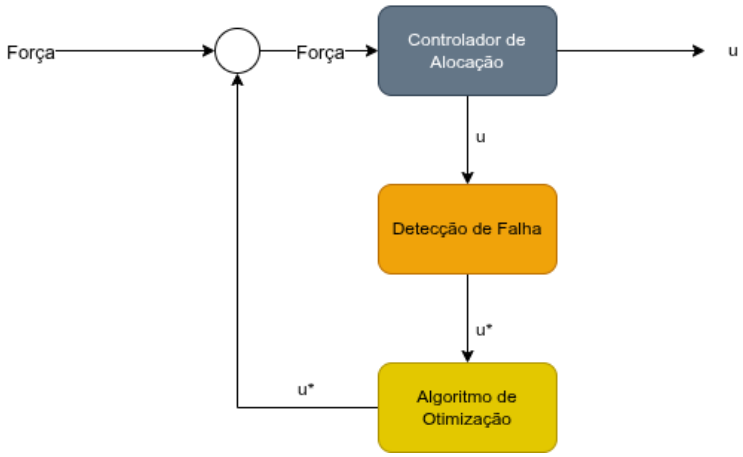
A metodologia proposta neste trabalho consiste no desenvolvimento e validação de um algoritmo de otimização voltado à redistribuição de forças em veículos subaquáticos (*BlueROV2*), com o objetivo de manter a navegabilidade do sistema em caso de falha total de um ou mais *thrusters*.

Os experimentos foram conduzidos em ambiente de simulação, utilizando o sistema operacional Ubuntu 22.04, o simulador Gazebo Ignition e o framework ROS 2 Control. 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 1, destacando a interação entre o controlador de alocação e o algoritmo de otimização, bem como as entradas e saídas do sistema. O desempenho do algoritmo será avaliado com base no erro de Pearson e o coeficiente de determinação r^2 e no desvio padrão do erro de cada repetição.

Figura 1: 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.

0.1 Metodologia experimental

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

Foram estabelecidos três cenarios de teste, variando conforme a presença ou ausência do algoritmo de otimização e a ocorrência de falhas nos *thrusters*. Os cenarios foram organizados da seguinte forma:

- Cenário 1: Trajetória sem falhas e sem o algoritmo de otimização;

- Cenário 2: Trajetória com falhas e sem o algoritmo de otimização;
- Cenário 3: Trajetória com falhas e com o algoritmo de otimização.

Cada cenário serão aplicados em cinco casos de falhas, sendo esses:

- Caso 1: Falha em dois *thrusters* horizontais paralelos;
- Caso 2: Falha em dois *thrusters* horizontais coincidentes;
- Caso 3: Falha em dois *thrusters* horizontais diagonais;
- Caso 4: Falha em um *thruster* horizontal e um vertical;
- Caso 5: Falha em um *thruster* horizontal.

Os testes foram realizados em três intervalos de tempo (30, 60 e 90 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, totalizando 225 execuções experimentais. Foram delimitadas 2 trajétórias para o ROV seguir, sendo elas:

- Trajetória 1: Movimento circular;
- Trajetória 2: Movimento quadrático.

As trajetorias são realizadas a partir de uma força de *wrench* aplicada nos *thrusters* do ROV, o que gera um movimento relacionado ao tipo de trajétorio, ou seja, a trajétoria circular não se trata de um movimento perfeito, mas sim de uma força aplicada que gere um movimento aproximado ao circular, o mesmo é aplicado na trajétoria quadrática.

Após as 225 execuções, os dados coletados serão analisados estatisticamente, utilizando métricas como o erro de Pearson e r^2, para a correlação e similaridade entre os casos com falhas e o caso original de pleno funcionamento, e o desvio padrão do erro para cada repetição. Essas análises permitirão avaliar a eficácia do algoritmo de otimização na manutenção da navegabilidade do ROV em diferentes cenários de falhas.