t_bulkhead)
    sketch.unsetPrimaryObject()

# ASSEMBLY

n_longstruc = n_cavernas + 1

def Instance_Assembly(part_name, instance_name, translation_vector):
    a = mdb.models['MIDSHIP SECTION MODEL'].rootAssembly
    p = mdb.models['MIDSHIP SECTION MODEL'].parts[part_name]
    a.Instance(name=instance_name, part=p, dependent=ON)
    p1 = a.instances[instance_name]
    p1.translate(vector=translation_vector)

if n_anteparas != 0:
    for j in range(n_anteparas + 1):
        for i in range(n_cavernas):
            Instance_Assembly('TRANSVERSAL STRUCTURE', 'TRANSVERSAL STRUCTURE-{}-{}'.format(i+1, j+1), (0.0, 0.0, (i+1)*L_longframe + i*t_caverna + j*(L_station + t_bulkhea

        for i in range(n_longstruc):
            Instance_Assembly('LONGITUDINAL STRUCTURE', 'LONGITUDINAL STRUCTURE-{}-{}'.format(i+1, j+1), (0.0, 0.0, i*L_longframe + i*t_caverna + j*(L_station + t_bulkhead)

    for j in range(n_anteparas):
        Instance_Assembly('BULKHEAD STRUCTURE', 'BULKHEAD STRUCTURE-{}'.format(j+1), (0.0, 0.0, (j+1)*L_station + j*t_bulkhead))
else:
    for i in range(n_cavernas):
        Instance_Assembly('TRANSVERSAL STRUCTURE', 'TRANSVERSAL STRUCTURE-{}'.format(i+1), (0.0, 0.0, (i+1)*L_longframe + i*t_caverna))

    for i in range(n_longstruc):
        Instance_Assembly('LONGITUDINAL STRUCTURE', 'LONGITUDINAL STRUCTURE-{}'.format(i+1), (0.0, 0.0, i*L_longframe + i*t_caverna))

# INTERACTIONS

a = mdb.models['MIDSHIP SECTION MODEL'].rootAssembly

if n_anteparas != 0:
    for k in range(n_anteparas + 1):
        for i in range(n_cavernas):
            for j in range(2):
                master_region = a.Surface(sidelFaces=a.instances['TRANSVERSAL STRUCTURE-{}-{}'.format(i+1, k+1)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (i+1)*L_lor
                slave_region  = a.Surface(sidelFaces=a.instances['LONGITUDINAL STRUCTURE-{}-{}'.format(j+i+1, k+1)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (i+1)*L_

                mdb.models['MIDSHIP SECTION MODEL'].Tie(name='SURFACE CONSTRAINT-{}-{}-{}'.format(i+1,j+1,k+1), master=master_region, slave=slave_region, positionToleranceM

    for k in range(n_anteparas):
        for j in range(2):
            master_region = a.Surface(sidelFaces=a.instances['BULKHEAD STRUCTURE-{}'.format(k+1)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (k+1)*L_station + k*t_bulk
            if j == 0:
                slave_region  = a.Surface(sidelFaces=a.instances['LONGITUDINAL STRUCTURE-{}-{}'.format(j+1,k+2)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (k+1)*(L_s
            else:
                slave_region  = a.Surface(sidelFaces=a.instances['LONGITUDINAL STRUCTURE-{}-{}'.format(j+n_cavernas,k+1)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (

            mdb.models['MIDSHIP SECTION MODEL'].Tie(name='SURFACE CONSTRAINT-{}-{}'.format(k+1,j+1), master=master_region, slave=slave_region, positionToleranceMethod=COMPU

else:
    for i in range(n_cavernas):
        for j in range(2):
            master_region = a.Surface(sidelFaces=a.instances['TRANSVERSAL STRUCTURE-{}'.format(i+1)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (i+1)*L_longframe + i*
            slave_region  = a.Surface(sidelFaces=a.instances['LONGITUDINAL STRUCTURE-{}'.format(j+i+1)].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (i+1)*L_longframe +

            mdb.models['MIDSHIP SECTION MODEL'].Tie(name='SURFACE CONSTRAINT-{}-{}'.format(i+1,j+1), master=master_region, slave=slave_region, positionToleranceMethod=COMPU

# MPC CONSTRAINTS

a.ReferencePoint(point=((B/2) - t_chapext, D/2, 0.0))
a.ReferencePoint(point=((B/2) - t_chapext, D/2, (n_anteparas + 1)*L_station + n_anteparas*t_bulkhead))

if n_anteparas != 0:
    list_longstr  = np.arange(1, n_longstruc + 1, 1)
    list_bulkhead = np.arange(1, n_anteparas + 2, 1)

    region1 = regionToolset.Region(referencePoints=(a.referencePoints[a.features['RP-1'].id], ))
    region2 = regionToolset.Region(faces=a.instances['LONGITUDINAL STRUCTURE-{}-{}'.format(min(list_longstr),min(list_bulkhead))].faces.findAt((((B/2) - t_chapext, -t_chape

    region3 = regionToolset.Region(referencePoints=(a.referencePoints[a.features['RP-2'].id], ))
    region4 = regionToolset.Region(faces=a.instances['LONGITUDINAL STRUCTURE-{}-{}'.format(max(list_longstr),max(list_bulkhead))].faces.findAt((((B/2) - t_chapext, -t_chape

else:
    list_longstr = np.arange(1, n_longstruc + 1, 1)

    region1 = regionToolset.Region(referencePoints=(a.referencePoints[a.features['RP-1'].id], ))
    region2 = regionToolset.Region(faces=a.instances['LONGITUDINAL STRUCTURE-{}'.format(min(list_longstr))].faces.findAt((((B/2) - t_chapext, -t_chapext/2, 0.0), )))

    region3 = regionToolset.Region(referencePoints=(a.referencePoints[a.features['RP-2'].id], ))
    region4 = regionToolset.Region(faces=a.instances['LONGITUDINAL STRUCTURE-{}'.format(max(list_longstr))].faces.findAt((((B/2) - t_chapext, -t_chapext/2, (n_anteparas + 1

mdb.models['MIDSHIP SECTION MODEL'].MultipointConstraint(name='MPC CONSTRAINT 1', controlPoint=region1, surface=region2, mpcType=BEAM_MPC, userMode=DOF_MODE_MPC, userType=0
mdb.models['MIDSHIP SECTION MODEL'].MultipointConstraint(name='MPC CONSTRAINT 2', controlPoint=region3, surface=region4, mpcType=BEAM_MPC, userMode=DOF_MODE_MPC, userType=0

# MATERIAL E SECAO

mdb.models['MIDSHIP SECTION MODEL'].Material(name=Material_Name)
mdb.models['MIDSHIP SECTION MODEL'].materials[Material_Name].Elastic(table=((E_Young, v_poisson), ))
mdb.models['MIDSHIP SECTION MODEL'].HomogeneousSolidSection(name='{} SECTION'.format(Material_Name), material=Material_Name, thickness=None)
mdb.models['MIDSHIP SECTION MODEL'].materials[Material_Name].Density(table=((density, ), ))

p = mdb.models['MIDSHIP SECTION MODEL'].parts['LONGITUDINAL STRUCTURE']
region = regionToolset.Region(cells=p.cells.findAt((((B/2) - t_chapext, -t_chapext/2, L_longframe/4), )))
p.SectionAssignment(region=region, sectionName='{} SECTION'.format(Material_Name), offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTION)

p = mdb.models['MIDSHIP SECTION MODEL'].parts['TRANSVERSAL STRUCTURE']
region = regionToolset.Region(cells=p.cells.findAt((((B/2) - t_chapext, -t_chapext/2, t_caverna/4), )))
p.SectionAssignment(region=region, sectionName='{} SECTION'.format(Material_Name), offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTION)

if n_anteparas != 0:
    p = mdb.models['MIDSHIP SECTION MODEL'].parts['BULKHEAD STRUCTURE']
    region = regionToolset.Region(cells=p.cells.findAt((((B/2) - t_chapext, -t_chapext/2, t_bulkhead/4), )))
    p.SectionAssignment(region=region, sectionName='{} SECTION'.format(Material_Name), offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTI

# STEP

mdb.models['MIDSHIP SECTION MODEL'].StaticStep(name='Step-1', previous='Initial', maxNumInc=100000, initialInc=0.1, minInc=1e-50, maxInc=0.1)

# CONDICOES DE CONTORNO

a.Set(referencePoints=(a.referencePoints[a.features['RP-1'].id], ), name='RP-1') # Criacao do Set RP-1
a.Set(referencePoints=(a.referencePoints[a.features['RP-2'].id], ), name='RP-2') # Criacao do Set RP-2

region5 = regionToolset.Region(referencePoints=(a.referencePoints[a.features['RP-1'].id], a.referencePoints[a.features['RP-2'].id], ))
```

```
mdb.models['MIDSHIP SECTION MODEL'].DisplacementBC(name='BC-1', createStepName='Initial', region=region5, u1=SET, u2=SET, u3=SET, ur1=UNSET, ur2=UNSET, ur3=SET, amplitude=U

Coef_LCtk_LBP = 0.755747023028532
LBP = L / Coef_LCtk_LBP
Cb = 0.85
kl = 110

if (LBP >= 61) and (LBP < 90):

    Cl = 0.044*LBP + 3.75
    MWS = -kl * Cl * (LBP**2) * B * (Cb + 0.7)

    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-1', createStepName='Step-1', region=a.sets['RP-1'], cm1= abs(MWS), distributionType=UNIFORM, field='', localCsys=N
    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-2', createStepName='Step-1', region=a.sets['RP-2'], cm1=-abs(MWS), distributionType=UNIFORM, field='', localCsys=N

elif (LBP >= 90) and (LBP < 300):

    Cl = 10.75 - ((300 - LBP)**1.5)/100
    MWS = -kl * Cl * (LBP**2) * B * (Cb + 0.7)

    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-1', createStepName='Step-1', region=a.sets['RP-1'], cm1= abs(MWS), distributionType=UNIFORM, field='', localCsys=N
    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-2', createStepName='Step-1', region=a.sets['RP-2'], cm1=-abs(MWS), distributionType=UNIFORM, field='', localCsys=N

elif (LBP >= 300) and (LBP < 350):

    Cl = 10.75
    MWS = -kl * Cl * (LBP**2) * B * (Cb + 0.7)

    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-1', createStepName='Step-1', region=a.sets['RP-1'], cm1= abs(MWS), distributionType=UNIFORM, field='', localCsys=N
    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-2', createStepName='Step-1', region=a.sets['RP-2'], cm1=-abs(MWS), distributionType=UNIFORM, field='', localCsys=N

elif (LBP >= 350) and (LBP < 500):

    Cl = 10.75 - ((300 - LBP)**1.5)/150
    MWS = -kl * Cl * (LBP**2) * B * (Cb + 0.7)

    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-1', createStepName='Step-1', region=a.sets['RP-1'], cm1= abs(MWS), distributionType=UNIFORM, field='', localCsys=N
    mdb.models['MIDSHIP SECTION MODEL'].Moment(name='Load-2', createStepName='Step-1', region=a.sets['RP-2'], cm1=-abs(MWS), distributionType=UNIFORM, field='', localCsys=N

else:
    mdb.models['MIDSHIP SECTION MODEL'].DisplacementBC(name='BC-2', createStepName='Step-1' , region=region1, u1=UNSET, u2=UNSET, u3=UNSET, ur1=   applied_rotation, ur2=UNS
    mdb.models['MIDSHIP SECTION MODEL'].DisplacementBC(name='BC-3', createStepName='Step-1' , region=region3, u1=UNSET, u2=UNSET, u3=UNSET, ur1= - applied_rotation, ur2=UNS

# HISTORY OUTPUTS

mdb.models['MIDSHIP SECTION MODEL'].historyOutputRequests['H-Output-1'].setValues(variables=('IRX1', 'IRX2', 'IRX3', 'IRRI11', 'IRRI22', 'IRRI33', 'IRRI12', 'IRRI13', 'IRRI
mdb.models['MIDSHIP SECTION MODEL'].HistoryOutputRequest(name='H-Output-2', createStepName='Step-1', variables=('RF1', 'RF2', 'RF3', 'RM1', 'RM2', 'RM3', 'RT', 'RM', 'RWM')
mdb.models['MIDSHIP SECTION MODEL'].HistoryOutputRequest(name='H-Output-3', createStepName='Step-1', variables=('RF1', 'RF2', 'RF3', 'RM1', 'RM2', 'RM3', 'RT', 'RM', 'RWM')

# PARTICIONAMENTO

p = mdb.models['MIDSHIP SECTION MODEL'].parts['LONGITUDINAL STRUCTURE']
f, e, d = p.faces, p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates=(B - 2*t_chapext - w_ds, D/2, L_longframe)),
                          sketchUpEdge=e.findAt(coordinates=(B - t_chapext, D/2, L_longframe)),
                          sketchPlaneSide=SIDE1, origin=(0.0, 0.0, L_longframe))

sketch = mdb.models['MIDSHIP SECTION MODEL'].ConstrainedSketch(name='SKETCH - PARTICIONAMENTO', sheetSize=130.42, gridSpacing=3.26, transform=t)
g, v, d, c = sketch.geometry, sketch.vertices, sketch.dimensions, sketch.constraints
sketch.setPrimaryObject(option=SUPERIMPOSE)

def Return_Middle(pA, pB):

    xA, yA = pA
    xB, yB = pB

    return ((xA + xB)/2, (yA + yB)/2)

with open('{}\\MIDSHIP SECTION - PARTITION DOTS.txt'.format(user_created_folder), 'r') as archive:
    lista = archive.readlines()
    for line in lista:
        xA = float(line.split(' ')[0])
        yA = float(line.split(' ')[1])
        xB = float(line.split(' ')[2])
        yB = float(line.split(' ')[3])

        sketch.Line(point1=(xA, yA), point2=(xB, yB))

p.PartitionFaceBySketch(sketchUpEdge=e.findAt(coordinates=(B - t_chapext, D/2, L_longframe)), faces=f.findAt(((B - 2*t_chapext - w_ds, D/2, L_longframe), )), sketch=sketch)
sketch.unsetPrimaryObject()

pickedEdges_list = []

with open('{}\\MIDSHIP SECTION - PARTITION DOTS.txt'.format(user_created_folder), 'r') as archive:
    lista = archive.readlines()
    for line in lista:
        xA = float(line.split(' ')[0])
        yA = float(line.split(' ')[1])
        xB = float(line.split(' ')[2])
        yB = float(line.split(' ')[3])
        pM = Return_Middle((xA, yA), (xB, yB))
        xM, yM = pM
        pickedEdges_list.append(e.findAt(coordinates=(xM, yM, L_longframe)))

pickedEdges = tuple(pickedEdges_list)
pickedCells = p.cells.findAt(((B/2, -t_chapext/2, L_longframe), ))
p.PartitionCellByExtrudeEdge(line=e.findAt(coordinates=(B - t_chapext, D - dh_deck, L_longframe/4)), cells=pickedCells, edges=pickedEdges, sense=REVERSE)

# MALHA

p = mdb.models['MIDSHIP SECTION MODEL'].parts['TRANSVERSAL STRUCTURE']
p.seedPart(size=Average_element_size, deviationFactor=0.1, minSizeFactor=0.1)
p.generateMesh()

p = mdb.models['MIDSHIP SECTION MODEL'].parts['LONGITUDINAL STRUCTURE']
p.seedPart(size=Average_element_size, deviationFactor=0.1, minSizeFactor=0.1)
p.generateMesh()

if n_anteparas != 0:
    p = mdb.models['MIDSHIP SECTION MODEL'].parts['BULKHEAD STRUCTURE']
    p.seedPart(size=Average_element_size, deviationFactor=0.1, minSizeFactor=0.1)
    p.generateMesh()

# JOB

mdb.Job(name='MIDSHIP-SECTION-FINAL-{}'.format(string_date), model='MIDSHIP SECTION MODEL',
    description='', type=ANALYSIS, atTime=None, waitMinutes=0, waitHours=0,
    queue=None, memory=90, memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
    modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',
    scratch='', resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=1,
    numGPUs=0)

# SUBMISSAO DO JOB

mdb.jobs['MIDSHIP-SECTION-FINAL-{}'.format(string_date)].submit(consistencyChecking=OFF)
mdb.jobs['MIDSHIP-SECTION-FINAL-{}'.format(string_date)].waitForCompletion()

# EXPORTACAO DOS PARAMETROS RELATIVOS AO MODELO EXTRUDADO - NA PASTA DE SALVAMENTO DO RESPECTIVO MODELO
```

```
# EXPORTACAO DOS PARAMETROS RELATIVOS AO MODELO EXECUTADO - NA PASTA DE SALVAMENTO DO RESPECTIVO MODELO
with open('{}\\MIDSHIP-SECTION-{} - PARAMETERS.txt'.format(model_folder, string_date), 'w') as archive:
    with open('{}\\MIDSHIP SECTION - PARAMETERS.txt'.format(user_created_folder), 'r') as original_archive:
        lista = original_archive.readlines()
    for line in lista:
        archive.write(line)

def MIDSHIP_SECTION_OUTPUT(simulation_folder):

    import os
    import glob

    os.chdir(simulation_folder)

    List_odb_files = []

    for archive in glob.glob("*.odb"):
        Name = archive
        List_odb_files.append(Name)

    Odb_file = List_odb_files[0]

    output_folder = '{}\\{} - OUTPUT DATA'.format(simulation_folder, Odb_file.replace('.odb',''))

    Midship = session.openOdb(name=Odb_file)

    os.mkdir(output_folder)
    os.chdir(output_folder)

    Mass = Midship.steps['Step-1'].mass
    MassCenter = Midship.steps['Step-1'].massCenter

    ICenter = Midship.steps['Step-1'].inertiaAboutCenter
    IOrigin = Midship.steps['Step-1'].inertiaAboutOrigin

    Frame_list = []
    RF3_list = []
    RM3_list = []
    list_max_bulk_tensions = []

    for i in range(len(Midship.steps['Step-1'].frames)):
        frame = Midship.steps['Step-1'].historyRegions['Node ASSEMBLY.1'].historyOutputs['RF3'].data[i][0]
        Frame_list.append(frame)
        RF3_list.append(abs(Midship.steps['Step-1'].historyRegions['Node ASSEMBLY.1'].historyOutputs['RF3'].data[i][1]))
        RM3_list.append(abs(Midship.steps['Step-1'].historyRegions['Node ASSEMBLY.1'].historyOutputs['RM3'].data[i][1]))

    list_longstr = []
    for item in Midship.rootAssembly.instances.keys():
        if 'LONGITUDINAL STRUCTURE' in item:
            list_longstr.append(item)

    middle_longstr = [list_longstr[:int(len(list_longstr)/2)][-1], list_longstr[int(len(list_longstr)/2):][0]]

    strees_list = []
    S_Field = Midship.steps['Step-1'].frames[-1].fieldOutputs['S']

    for j in range(len(S_Field.bulkDataBlocks)):
        Instance = S_Field.bulkDataBlocks[j].instance
        InstanceName = Instance.name
        if InstanceName in middle_longstr:
            for k in range(len(Instance.elements)):
                strees_list.append(S_Field.bulkDataBlocks[j].mises[k])
        else:
            pass

    Max_Stress_Midship = max(strees_list)

    with open('{}\\{} - OUTPUT DATA.txt'.format(output_folder, Odb_file.replace('.odb','')), 'w') as archive:
        archive.write('Mass: {}\n'.format(Mass))
        archive.write('Inertia at Center of Mass: {}\n'.format(ICenter))
        archive.write('Inertia at Origin (Ixx, Iyy, Izz, Ixy, Iyz, Izx): {}\n'.format(IOrigin))
        archive.write('RF3 (RP-1 - absolute value): {}\n'.format(max(RF3_list)))
        archive.write('RM3 (RP-1 - absolute value): {}\n'.format(max(RM3_list)))
        archive.write('Mises Stress (max at amidship): {}\n'.format(Max_Stress_Midship))

    session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=344, height=193)
    session.viewports['Viewport: 1'].makeCurrent()
    session.viewports['Viewport: 1'].maximize()
    session.viewports['Viewport: 1'].setValues(displayedObject=Midship)

    session.viewports['Viewport: 1'].view.setProjection(projection=PERSPECTIVE)
    session.viewports['Viewport: 1'].odbDisplay.basicOptions.setValues(curveRefinementLevel=EXTRA_FINE)
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(triadFont='-*-futura lt bt-bold-r-normal-*-*-120-*-*-p-*-*-*')
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(legendFont='-*-futura lt bt-medium-r-normal-*-*-90-*-*-p-*-*-*')
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(titleFont='-*-futura lt bt-medium-r-normal-*-*-80-*-*-p-*-*-*')
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(stateFont='-*-futura lt bt-medium-r-normal-*-*-80-*-*-p-*-*-*')
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(legendBox=OFF, legendMinMax=ON, legendPosition=(1, 99), titlePosition=(13, 15), statePosition=(13,
    session.viewports['Viewport: 1'].odbDisplay.contourOptions.setValues(contourMethod=TESSELLATED)
    session.viewports['Viewport: 1'].odbDisplay.commonOptions.setValues(deformationScaling=UNIFORM, uniformScaleFactor=1, edgeColorWireHide='#FFFFFF', fillColor='#FFFFFF')
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(UNDEFORMED, ))
    session.viewports['Viewport: 1'].odbDisplay.commonOptions.setValues(visibleEdges=EXTERIOR)
    session.pngOptions.setValues(imageSize=(4096, 2340))
    session.printOptions.setValues(compass=ON)

    views_list = ['Front', 'Right', 'Left', 'Iso']
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(title=OFF, state=OFF)

    for i in range(len(views_list)):
        session.viewports['Viewport: 1'].view.setValues(session.views[views_list[i]])
        session.printToFile(fileName='{}\\{} - {} - {}'.format(output_folder, i+1, Odb_file.replace('.odb',''), views_list[i].replace('.','').upper()), format=PNG, canvasOb

    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(DEFORMED, ))
    session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])
    session.viewports['Viewport: 1'].viewportAnnotationOptions.setValues(title=ON, state=ON)
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(CONTOURS_ON_DEF, ))
    session.viewports['Viewport: 1'].odbDisplay.commonOptions.setValues(visibleEdges=FREE)
    session.viewports['Viewport: 1'].odbDisplay.commonOptions.setValues(deformationScaling=AUTO)

    S_list = [(INVARIANT, 'Mises'),
              (INVARIANT, 'Max. Principal'),
              (INVARIANT, 'Max. Principal (Abs)'),
              (INVARIANT, 'Mid. Principal'),
              (INVARIANT, 'Min. Principal'),
              (INVARIANT, 'Tresca'),
              (INVARIANT, 'Pressure'),
              (INVARIANT, 'Third Invariant'),
              (COMPONENT, 'S11'),
              (COMPONENT, 'S22'),
              (COMPONENT, 'S33'),
              (COMPONENT, 'S12'),
              (COMPONENT, 'S13'),
              (COMPONENT, 'S23')]

    for i in range(len(S_list)):
        session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(variableLabel='S', outputPosition=INTEGRATION_POINT, refinement=S_list[i], )
        session.printToFile(fileName= '{}\\{} -  STREES - {}'.format(output_folder, Odb_file.replace('.odb',''), S_list[i][1].replace('.','').upper()), format=PNG, canvasOb

    U_list = [(INVARIANT, 'Magnitude'),
```

```
            (COMPONENT, 'U1'),
            (COMPONENT, 'U2'),
            (COMPONENT, 'U3')]

    for i in range(len(U_list)):
        session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(variableLabel='U', outputPosition=NODAL, refinement=U_list[i], )
        session.printToFile(fileName= '{}\\{} - DEFORMATION U - {}'.format(output_folder, Odb_file.replace('.odb',''), U_list[i][1].replace('.','').upper()), format=PNG, ca

    UR_list = [(INVARIANT, 'Magnitude'),
               (COMPONENT, 'UR1'),
               (COMPONENT, 'UR2'),
               (COMPONENT, 'UR3')]

    for i in range(len(UR_list)):
        session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(variableLabel='UR', outputPosition=NODAL, refinement=UR_list[i], )
        session.printToFile(fileName='{}\\{} - DEFORMATION UR - {}'.format(output_folder, Odb_file.replace('.odb',''), UR_list[i][1].replace('.','').upper()), format=PNG, c

    Midship.close()

MIDSHIP_SECTION_OUTPUT(model_folder)
```

9 Apêndice - arquivo *MAIN.py*

```
exec(open(r'PARAMETERS.py').read())
from PARAMETERS import Return_Parameters_File
from GEOMETRY import *
import time
spacing_list = [0.75, 0.875, 1.00, 1.125, 1.25, 1.375, 1.50, 1.625, 1.75, 1.875, 2.00, 2.125, 2.25, 2.375, 2.50, 2.625, 2.75]

for spacing in spacing_list:
    start = time.perf_counter()
    parameters = {'n_anteparas'         :  3    ,
                  'n_cavernas'          :  9    ,
                  'L_longframe'         :  5.0  ,
                  'B_tkBBBE'            : 15.0  ,
                  'R_bojo'              :  3.0  ,
                  'h_db'                :  2.5  ,
                  'w_ds'                :  2.0  ,
                  'dh_deck'             :  1.0  ,
                  't_longc'             :  0.0254,
                  't_longf'             :  0.0254,
                  't_quilha'            :  0.0254,
                  't_antlong'           :  0.0254,
                  't_caverna'           :  0.0254,
                  't_bulkhead'          :  0.0254,
                  't_chapint'           :  0.0254,
                  't_chapext'           :  0.0254,
                  't_chapdeck'          :  0.0254,
                  'spacing_bottom'      :  spacing ,
                  'spacing_side'        :  spacing ,
                  'spacing_deck'        :  spacing ,
                  'spacing_bojo'        :  spacing ,
                  'K3'                  :  9.0  ,
                  'K4'                  : 10.0  ,
                  'K5'                  :  8.0  ,
                  'K6'                  :  8.0  ,
                  'K7'                  :  8.0  ,
                  'a_RCD'               :  0.35 ,
                  'b_RCD'               :  0.25 ,
                  't_RCD'               :  0.0254,
                  'h_RCD'               :  0.0254,
                  'nRef_cost'           :  2    ,
                  'Ref_cost_type'       : 'T'   ,
                  'aRef_cost'           :  0.25 ,
                  'bRef_cost'           :  0.15 ,
                  'tRef_cost'           :  0.0254,
                  'hRef_cost'           :  0.0254,
                  'Ref_bottom_type'     : 'T'   ,
                  'a_bottom'            :  0.25 ,
                  'b_bottom'            :  0.15 ,
                  't_bottom'            :  0.0127,
                  'h_bottom'            :  0.0127,
                  'nRef_q'              :  2    ,
                  'Ref_q_type'          :  'L1' ,
                  'a_q'                 :  0.25 ,
                  'b_q'                 :  0.15 ,
                  't_q'                 :  0.0127,
                  'h_q'                 :  0.0127,
                  'Ref_Lside_type'      :  'L2' ,
                  'Ref_Rside_type'      :  'L1' ,
                  'a_side'              :  0.25 ,
                  'b_side'              :  0.15 ,
                  't_side'              :  0.0127,
                  'h_side'              :  0.0127,
                  'Ref_deck_type'       :  'T'  ,
                  'a_deck'              :  0.25 ,
                  'b_deck'              :  0.15 ,
                  't_deck'              :  0.0127,
                  'h_deck'              :  0.0127,
                  'nRI_al'              :  0    ,
                  'RI_al_type'          :  'T'  ,
                  'aRI_al'              :  0.25 ,
                  'bRI_al'              :  0.15 ,
                  'tRI_al'              :  0.0127,
                  'hRI_al'              :  0.0127,
                  'nRII_al'             :  int(math.floor(Cartesian_Distance(p12, p35)/spacing) - 1),
                  'RII_al_type'         :  'L2' ,
                  'aRII_al'             :  0.25 ,
                  'bRII_al'             :  0.15 ,
                  'tRII_al'             :  0.0127,
                  'hRII_al'             :  0.0127,
                  'dh_deckC'            :  2.0  ,
                  'dh_sideC'            :  1.5  ,
                  'dh_bottomC'          :  1.5  ,
                  'dh_antlongC'         :  1.5  ,
                  'R_caverna'           :  3.0  ,
                  'Material_Name'       :  'STEEL',
                  'E_Young'             :  210e9  ,
                  'v_poisson'           :  0.33   ,
                  'density'             :  7850.0 ,
                  'applied_rotation'    :  0.045  ,
                  'Average_element_size':  1.50   }
    Return_Parameters_File(parameters)
    exec(open(r'GEOMETRY.py').read())
    import os
    os.system('cmd /c "abaqus cae noGUI="{}""'.format(r'C:\Users\israe\Desktop\MIDSHIP SECTION MODEL\MIDSHIP SECTION FINAL.py'))
    end = time.perf_counter()
    print(f'Tempo de ExecuÃ§Ã£o: {end - start:.4f}')
```