



**UNIVERSIDADE SENAI CIMATEC**  
**GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

**OTIMIZAÇÃO DE THRUSTERS PARA TOLERÂNCIA A FALHAS EM  
VEÍCULOS REMOTAMENTE OPERADOS (ROVS) EM 4 DOFS**

Júlia Maria Nascimento Ribeiro<sup>1</sup>

Orientadora: Msc. Rebeca Tourinho Lima<sup>2</sup>

Coorientador: Msc. Lucas Marins Batista<sup>3</sup>

<sup>1</sup>Universidade Senai Cimatec, E-mail: julia.ribeiro@aln.senaicimatec.edu.br;

<sup>2</sup>Universidade Senai Cimatec, E-mail: rebeca.lima@doc.senaicimatec.edu.br;

<sup>3</sup>Universidade Senai Cimatec, E-mail: lucas.batista@fieb.org.br;

**THRUSTER OPTIMIZATION FOR FAULT TOLERANCE IN REMOTELY  
OPERATED VEHICLES (ROVS) IN 4 DOFS**

**Resumo:** Este trabalho aborda o desenvolvimento e validação de um algoritmo de otimização para tolerância à falha em veículos subaquáticos remotamente operados (ROVs), especificamente no BlueROV2 Standard que conta com 6 thrusters. O objetivo principal é manter a navegabilidade do veículo em casos de falha total de um ou mais thrusters, através da redistribuição otimizada das forças entre os thrusters remanescentes. A metodologia envolveu simulações computacionais utilizando o Gazebo Ignition e ROS 2 Control, testando diferentes cenários de falhas em trajetórias de movimento (surge, sway, heave e yaw). Os resultados demonstram que o algoritmo é eficaz na manutenção da navegabilidade do ROV, especialmente quando as combinações mínimas de thrusters são respeitadas, melhorando significativamente a velocidade média e distância percorrida mesmo em condições de falha. O trabalho contribui para a melhoria no processo de operação de ROVs e em sua capacidade de conclusão de tarefas.

**Palavras-Chaves:** ROV; otimização; tolerância a falhas; thrusters; veículos subaquáticos

**Abstract:** This work addresses the development and validation of an optimization algorithm for fault tolerance in remotely operated underwater vehicles (ROVs), specifically in the BlueROV2 Standard that has 6 thrusters. The main objective is to maintain the vehicle's navigability in cases of total failure of one or more thrusters, through the optimized redistribution of forces among the remaining thrusters. The methodology involved computational simulations using Gazebo Ignition and ROS 2 Control, testing different failure scenarios in movement trajectories (surge, sway, heave, and yaw). The results demonstrate that the algorithm is effective in maintaining the ROV's navigability, especially when the minimum thruster combinations are respected, significantly improving the average speed and distance traveled even under failure conditions. This work contributes to improving the ROV operation process and its ability to complete tasks.

**Keywords:** ROV; optimization; fault tolerance; thrusters; underwater vehicles

## 1 INTRODUÇÃO

Segundo Biazon (2017), os oceanos representam cerca de 70% da superfície terrestre na atualidade, mas apesar da sua vasta ocupação, somente 5% do território oceânico é conhecido pelo ser humano. A lacuna de conhecimento gera uma crescente demanda na indústria por recursos encontrados *offshore*, como petróleo, recursos minerais (óleos, gás e metais) e biodiversidades.

A indústria apresenta uma crescente expansão em relação ao oceano. No relatório anual de sustentabilidade 2024 da Petrobras (PETROBRAS, 2024) é relatado que 98% da produção do ano de 2024 veio de águas profundas ou ultraprofundas. Esse cenário de exploração crescente cria uma demanda cada vez maior por tecnologias e mão de obra especializada.

Mergulhadores profissionais são uma mão de obra especializada amplamente utilizada na indústria para trabalhos *offshore*, seja para manutenção ou instalação de equipamentos. Segundo (OSHA, 2023), a organização de trabalho americana, os perigos enfrentados por profissionais do mergulho incluem afogamento, riscos respiratórios e cardiovasculares, hipotermia e riscos físicos devido ao manuseio de equipamentos pesados embaixo d'água, além dos riscos relacionados à descompressão e possíveis acidentes causados pelo ambiente de alto risco e nível elevado de estresse.

Dessa maneira, a indústria está constantemente em busca de melhorar a segurança no trabalho. Dentro desse contexto surgem os ROVs (*Remotely Operated Vehicles*), sendo esses uma alternativa estratégica para retirar o trabalhador de uma zona iminente de perigo.

ROVs são veículos subaquáticos que variam de tamanho, podendo ser de maior ou menor porte, e possuem a capacidade de se autopropelir abaixo da superfície da água, podendo, em alguns casos, também operar na superfície da água, dependendo de seu projeto e aplicação. Por serem veículos remotamente operados, podem ser utilizados em aplicações em alta profundidade de forma mais segura por não precisarem da presença de um ser humano na área de perigo.

Ao tirar o ser humano da zona de perigo são reduzidos os riscos fatais, além de oferecer vantagens operacionais como a possibilidade de um trabalho submerso contínuo e prolongado.

Apesar de a utilização de ROVs ser vantajosa e funcional, a operação desses veículos é um desafio devido às perturbações geradas pelos oceanos, que contam com correntes marítimas imprevisíveis, baixa visibilidade, entre outras adversidades.

Assim, para lidar com as adversidades enfrentadas no ambiente subaquático, os ROVs necessitam de sistemas de controle capazes de compensar os ruídos gerados pelas perturbações

ambientais, garantindo estabilidade, navegação autônoma e execução adequada das atividades para as quais foram projetados. Dessa forma, diversas técnicas de controle foram desenvolvidas, variando desde a modelagem do sistema até a utilização de inteligência artificial.

Em (XU; WANG, 2018) são citados alguns tipos de controladores, entre eles, o controlador PID, caracterizado por ser um controlador simples e amplamente utilizado, é um controle de resposta de velocidade, com efeito linear. Apesar de ser amplamente utilizado, não é o ideal para processos complexos e não lineares ou variantes no tempo.

As redes neurais também são apontadas como um possível controlador para um ROV, sendo essas caracterizadas por serem capazes de aproximar qualquer função não linear na teoria. Apresentam uma alta capacidade de processamento paralelo, o que gera uma grande tolerância a falhas e eficiência no tratamento de múltiplos sinais de entrada e saída.

Apesar das inúmeras vantagens, é necessário ter cautela, pois há a tendência de convergir para mínimos locais, além de sofrer com a lentidão na aprendizagem, limitação no número de camadas e riscos de *overfitting*. A limitação de camadas acontece quando a rede neural tem poucas camadas, o que impede que a mesma aprenda padrões complexos. Já o *overfitting* ocorre quando uma rede neural é treinada de forma excessiva ou por tempo prolongado, causando que a mesma decore os dados do teste, indo muito bem nos testes, mas apresentando resultados ruins com novos dados.

Por último é mencionado, em Xu e Wang (2018), o controlador *fuzzy*, caracterizado por sua independência de uma modelagem matemática exata do objeto de controle, tendo sua essência em estratégias de autocontrole.

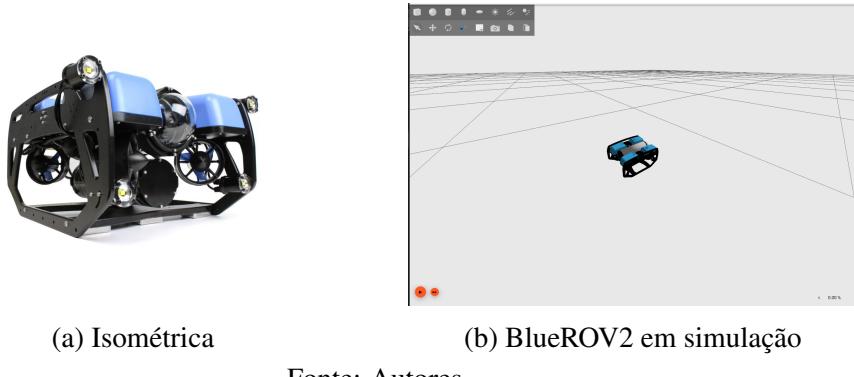
Segundo Christ e Wernli Sr (2013), o sistema de propulsão impacta diretamente no design de um ROV, impactando assim nos tipos de *thrusters* escolhidos e na configuração dos mesmos. *Thrusters* são dispositivos responsáveis por impulsionar a embarcação, ou seja, são os propulsores que garantem sua movimentação.

Um dos desafios enfrentados por veículos que utilizam *thrusters* é a possibilidade de falha em um ou mais desses dispositivos durante a operação, o que levanta a questão de como recuperar o controle do veículo ou garantir que ele continue navegando mesmo diante dessas falhas.

Dessa forma, o objetivo principal deste trabalho é abordar o problema da falha de *thrusters* por meio da aplicação de um algoritmo de otimização para gerar tolerância a falhas, através da otimização da redistribuição das forças restantes pós-falha, buscando manter a estabilidade e a funcionalidade do sistema mesmo em condições adversas.

Os estudos relacionados ao algoritmo de otimização serão aplicados ao robô BlueROV2, um veículo subaquático do tipo mini ROV desenvolvido pela empresa Blue Robotics. A Figura 1 apresenta uma representação visual do robô real e na simulação.

Figura 1: ROV BlueROV2.



Fonte: Autores.

A modelagem do algoritmo otimizador será desenvolvida levando em consideração as dimensões, características e requisitos específicos do BlueROV2.

Todos os testes e simulações serão realizados em ambiente computacional, utilizando ferramentas como o Gazebo Sim, para simular a movimentação do ROV pré e pós-falha, para a observabilidade da realocação das forças otimizadas.

Nas seções seguintes, são apresentados em detalhe os fundamentos teóricos que sustentam o estudo, a metodologia empregada para o desenvolvimento da pesquisa, os principais resultados alcançados e, por fim, as conclusões obtidas a partir das análises realizadas.

## 2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica fundamental para modelagem de ROV é descrita em *Handbook of marine craft hydrodynamics and motion control* (FOSSEN, 2021), onde é demonstrado o modelo matemático para uma embarcação marítima com 6 DOFs. As equações de movimento adotadas do modelo dinâmico apresentado por Fossen contam com a equação de cinemática, Equação 1, e a equação de cinética, Equação 2.

$$\dot{\eta} = J(\eta)v \quad (1)$$

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \quad (2)$$

Onde:

- $J(\eta)$  – É a matriz de transformação entre Body e NED;
- $M$  – É a matriz de massa total;
- $C$  – É a matriz de Coriolis;
- $D$  – É a matriz de amortecimento hidrostático;
- $g$  – É o vetor geral de restauração de força;

As transformações em *Body* e *NED* (Norte, Leste, Descer) são essenciais para a modelagem de veículos subaquáticos, pois permitem a conversão entre diferentes sistemas de coordenadas utilizados na navegação e controle do ROV. Essas transformações são cruciais para garantir que

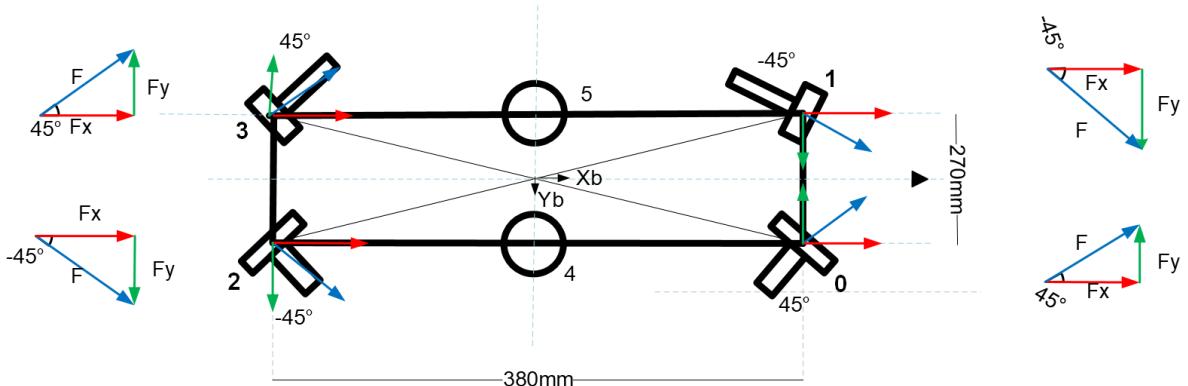
os cálculos de movimento e controle sejam precisos e consistentes, independentemente do referencial adotado. Sendo *Body* o referencial atrelado ao veículo, e *NED* o referencial fixo ao solo. A Equação 1 representa a cinemática do sistema, descrevendo assim os aspectos geométricos da movimentação do ROV em relação a diferentes coordenadas, enquanto a Equação 2 representa a cinética do modelo, que consiste na análise de forças e momentos incluídas no ROV durante o movimento.

Baseando-se nas equações de cinemática e cinética para 6 DOFs, é possível modelar o sistema do *BlueROV2 Standard*, que só é capaz de se movimentar em 4 DOFs. Dessa forma, os graus de liberdade que não são acessados pelo robô são zerados nas matrizes.

## 2.1 Modelo e Alocação de *Thruster*

O modelo de ROV utilizado (*BlueROV2 standard*) conta com 6 *thrusters*, como é possível visualizar na Figura 2, sendo 4 horizontais e 2 verticais, tendo sua configuração ilustrada também na Figura 2. Dessa forma, os *thrusters* dianteiros (0 e 1) giram no sentido anti-horário e os traseiros (2 e 3) giram no sentido horário.

Figura 2: Diagrama de *thruster* e forças, vista superior dos *thrusters*



Fonte: Autores.

Apesar da configuração física atípica dos *thrusters*, posicionados em direções opostas, a alocação de forças é corretamente realizada graças à rotação em sentidos contrários. Caso todos os *thrusters* rotacionassem no mesmo sentido, essa disposição geraria forças opostas, comprometendo o controle do sistema. No entanto, como evidenciado na Figura 2, a rotação alternada dos *thrusters* soluciona esse problema, permitindo a geração das forças desejadas.

Conforme apresentado por Wu (2018), o modelo de um *thruster* pode ser apresentado de forma linear pela Equação 3. Onde a força do *thruster* é representada pelo Vetor 4, os *inputs* de controle são representados pelo Vetor 5, e os coeficientes de *thruster* são representados pela Matriz diagonal 6.

$$F = Ku \quad (3)$$

$$F = [F_1 \ F_2 \ \dots \ F_n]^T \quad (4)$$

$$u = [u_1 \ u_2 \ \dots \ u_n]^T \quad (5)$$

$$K = diag [K_1 \ K_2 \ \dots \ K_n] \quad (6)$$

Considerando o Vetor de força  $f$  em 7 e o Vetor de momentos  $r$  em 8, podemos calcular as forças e momentos em 6 DOFs pela seguinte fórmula:

$$f = [f_x \ f_y \ f_z]^T \quad (7)$$

$$r = [l_x \ l_y \ l_z]^T \quad (8)$$

$$\tau = \begin{bmatrix} f \\ r \times f \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ F_z l_y - F_y l_z \\ F_x l_z - F_z l_x \\ F_y l_x - F_x l_y \end{bmatrix} \quad (9)$$

Considerando a limitação do BlueROV2 standard para somente 4 DOFs, é possível ajustar as Equações 8 e 9 para obter as forças em 4 DOFs como desejado. Sendo necessário zerar os momentos que não serão aplicados ao ROV, tem-se então como resultado:

$$r = [0 \ 0 \ l_z]^T \quad (10)$$

$$\tau = \begin{bmatrix} f \\ r \times f \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ F_y l_x - F_x l_y \end{bmatrix} \quad (11)$$

A alocação de controle é então modelada por:

$$\tau = T(\alpha)F = T(\alpha)Ku \quad (12)$$

Sendo  $T$  a matriz de alocação, onde  $T \in \mathbb{R}^{4 \times 6}$  e  $\alpha$  o vetor de ângulos de rotação dos *thrusters*, onde  $\alpha \in \mathbb{R}^6$ . Como consequência, a matriz de configuração de *thruster*  $T$  pode ser calculada usando a Equação 11. Uma vez determinadas as forças e momentos dos *thrusters*, é possível formalizar o problema da alocação, cujo objetivo é distribuir corretamente os esforços para os propulsores, para realizar a ação de controle desejada.

## 2.2 Alocação de controle

Wu (2018) define a alocação de controle como o processo que computa o sinal de entrada do controle  $u$  e o aplica nos *thrusters*, de forma que o controle geral de forças  $\tau$  possa ser gerado. Partindo da Equação 12, é possível calcular o vetor de entradas de controle através da seguinte equação:

$$u = K^{-1}T^{-1}\tau \quad (13)$$

Contudo, levando em conta que a matriz  $T$  é uma matriz não quadrática, é aplicada a Moore-Penrose pseudo-inversa  $T^+$ , dada por:

$$T^+ = T^T(TT^T)^{-1} \quad (14)$$

Assim, podemos obter o vetor de entradas do controle por:

$$u = K^{-1}T^+\tau \quad (15)$$

## 2.3 Formulação da Otimização

Em *Efficient optimal constrained control allocation via multiparametric programming* (JOHANSEN; FOSSEN; TONDEL, 2005) é sugerida uma formulação de otimização para o problema de alocação de controle. É considerado então o seguinte problema de otimização:

$$\min_{u,s,\tilde{u}} (s^T Q s + u^T W u + \beta \tilde{u}) \quad (16)$$

A Equação 16 está sujeita ao seguinte:

$$Tu = \tau + s \quad (17)$$

$$u_{\min} \leq u \leq u_{\max} \quad (18)$$

$$-\tilde{u} \leq u_1, u_2, \dots, u_N \leq \tilde{u} \quad (19)$$

A variável  $s$  é o termo que garante a restrição proposta em 18, que faz com que o resultado da força generalizada  $Tu$  desvie das especificações de  $\tau$ , caso seja necessário. O segundo termo do critério corresponde ao critério de mínimos quadrados, enquanto o terceiro termo minimiza a maior força entre os atuadores, devido a 19. O parâmetro  $\beta \geq 0$  controla a ponderação relativa desses dois critérios, permitindo que sejam tratados os compromissos entre o uso médio e o pior caso do controle. Os parâmetros  $u_{\min}$  e  $u_{\max}$  representam os limites mínimos e máximos de cada *thruster*, respectivamente e o parâmetro  $u_N$  representa o limite absoluto de magnitude para cada *thruster* no termo de minimização  $\beta \tilde{u}$ .

## 3 METODOLOGIA

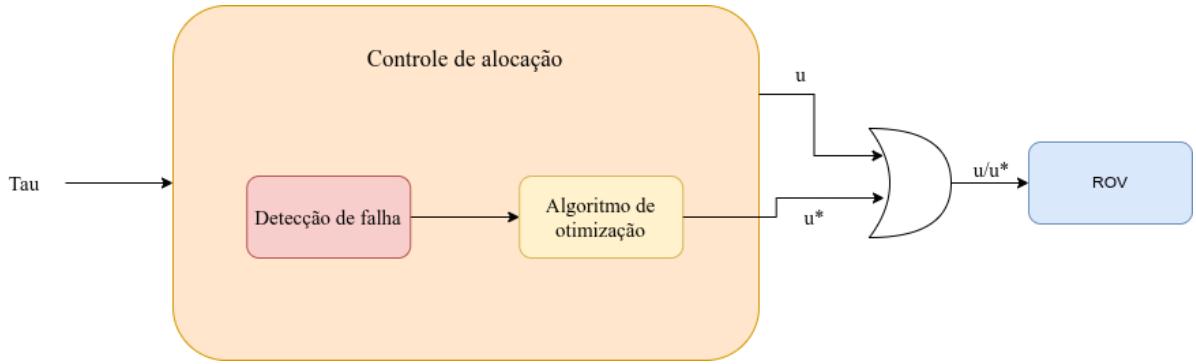
A metodologia proposta neste trabalho consiste no desenvolvimento e validação de um algoritmo de otimização voltado à redistribuição de forças em veículos subaquáticos (*BlueROV2*), com o objetivo de manter a navegabilidade do sistema em caso de falha total de um ou mais *thrusters*.

Os experimentos foram conduzidos em ambiente de simulação, utilizando o sistema operacional Ubuntu 22.04, o simulador Gazebo Ignition 6.16.0 e o framework *ROS 2 Control 2.49.0 jammy*. O modelo *BlueROV2 Standard* foi adotado como base para representar o comportamento dinâmico do veículo e verificar o desempenho do método proposto. O controle do ROV é feito por um controlador de alocação de esforços, responsável por distribuir as forças entre os *thrusters*. O algoritmo de otimização será integrado em paralelo com o controle de alocação, de modo a realizar a redistribuição adaptativa das forças quando houver falha em um ou mais propulsores. Vale destacar que o reconhecimento de falha não é abordado neste trabalho, por ser um desafio à parte da otimização das forças.

O diagrama do fluxo de controle proposto é apresentado na Figura 3, destacando a interação entre o controlador de alocação e o algoritmo de otimização, bem como as entradas e saídas do sistema, é exposto o paralelismo entre o controle e o otimizador, de forma que o algoritmo de otimização só é ativado a partir da identificação da falha pelo detector.

O desempenho do algoritmo será avaliado com base na comparação entre o erro médio obtido com e sem o algoritmo e o desvio padrão do erro, considerando dois critérios principais: a velocidade de movimentação do ROV em determinada direção e o erro de deslocamento, em caso de falha de um ou mais *thrusters*, para determinada direção.

Figura 3: Fluxo de controle.



Fonte: Autores.

A partir dessas análises é esperado verificar a robustez e a eficiência do método de otimização proposto em comparação ao controle convencional sem a tolerância a falhas.

### 3.1 Metodologia experimental

A validação do algoritmo de otimização será realizada por meio do seguinte procedimento experimental:

Foram estabelecidos três cenários de teste, cinco casos de falhas e quatro trajetórias, conforme detalhado na Tabela 1:

Tabela 1: Configuração experimental: cenários, casos de falhas e trajetórias

Cenários	Casos de Falhas		Trajetórias
	ID	Descrição ( <i>thrusters</i> )	
Sem falhas e sem otimização (caso 0)	1	Dois <i>thrusters</i> horizontais coincidentes (0 e 2)	<i>Surge</i> (X)
Com falhas e sem otimização	2	Dois <i>thrusters</i> horizontais paralelos (3 e 2)	<i>Sway</i> (Y)
Com falhas e com otimização	3	Dois <i>thrusters</i> horizontais diagonais (0 e 3)	<i>Heave</i> (Z)
	4	Um <i>thruster</i> horizontal e um vertical (0 e 4)	<i>Yaw</i> (rot. Z)
	5	Um <i>thruster</i> horizontal (0)	

Fonte: Autores.

Os testes foram realizados para o intervalo de tempo de 30 segundos para cada tipo de trajetória, a fim de observar o comportamento do sistema ao longo do tempo. Cada combinação de cenário, caso e tempo foi repetida cinco vezes.

O método de teste para *Heave* e *Yaw* foi diferente dos outros, devido a limitações físicas do ROV. Para o movimento de *Heave*, foi somente realizado o caso 4, pois esse é o único que afeta o movimento vertical do ROV, e para o movimento de *Yaw*, foram realizados todos os casos,

mas os dados de interesse foram voltados à orientação do ROV, ou seja, quantidade de rotações feitas, velocidade de rotação, e capacidade de manter a orientação desejada.

As trajetórias são realizadas a partir de uma força tipo *wrench* aplicada nos *thrusters* do ROV, o que gera um movimento relacionado ao tipo de trajetória, ou seja, a trajetória *surge* não se trata do movimento perfeito naquela direção, mas sim de uma força em X, gerando um movimento naquela direção, ainda estando sujeito às forças que ocorrem no ambiente simulado, como por exemplo aquelas geradas pela variação de centro de massa e centro de flutuabilidade.

Os dados coletados serão analisados estatisticamente, utilizando métricas como o erro médio e desvio padrão, para analisar o efeito do algoritmo de otimização em relação a cada tipo de falha em relação às trajetórias quando comparadas ao caso original de pleno funcionamento. Essas análises permitirão avaliar a eficácia do algoritmo de otimização na manutenção da navegabilidade do ROV em diferentes cenários de falhas.

## 4 RESULTADOS E DISCUSSÃO

Nesta seção, são apresentados os resultados obtidos a partir dos experimentos conduzidos para avaliar o desempenho do algoritmo de otimização proposto na metodologia. Os resultados são organizados em tabelas que ilustram o impacto do algoritmo na navegabilidade do ROV em diferentes cenários de falhas.

### 4.1 Combinações Mínimas de *Thrusters*

Na Tabela 2, são apresentadas as combinações mínimas de *thrusters* necessários para manter a navegabilidade do ROV em cada direção de movimento (*surge*, *sway*, *heave*, *yaw*), considerando os diferentes casos de falhas simuladas.

Tabela 2: Combinações mínimas de *thrusters* por direção (*surge*, *sway*, *heave*, *yaw*)

Direção	Combinação Mínima de <i>Thrusters</i>
<b>Surge</b>	Dois <i>thrusters</i> paralelos ou diagonais
<b>Sway</b>	Dois <i>thrusters</i> laterais ou diagonais
<b>Heave</b>	Dois <i>thrusters</i> verticais
<b>Yaw</b>	Dois <i>thrusters</i> diagonais

Fonte: Autores.

A partir da Tabela 2, é possível observar que alguns casos de falhas caem fora das combinações mínimas necessárias para manter a navegabilidade do ROV em determinadas direções. Isso indica que, em tais situações, o algoritmo de otimização pode não redistribuir as forças de maneira eficaz para compensar a ausência dos *thrusters* falhos, resultando na incapacidade do ROV de se mover conforme desejado.

Destaca-se o Caso 3 como um cenário particular, onde o algoritmo de otimização apresenta impacto limitado em todas as direções de movimento (*surge*, *sway* e *yaw*), com desempenho equivalente ou apenas marginalmente superior ao caso sem otimização. Essa característica sugere que a configuração específica de falhas do Caso 3 permite ao ROV manter navegabilidade adequada independentemente da aplicação do algoritmo, vindo da redundância natural dos *thrusters* restantes.

## 4.2 Análise dos Resultados por Direção de Movimento

A Tabela 3 apresenta os erros médios obtidos após as repetições dos testes para a direção *Surge*. Os erros foram obtidos comparando a trajetória ideal sem o algoritmo de otimização com as trajetórias realizadas em cada caso de falha, tanto com o algoritmo de otimização (COM) quanto sem o algoritmo (SEM).

Nos casos apresentados a seguir, é possível observar que os casos que apresentam o otimizador apresentaram erros menores em relação à velocidade e à distância final alcançada, mostrando a capacidade do algoritmo em manter a navegabilidade do ROV mesmo em situações de falha nos *thrusters*. O algoritmo demonstra especial eficácia na redução da variabilidade dos resultados, como evidenciado no Caso 2, onde o desvio padrão da velocidade foi reduzido de 0,008 sem otimização, para 0,004 com otimização (redução de 50%) e o desvio padrão da distância percorrida foi reduzido de 0,247 m para 0,115 m (redução de 53%) com o uso da otimização.

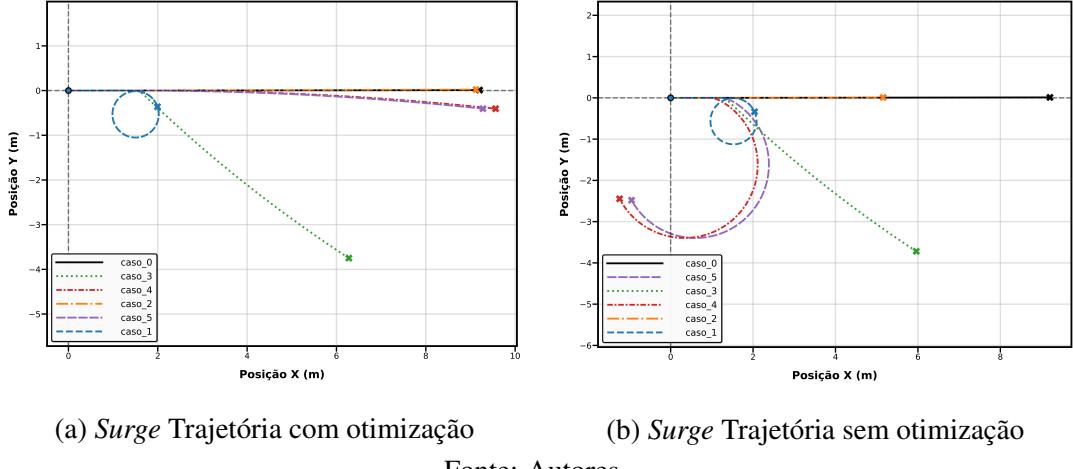
Tabela 3: Erro médio por Direção (*Surge*)

Caso		Vel (m/s)	Distância (m)	X (m)	Y (m)	Z (m)	yaw (°)
1	COM	0,005	0,448	-7,198	1,040	0,013	-360
	SEM	0,096	3,475	7,105	-1,116	0,014	-360
2	COM	-0,025	-0,024	-0,068	-0,01	0,041	-0,1
	SEM	0,091	3,630	3,592	0,00	0,038	0
3	COM	0,037	1,716	5,237	-3,770	-0,088	-11,4
	SEM	0,033	1,684	3,003	-3,820	0,006	-11,6
4	COM	-0,006	-0,171	-0,209	-0,49	-0,002	-6,9
	SEM	0,027	0,939	5,823	-3,39	0,012	-360
5	COM	-0,007	-0,008	-0,008	-0,38	0,012	-6,18
	SEM	0,025	1,129	5,862	-3,39	0,016	-360

Fonte: Autores.

A figura 4 ilustra as trajetórias do ROV para a direção *Surge* para todos os 5 casos de falha e o caso ideal, tanto com o algoritmo de otimização em ação quanto sem o algoritmo. É possível visualizar que a trajetória do caso 1, em azul, se destaca entre as demais trajetórias do caso com otimização por não ter sido capaz de manter a trajetória em *surge*, isso ocorre devido ao tipo de falha do caso 1, pois a falha do caso 1 não obedece à combinação mínima de *thrusters* para a movimentação em *surge*, o ROV não é capaz de se movimentar adequadamente naquela direção, mesmo com o algoritmo de otimização ativo. Ainda na figura 4, é possível observar que as trajetórias dos casos sem otimização, em comparação com a trajetória ideal, em preto, apresentam maior dificuldade em se manter na trajetória, nos casos em que a trajetória se mantém, a distância percorrida no período de teste é significativamente menor do que a ideal, mostrando assim a eficiência do algoritmo de otimização em manter a navegabilidade do ROV.

Figura 4: Trajetórias do ROV em *Surge* com e sem otimização



Fonte: Autores.

A tabela 4 apresenta os resultados dos erros médios medidos na direção *sway*. Na trajetória *sway*, observa-se o mesmo comportamento dos resultados anteriores, onde os casos que apresentam o otimizador apresentam erros menores, com um desvio de direção menor quando comparados aos mesmos casos sem o otimizador. Destaca-se particularmente o Caso 4, onde a aplicação do algoritmo de otimização resultou em uma redução significativa do desvio padrão da distância percorrida, de 0,324 sem otimização, para 0,050 com otimização (redução de 84,75%), demonstrando maior consistência e precisão no controle do movimento lateral.

Tabela 4: Erro médio por Direção (*Sway*)

Caso		Vel (m/s)	Distância (m)	X (m)	Y (m)	Z (m)	yaw (°)
1	COM	0,006	-0,100	-0,051	-0,100	0,006	-0,44
	SEM	0,055	1,870	0,844	1,694	0,001	10,72
2	COM	-0,006	0,248	0,714	5,500	0,026	-333,8
	SEM	0,052	1,897	0,470	5,450	0,016	-333,8
3	COM	0,003	-0,156	-3,885	2,40	-0,015	11,2
	SEM	0,004	0,123	-3,368	2,46	-0,001	10,92
4	COM	-0,005	-0,007	-0,313	0,10	0,011	-6,26
	SEM	0,009	0,334	-1,013	4,28	0,003	-333,8
5	COM	-0,008	-0,206	-0,767	-0,07	0,004	-7,46
	SEM	0,010	0,484	-1,011	4,28	0,012	-333,8

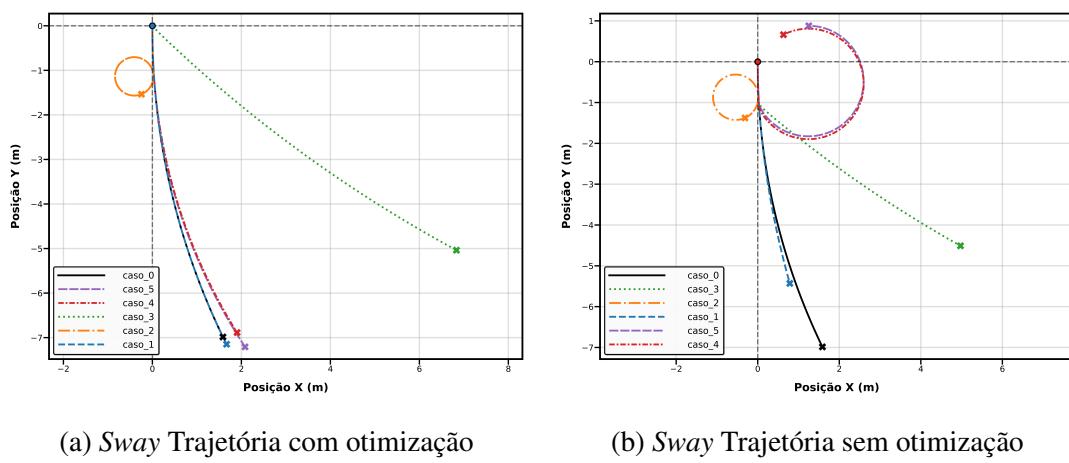
Fonte: Autores.

Na figura 5 ilustra as trajetórias do ROV para a direção *Sway* para todos os 5 casos de falha e o caso ideal, tanto com o algoritmo de otimização em ação quanto sem o algoritmo. É possível visualizar que a trajetória do caso 2, em vermelho, tem o mesmo comportamento do caso 1 em *surge*, como não obedece a combinação mínima de *thrusters* para a movimentação em *sway*, o ROV não é capaz de se movimentar adequadamente naquela direção, mesmo com o algoritmo de otimização ativo.

Contudo, ao analisar a tabela 4, é possível visualizar que, mesmo com a incapacidade de se mover na direção correta, o erro entre a distância percorrida ideal e a distância percorrida no teste com o otimizador ativo e de apenas 0,248 m, indicando que o otimizador ainda atua na navegabilidade do ROV, possibilitando que o mesmo percorra uma distância próxima ao ideal no mesmo intervalo de tempo mesmo com falha; porém, a limitação física da distribuição dos *thrusters* no modelo não permite que o ROV siga na direção correta, mesmo comportamento apresentado no caso 1 em *surge*.

Ainda na figura 5, é possível observar que as trajetórias dos casos sem otimização, apresentam comportamentos similares aos observados na direção *surge*, com desvios significativos em relação à trajetória ideal, evidenciando a dificuldade do ROV em manter o curso desejado sem o suporte do algoritmo de otimização.

Figura 5: Trajetórias do ROV em *Sway* com e sem otimização



Fonte: Autores.

Na tabela 5 são apresentados os resultados dos erros médios obtidos na direção *yaw*. Considerando a combinação ideal da trajetória *yaw* como sendo a combinação diagonal de *thrusters*, os resultados reforçam que a combinação do caso 3 obteve os melhores resultados nos testes. Contudo, o impacto do algoritmo de otimização na trajetória *yaw* é menos pronunciado em comparação com as outras direções, devido às características da movimentação rotacional do ROV juntamente com as limitações físicas do veículo em situação de falhas nos *thrusters*. A complexidade inerente do movimento rotacional é evidenciada pelos elevados desvios padrão observados, como no Caso 1, onde o desvio da velocidade angular ( $1,272^{\circ}/s$ ) e do ângulo *yaw* ( $38,261^{\circ}$ ) são consideravelmente superiores aos desvios típicos dos movimentos lineares. Como métrica foi observada a capacidade do ROV completar o número de voltas mais próximo do ideal, a capacidade de se manter na orientação correta sem se desviar nas outras direções e a velocidade média de rotação.

Assim nesse contexto foi observado que os casos que apresentavam somente um *thruster* em falha (casos 4 e 5) apresentaram resultados mais satisfatórios, pois além de atenderem à combinação mínima de *thrusters* para a movimentação em *yaw*, apresentam um *thruster* a mais em funcionamento, o que possibilita uma melhor redistribuição das forças, nesses casos apesar de apresentarem resultados melhores que os demais casos (1 e 2) a presença da otimização não foi capaz de melhorar os resultados quando comparados ao cenário sem otimização, indicando que há uma limitação do algoritmo em melhorar a movimentação em *yaw* em situações de falhas nos *thrusters*. Apesar disso, o algoritmo de otimização ainda conseguiu manter a navegabilidade

do ROV, seguindo assim a trajetória desejada, mas com eficiência reduzida.

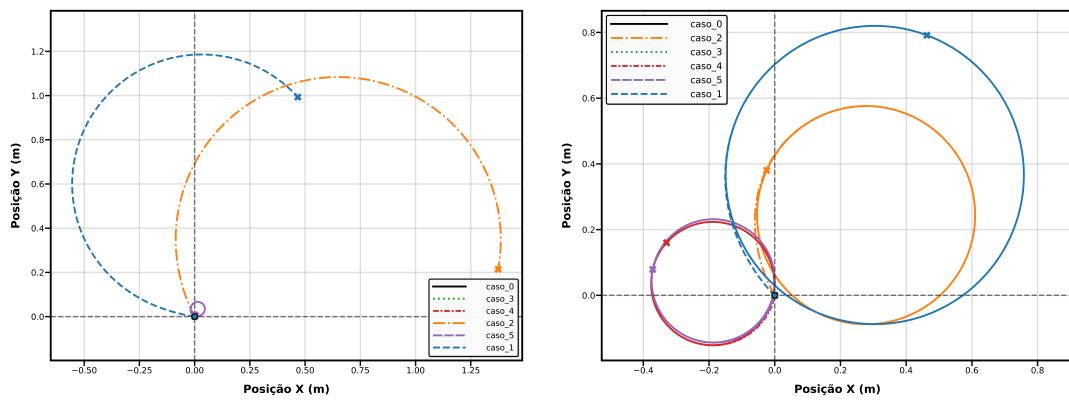
A figura 6 ilustra as trajetórias do ROV para a direção *Yaw* em todos dos casos de falha, incluindo o caso ideal sem falhas (caso 0). Observa-se que, assim como mostrado na tabela 5, o otimizador não apresenta uma melhora significativa na trajetória do ROV em *Yaw*, sendo incapaz de manter a trajetória no caso 1, e apesar de conseguir completar a trajetória nos casos 3 e 5, ainda apresenta desvios consideráveis em relação à trajetória ideal, não sendo capaz de completar o mesmo número de voltas, o que pode ser visto na tabela 5. Analisando ainda a tabela 5, o algoritmo não foi capaz de melhorar a velocidade média de rotação mesmo nos casos que atendem às combinações mínimas de *thrusters* para a movimentação em *Yaw* (casos 3 e 5), sendo capaz de garantir que o ROV complete as trajetórias, mas não no mesmo período de tempo e com uma velocidade significativamente reduzida.

Tabela 5: Erro médio por Direção (*Yaw*)

Caso		Vel (°/s)	Nº de Voltas	X (m)	Y (m)	Z (m)	yaw (°)
1	COM	92,360	8,020	-1,026	-1,101	0,006	2897,22
	SEM	48,020	4,180	-0,910	-0,914	0,002	1502,48
2	COM	99,78	9,020	-1,440	-1,31	0,008	3138,84
	SEM	49,120	4,3	-0,672	-0,67	0,001	1550,44
3	COM	44,880	3,900	-0,001	0,00	0,008	1399,74
	SEM	49,300	4,34	-0,001	0,00	0,012	1556,72
4	COM	41,060	2,720	-0,069	-0,06	-0,001	1235,02
	SEM	25,060	2,26	-0,376	-0,38	0,007	806,1
5	COM	41,360	3,520	-0,066	-0,07	0,002	1265,32
	SEM	25,580	2,14	-0,375	-0,38	-0,001	775,62

Fonte: Autores.

Figura 6: Trajetórias do ROV em *Yaw* com e sem otimização



(a) *Yaw* Trajetória com otimização

(b) *Yaw* Trajetória sem otimização

Fonte: Autores.

Os valores de desvio padrão em *yaw* demonstram comportamento significativamente mais complexo e instável em relação aos movimentos lineares, confirmando a natureza desafiadora do controle rotacional em situações de falha. Por exemplo, no Caso 2, o desvio padrão da velocidade angular atinge 7,190°/s (COM) comparado a 1,602°/s (SEM), enquanto o desvio do ângulo *yaw* alcança valores extremos de 222,878 (COM) versus 54,308 (SEM), ordens de magnitude superiores aos desvios observados nas direções *surge* e *sway*, que tipicamente ficam abaixo de 1,0 . As variáveis lineares (X, Y e Z) mantêm dispersão mínima durante as rotações, sugerindo boa estabilidade posicional, mas a variabilidade angular evidencia os desafios inerentes ao controle de atitude em condições de falha dos *thrusters*.

Em relação à manutenção da trajetória ideal, o otimizador enfrenta dificuldades devido às limitações geradas pela disposição dos *thrusters* do ROV, essas limitações são refletidas nos casos que não respeitam as combinações mínimas de *thrusters*, e na trajetória *heave*.

A combinação mínima exigida para que haja o funcionamento ideal da trajetória *heave* são de 2 *thrusters* como indicado na tabela 2, mas o ROV utilizado nos testes apresenta somente dois *thrusters* verticais em sua estrutura, impossibilitando que seja realizada a otimização no caso de falha de um dos *thrusters* verticais, o que impossibilita a movimentação ideal na direção *heave*. Em decorrência dessa limitação física do ROV, os resultados obtidos para a trajetória *heave* não são apresentados nesta seção.

## 5 CONSIDERAÇÕES FINAIS

O presente trabalho abordou o desenvolvimento e a validação de um algoritmo de otimização para tolerância à falha em veículos subaquáticos, mais especificamente no *BlueROV2 Standard*, que possui 4 DOFs e 6 *thrusters*. O objetivo principal foi conservar a navegabilidade do veículo em casos de falha total de um ou mais *thrusters*. A metodologia proposta envolveu a implementação do algoritmo em um ambiente de simulação, e com os testes foi possível constatar a eficiência do algoritmo proposto, que sofre com limitações causadas pela configuração de *thrusters* do veículo. Os resultados obtidos demonstram que o algoritmo de otimização proposto é eficaz na redistribuição das forças entre os *thrusters* remanescentes, permitindo que o ROV mantenha sua navegabilidade mesmo em situações de falha, desde que as condições mínimas de navegabilidade sejam mantidas.

O otimizador demonstra sua eficiência principalmente no âmbito da velocidade média e distância percorrida, mesmo em casos onde a combinação mínima de *thrusters* não é atendida. Como o *BlueROV2 Standard* possui uma configuração de 6 *thrusters*, o otimizador encontra limitações na redistribuição das forças quando múltiplos *thrusters* falham, o que causa uma dificuldade em manter a trajetória ideal, sendo possível visualizar erros significativos em *Yaw* nos casos que não atendem às combinações mínimas para aquela direção. Nos casos que atendem às combinações mínimas, o otimizador é capaz de diminuir o erro médio em *yaw*, que é causado pelas falhas dos *thrusters*, mantendo assim a capacidade do ROV de seguir a trajetória desejada.

O caso 3 se destaca por apresentar erros muito próximos entre os cenários com e sem otimização em qualquer trajetória, indicando que a falha diagonal representada pelo caso 3 tem a capacidade de se autocompensar devido ao posicionamento dos *thrusters* no ROV, o que minimiza o impacto da falha na capacidade de movimentação do veículo. O maior fator de melhoria observado para o algoritmo de otimização é observado tanto na distância percorrida quanto na velocidade média, mantendo a capacidade do ROV de manter a velocidade o mais próxima do ideal possível e preservando também a capacidade de percorrer distâncias similares ao caso ideal, mesmo com a falha de *thrusters*, para todos os casos na maioria das trajetórias, enfren-

tando limitações somente na trajetória em *yaw*.

Apesar das dificuldades enfrentadas pelo otimizador em manter a trajetória ideal em todos os casos, os resultados obtidos indicam que o algoritmo proposto é eficaz na redistribuição das forças entre os *thrusters* remanescentes, permitindo que o ROV mantenha sua navegabilidade mesmo em situações de falha, desde que as condições mínimas de navegabilidade sejam mantidas.

Sendo assim, o trabalho contribui para o avanço da segurança operacional de ROVs em aplicações industriais submarinas, oferecendo uma solução viável para mitigar os efeitos de falhas em *thrusters* e garantindo a continuidade das operações em ambientes desafiadores. Para trabalhos futuros, visa-se a implementação do algoritmo no ROV em sua configuração *heavy*, que conta com 8 *thrusters*, o que pode proporcionar uma maior flexibilidade na redistribuição das forças e potencialmente melhorar o desempenho do algoritmo em cenários de falha múltipla, permitindo também que a otimização de *heave* seja realizada. Também é possível realizar a implementação do algoritmo no *BlueROV2* real, para validação dos resultados obtidos em simulação, e observação dos efeitos do algoritmo no mundo real, sendo possível combinar a existência do algoritmo de otimização com controladores que permitirão ao ROV compensar os desvios de trajetória causados pela não correspondência dos *thrusters* mínimos de operação.

## AGRADECIMENTOS

## REFERÊNCIAS

- BIAZON, Tássia. **Nas profundezas dos oceanos**. Revista Marítima Brasileira, v. 137, n. 7-9, p. 66–82, 2017.
- CHRIST, Robert D.; WERNLI SR, Robert L. **The ROV manual: a user guide for remotely operated vehicles**. Oxford, Reino Unido: Butterworth-Heinemann, 2013.
- FOSSEN, Thor I. **Handbook of marine craft hydrodynamics and motion control**. Chichester: John Wiley & Sons, 2021.
- JOHANSEN, Tor A.; FOSSEN, Thor I.; TONDEL, Petter. **Efficient optimal constrained control allocation via multiparametric programming**. Journal of Guidance, Control, and Dynamics, v. 28, n. 3, p. 506–515, 2005.
- OCCUPATIONAL SAFETY AND HEALTH ADMINISTRATION. **Commercial Diving: Hazards and Solutions**. Washington, D.C., EUA: U.S. Department of Labor, 2023.
- PETROBRAS. **Relatório de sustentabilidade Petrobras 2024**. Rio de Janeiro, Brasil: Petrobras, 2024.
- WU, Chu-Jou. **6-DOF modelling and control of a remotely operated vehicle**. 2018. – Flinders University, College of Science e Engineering, Adelaide, Australia.
- XU, Jiankang; WANG, Nan. **Optimization of ROV control based on genetic algorithm**. In: OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO). Kobe, Japão: IEEEExplore, 2018. p. 1–4.