UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

ALEX KENJI UYEDA MAJIMA
LUCAS HIDEKI SAKURAI

Controlador baseado em redes neurais para navegação em canal de acesso treinado por dados de manobras realizadas por práticos

São Paulo
2018

ALEX KENJI UYEDA MAJIMA
LUCAS HIDEKI SAKURAI

Controlador baseado em redes neurais para navegação em canal de acesso treinado por dados de manobras realizadas por práticos

Versão original

Monografia apresentada ao Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro.

Área de concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Eduardo Aoun Tannuri

São Paulo
2018

Catalogação-na-publicação

Majima, Alex Kenji Uyeda
Controlador baseado em redes neurais para navegação em canal de acesso treinado por dados de manobras realizadas por práticos / A. K. U. Majima, L. H. Sakurai -- São Paulo, 2018.
86 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Redes neurais 2.Simulação 3.Navios I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t. III.Sakurai, Lucas Hideki

Este relatório é apresentado como requisito parcial para obtenção do grau de engenheiro mecatrônico na Escola Politécnica da Universidade de São Paulo. É o produto do meu próprio trabalho, exceto onde indicado no texto. O relatório pode ser livremente copiado e distribuído desde que a fonte seja citada.

Alex Kenji Uyeda Majima

Lucas Hideki Sakurai

Agradecimentos

Ao nosso orientador, Prof. Dr. Eduardo Aoun Tannuri, pelo apoio, incentivo e confiança durante todo o desenvolvimento do projeto.

Aos nossos familiares que sempre apoiaram as nossas decisões e deram todo o suporte para alcançarmos os nossos objetivos.

Aos nossos amigos do IFSP e da EPUSP que nos acompanharam durante os últimos anos compartilhando momentos incríveis.

Ao corpo técnico e alunos do TPN, especialmente ao mestrando José Amendola Netto Andrade, pelo suporte e pelas sugestões ao projeto.

Resumo

O presente trabalho visa abordar o complexo problema de controle de embarcações marítimas que adentram um porto por meio de um canal de acesso. Devido a essa complexidade, profissionais conhecidos como práticos realizam as manobras para garantir agilidade e segurança com a tarefa de conduzir a embarcação dentro dos limites do canal e reduzir sua velocidade até o final do mesmo permitindo atracar no porto. De forma a usufruir da experiência desses profissionais e simular a sua tomada de decisões de comandos, optou-se pela utilização de redes neurais como forma de controle, pois esta técnica requer o treinamento de uma arquitetura de rede que adere aos dados de referência, no caso, a sequência de comandos de simulações realizadas por práticos. A primeira etapa do projeto foi a elaboração desses dados de forma a garantir a sua qualidade e adequação aos parâmetros de entrada e de saída planejados. A segunda etapa foi o ciclo iterativo de definição e ajuste da arquitetura por meio do treinamento e avaliação de performance. A terceira e última etapa foi criar uma interface para validação por simulação e possibilitar a interação da rede neural com o integrador numérico de forma autônoma. Como forma de desenvolver a rede foi utilizado a linguagem de programação *Python* associado ao *framework* chamado *TensorFlow*. Desta maneira, o documento discorre sobre as fases de elaboração e a possibilidade de flexibilizar a automação de embarcações pela utilização de rede neurais para um mais amplo conjunto de condições, o que poderá aproximar a tecnologia atual para condições mais realistas, garantir maior segurança ao reduzir a interferência humana e otimizar o tempo de espera e de entrada nos portos.

Palavras-chaves: Redes neurais. Simulação. Navios.

Abstract

This work aims to tackle the complex task of ship handling which berth in a port through an access channel. Due to this complexity, professionals known as maritime pilots perform those maneuvers to ensure agility and safety while travelling through the channel's limits and reducing its speed until the end allowing it to berth in the port. In order to benefit from the experience of those professionals and simulate their command decision making, it was opted for the application of neural networks as means of control, since this technique requires the training of a network architecture which will adhere to the reference data, in this case, the command sequence from simulations performed by maritime pilots. The first stage of the project consisted of preparing these data in order to guarantee its quality and suitability to the intended input and output parameters. The next stage was the interactive cycle of definition and adjustment of the architecture through training and performance evaluation. The last stage was the development an interface for validation through simulation and enable an autonomous interaction between the neural network and the numerical integrator. As means of development of the network the programming language *Python* combined with the framework called *TensorFlow* was used. Thus, this document discusses the development stages and the possibility of easing the automation of vessels using neural networks for a wider set of conditions, which may bring closer the current technology to more realistic conditions, ensure more safety as human interference is lessen and optimize the waiting time and berthing at ports.

Key-words: Neural networks. Simulation. Vessel.

Lista de figuras

Lista de tabelas

Sumário

1 Introdução

1.1 Descrição do tema e motivação

O tema do projeto é a utilização de redes neurais como uma forma de controle para navegação de embarcações marítimas. A proposta surgiu da complexa tarefa de realizar a manobra de navios nos portos. Quando os navios estão próximos ao seu destino, é necessário o auxílio de profissionais especializados que possuem os conhecimentos específicos daquela região (relevo submarino, condições do mar e outros), garantindo a segurança e a correta realização da operação. Os especialistas que comandam essa manobra de dentro do navio e ao lado do capitão são conhecidos como práticos. Eles possuem o suporte de embarcações menores, mas de alta potência, conhecidos como rebocadores que seguem os comandos do prático para posicionar e movimentar o navio.

Entre as diversas manobras realizadas pelo prático está a navegação do navio em meio ao canal de acesso. Esta operação tem como objetivo realizar a condução do navio em média velocidade por um canal estreito e em geral dragado que direciona o mesmo até o porto, no qual ele deverá estar em uma baixa velocidade para que os rebocadores possam realizar o posicionamento final até o berço. Existem vários fatores que tornam sua execução complexa, um dos principais fatores é a redução de velocidade que amplifica a dominância dos efeitos ambientais sobre o navio dificultando o seu controle, portanto, soluções de controle automático de trajetória são temas de pesquisa. Os controles baseados em modelagem podem ser custosos de serem desenvolvidos além de ter eficiência apenas em casos muito específicos. Por exemplo, na utilização de linearização, ao simplificar o modelo, além de reduzir os detalhes de seus efeitos, existe a necessidade de trabalho em torno das condições definidas, o que, consequentemente, inviabiliza sua utilização em situações reais.

Como os práticos possuem muita experiência na execução desta manobra, o projeto visa criar um controlador que possa se beneficiar de seu conhecimento e intuição para contornar esses impedimentos. Portanto, a utilização de redes neurais se mostra interessante, pois esta permite conceber uma solução que se adapta e simula os dados de treinamento, logo, poderá ser utilizado uma sequência temporal de estados e comandos em manobras similares realizadas por esses profissionais servindo de insumo para o aprendizado da rede.

De forma a validar o controlador desenvolvido, foram realizadas simulações com o integrador numérico do TPN (Tanque de Provas Numérico) a partir de um módulo de

conexão que permite o envio das condições de estado do simulador para a rede neural e, em resposta, as ordens de comando adequadas para a manobra no sentido oposto, assim, demonstrando se o mesmo está apto a manobrar uma embarcação como os práticos que geraram os dados de treinamento.

Portanto, a rede neural fornece uma forma alternativa de lidar com as complexidades envolvidas, permitindo o desenvolvimento futuro de sistemas de controle autônomo de navegação em áreas abrigadas. Isso pode ter diversas implicações, como o melhor controle da velocidade de avanço pelo controle do propulsor permitindo embarcações de maior porte em portos de canais de acesso mais curtos, otimização de tempo para realização da manobra reduzindo as filas para entrada no porto e aumento da segurança ao reduzir a influência do ser humano.

1.2 Objetivos e requisitos

Sumariamente, o projeto tem como objetivo elaborar uma rede neural treinada com base na experiência de práticos reais, pois isso permite simular a complexa sequência de decisões que são baseadas na experiência deles.

Uma manobra de entrada no canal é definida ter sido realizada com sucesso quando a embarcação permanece dentro dos limites por toda a sua extensão e atinge o seu final com velocidade baixa o suficiente para que a manobra seguinte de atracação não apresente riscos às pessoas, ao porto e às outras embarcações. De forma mais precisa, podem ser definidos os seguintes requisitos que validam a manobra:

- Distância mínima das margens: 0.5 boca em relação a cada margem, sendo a boca máxima o termo que se refere a maior largura da seção transversal da embarcação;
- Velocidade máxima ao final do canal: 5.0 nós (2.5 m/s).

Para que o controle seja viável, a rede neural também deve conseguir ler e interpretar as entradas, por exemplo, a distância com a linha de centro do canal, para assim decidir quais os comandos de leme (direção) e de máquina (rotação do propulsor) adequados para cumprir o objetivo sendo sua validação final realizada ao simular sua performance com o integrador numérico do TPN.

Como extensões do projeto, ou seja, além do objetivo primário, é de interesse generalizar o máximo possível, desta maneira, seriam realizadas novas implementações

e otimizações para cumprir condições extremas que atingiriam os limites dos requisitos supracitados, por exemplo, condições ambientais mais intensas que os presentes nos dados de treinamento, canais de acesso mais curtos para embarcações de mesmo porte, canais de mesma extensão para embarcações de maior porte e outros testes. Essa extensão seria relevante pois tornaria a rede neural mais consistente e segura além de permitir averiguar critérios e normas de segurança ou mesmo realizar o planejamento de canais para novos projetos de portos.

1.3 Organização do texto

O presente trabalho está elaborado de forma a compreender as pesquisas e as etapas realizadas no desenvolvimento do projeto de forma que facilite o embasamento do leitor na temática do trabalho e permita compreender as decisões dos autores até os resultados obtidos.

No capítulo 2 é apresentado o estado da arte demonstrando o desenvolvimento da área de automação por diversas frentes até as abordagens das metodologias utilizadas nesse projeto e pesquisas semelhantes que enfatizam a relevância na utilização de redes neurais na área de manobra de embarcações em canais de acesso.

No capítulo 3 é abordado o conteúdo teórico para se compreender o funcionamento de uma rede neural, desde formas de aprendizado e sua composição até os processos de propagação e retropropagação que são os algoritmos base para gerar o resultado e adequar os pesos e vieses da rede neural, respectivamente. Também são introduzidos alguns modelos de redes neurais recorrentes utilizados no projeto.

No capítulo 4 é apresentado a forma em que o problema será resolvido, ou seja, são ressaltadas as técnicas implementadas e quais as ferramentas auxiliares que são utilizadas em cada etapa.

No capítulo 5 são abordados os conceitos e propriedades fundamentais que influenciam na simulação da embarcação e a forma como foram processados os dados de treinamento a partir do banco de simulações realizadas por práticos no TPN. No processamento, as etapas são descritas por meio de um exemplo para tornar a explicação mais didático e realista.

No capítulo 6 é apresentado a elaboração da rede neural focando na arquitetura da

mesma e relacionando como o embasamento teórico do capítulo 3 foi utilizado no projeto.

No capítulo 7 é abordado o processo de elaboração da validação da rede neural por meio de uma integração entre a solução desenvolvida e o integrador numérico que permite simular as condições de navegação.

No capítulo 8 são apresentados os resultados obtidos com as redes neurais desenvolvidas durante este projeto por meio da descrição das etapas e dificuldades verificadas no decorrer do trabalho.

No capítulo 9 é feita uma análise dos resultados obtidos com as possíveis explicações do comportamento observado e verificação dos requisitos levantados. Por fim, são feitas sugestões de projetos para continuidades da pesquisa.

No capítulo 10 o trabalho é finalizado ao citar as conclusões obtidas.

2 Estado da arte

O desenvolvimento da engenharia sempre esteve muito relacionado aos problemas da humanidade. Em cada período, novas tecnologias permitiram o avanço do conhecimento científico e trouxeram benefícios a sociedade. Embora muitos temas sejam estudados em paralelo nas grandes universidades, alguns ganham um destaque maior, seja por se acreditar que o futuro caminha nessa direção ou apenas por questões midiáticas tornando a linha de conhecimento mais "popular". Nos dias de hoje muito se fala de automação, seja de fábricas, processos, robôs e outros, mas especialmente para o contexto desse trabalho: veículos de transporte. Por anos os carros autônomos foram um sonho da humanidade, hoje há grandes avanços nessa frente, por exemplo, (PADEN et al., 2016) demonstram como o problema dos carros autônomos na verdade é uma série de subproblemas que podem ser resolvidos por diversas técnicas em conjunto, por exemplo, o primeiro problema é a definição da rota para atingir um determinado destino, essa necessidade é resolvida na camada de "roteamento" em que seu resultado serve como entrada para a camada "comportamental" que define como interagir com os outros veículos e assim por diante até se obter um veículo completamente autônomo; (ALTHOFF; MERGEL, 2011) realizam o estudo da adequação de técnicas de probabilidade, cadeia de Markov e Monte Carlo, para avaliação de segurança em carros autônomos, mais especificamente nos casos de obstáculos na trajetória e riscos de colisão; entretanto, não apenas de tecnologia e métodos matemáticos essa área é composta, questões éticas também são envolvidas como consta no artigo de (LIN, 2016) em que cita diversas situações para reflexão sobre como os carros autônomos devem reagir.

As tecnologias de automação são mais populares nos veículos terrestres devido a proximidade com o dia a dia das pessoas, mas isso não significa que essas inovações estão restritas a este âmbito. Pelo contrário, o desenvolvimento ocorre em outros meios, por exemplo, veículos marítimos. Embora as condições de controle sejam muito diferentes nos dois casos, por exemplo, as inércias envolvidas, o meio em que o veículo está imerso, as normas regulamentativas para segurança e outros, muitas técnicas podem ser reaproveitadas e inclusive algumas motivações são compartilhadas. Nos dois casos, o risco de colisão é um problema vital, a automação pode ser um caminho promissor para solucionar ou, pelo menos, minimizar isso, já que 75% a 96% dos acidentes possui influência de erro humano

segundo (PERERA; CARVALHO; SOARES, 2009).

No âmbito naval, o risco de colisões ganhou muita importância devido a crescente frota de navios. Uma forma de evitar essas fatalidades é a regulamentação da navegação. A IMO (*International Maritime Organization*) elaborou um conjunto de normatizações conhecidos como COLREGS (*Collision Regulations*) que determina situações e manobras a serem realizadas para evitar colisões. (BELCHER, 2002) realiza um estudo sobre as COLREGS em que conclui que uma regulamentação não é suficiente para evitar riscos, seja pela questão subjetiva já que depende da interpretação momentânea de quem está a bordo, mas seja também pela impossibilidade de registrar e normatizar todas as situações adversas possíveis.

Assim, as soluções tecnológicas de auxílio a tripulação se tornam ferramentas cruciais que reduzem a subjetividade do homem na observação das condições momentâneas para a tomada de decisões. (STATHEROS; HOWELLS; MAIER, 2008) citam algumas tecnologias de navegação entre elas o GPS (*Global Positioning System*), o Radar, o ARPA (*Automatic Radar Plotting Aid*) e instrumentos de monitoração das condições atmosféricas e da água. Esses instrumentos apenas fornecem dados mais precisos e objetivos a tripulação, por isso não são suficientes para evitar colisão, já que a interpretação e tomada de decisão ainda depende do homem. Tecnologias auxiliares que realizam parte da interpretação desses dados são muito úteis, por exemplo, (SATO; ISHII, 1998) desenvolveram uma análise utilizando imagens por infravermelho para prever rotas de outros navios por meio de informações adicionais ao Radar como tamanho, tipo e alterações aparentes no aspecto da embarcação, dados esses que seriam identificados pela tripulação estando sujeitos a subjetividade.

De forma a restringir a influência humana na interpretação e decisão das ações, uma solução adicional seria a elaboração de pilotos automáticos. Assim, é necessário a existência de uma malha de controle que recebe os dados dos instrumentos anteriormente citados, interpreta a situação atual e define os comandos adequados para atingir um dado objetivo. (FOSSEN, 2000) realiza um estudo do desenvolvimento da teoria de controle não linear baseado em modelos para sistemas marítimos que permitiu obter sistemas de equilíbrio dinâmico, seguidores de trajetória e controle de sistemas subatuados por meio de diversas tecnologias desde controladores PID (*proportional–integral–derivative*) até pilotos automáticos mais avançados com uso de LQG (*Linear–quadratic–Gaussian*) e controle por $\mathcal{H}_\infty$. Uma das grandes dificuldades ao se utilizar esses controladores é a consideração das

não linearidades envolvidas que perturbam o sistema como as condições ambientais, nesse sentido, (TANNURI et al., 2010) utilizam uma técnica alternativa baseada em controle não linear robusto por modos deslizantes, apresentando robustez e facilidade de ajuste.

Não restrito a apenas aplicações marítimas, desenvolvimentos de tecnologias de outras áreas também podem ser reaproveitadas. Além dos carros autônomos, uma segunda área muito influente atualmente e correlata a primeira é a da inteligência artificial. (MAKRIDAKIS, 2017) aborda o desenvolvimento desse campo como uma nova revolução assim como foi a revolução industrial e a revolução digital impactando a sociedade toda. Um exemplo que demonstra a influência nos dias de hoje é o fato da Google em 2012 ter apenas 2 projetos de *deep learning* enquanto em 2017 mais de 1000 projetos estavam em andamento. A aplicabilidade desses conceitos é muito ampla, por exemplo: em neurociência e visão computacional, (KRUTHIVENTI; AYUSH; BABU, 2017) utilizam redes neurais convolucionais para identificar o padrão atuante no mecanismo de atenção visual do ser humano; em análise de históricos financeiros, (WAN; SI, 2017) utilizam ANFIS (*adaptive neuro fuzzy inference system*) que define padrões e tendências da flutuação dos dados; em medicina, (BEHESHTI; DEMIREL; MATSUDA, 2017) utilizam algoritmos genéticos para análise de imagens de ressonância magnética para identificação de tendências de Alzheimer; em geotecnia, (SHAHIN, 2016) explora a área por diversas técnicas (redes neurais, algoritmos genéticos e regressão polinomial evolutiva), segundo o autor existem problemas muito complexos e não muito bem compreendidos nessa área o que torna os modelos imprecisos ou muito simplificados, sendo a inteligência artificial uma alternativa pois esta aprende baseada nos dados reais e completos podendo ser refinado por novos treinamentos.

Da mesma maneira que os exemplos anteriores demonstraram a versatilidade das técnicas de inteligência artificial para resolução de problemas complexos, a mesma pode ser utilizada no contexto marítimo, especialmente para resolver problemas de colisão e condução automática. (ROBERTS et al., 2003) apresenta a evolução dos pilotos automáticos marítimos em que o modelo original utilizava controladores PID e evoluiu para controladores inteligentes com a utilização de lógica fuzzy e redes neurais como forma de simular as complexas tomadas de decisões dos timoneiros. (LEE; KIM, 2004) e (LEE; KWON; JOH, 2004) apresentam utilizações de lógica fuzzy em condução de veículos marítimos de forma a evitar situações de colisão respeitando normas da COLREGS, demonstrando ser uma alternativa tanto para reelaboração de trajetórias em tempo real quanto garantir que os

desvios de trajetória sejam realizados dentro de uma zona segura e efetivos contra objetos estáticos e em movimento. (AHMED; HASEGAWA, 2013) estudam o complexo movimento de atracação de embarcações que sofre muita influência dos efeitos ambientais especialmente em velocidades baixas, para solucionar o problema os autores estudaram o uso de redes neurais associadas a um controlador PD de forma a simular as ações realizadas pelos homens, o controlador inteligente apresentou resultados satisfatórios mesmo em situações diferentes dos dados de treinamento. (IM; NGUYEN, 2017) realizam o treinamento de uma rede neural com dados de atracação em um determinado porto e realizam os testes de verificação da solução em condições de um outro porto obtendo resultados limitados, mas que demonstram a versatilidade que o controle inteligente possui.

Existe um ponto de semelhança entre os vários artigos citados e até mesmo outros artigos de controle em geral que é a utilização de simulações como prova de conceito. Especialmente na área naval, os objetos de estudo são difíceis de serem estudados em modelos reais, seja pelo seu tamanho, custo, disponibilidade e outros fatores, sendo assim, o uso de simulações é um ponto crucial na realização das pesquisas. (SOUZA JR. et al., 2009) apresentam duas simulações computacionais de manobras em canais brasileiros, desde a concepção dos modelos até a entrada de comandos para análise de respostas. Projetos de simuladores completos como esses ou até mais simples permitem realizar análises detalhadas, rápidas e versáteis.

3 Abordagem teórica

3.1 Variância e Correlação

Uma etapa importante para o desenvolvimento das redes neurais envolve a análise de correlação dos dados, com ela podemos avaliar a qualidade da base de dados. Para tanto é preciso compreender os conceitos de variância e correlação. A variância mede a dispersão de uma distribuição em torno da média (BARBER, 2012):

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n} \tag{1}$$

Onde x_i representa cada elemento da distribuição, μ é a média dessa distribuição e n é a quantidade de elementos.

A covariância é a propriedade que mostra o nível de relação entre duas variáveis X e Y aleatórias não independentes (DEVORE, 2012).

$$Cov(X,Y) = \sum_{x}\sum_{y}(x - \mu_x)(y - \mu_y)p(x,y) \tag{2}$$

Sendo $p(x,y)$ a função de probabilidade conjunta das variáveis X e Y.

Contudo, os valores obtidos pela covariância apresentam uma falha pelo fato de dependerem criticamente das unidades de medida (DEVORE, 2012). O coeficiente de correlação visa resolver este problema e é definido como:

$$Corr(X,Y) = \rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{3}$$

Em que σ_X e σ_Y é o desvio padrão das variáveis X e Y, respectivamente.

Tal definição limita o valor de correlação entre $-1 \leq Corr(X,Y) \leq 1$, sendo possível analisar a "força" da relação entre as variáveis independentemente de sua unidade. Quanto maior o módulo de seu valor, mais forte é a correlação entre as variáveis. O sinal positivo indica que as duas variáveis possuem a mesma tendência de comportamento e o sinal negativo representa o oposto.

3.2 Aprendizagem

Antes de explicar os princípios matemáticos que regem uma rede neural é interessante compreender uma das etapas fundamentais para seu funcionamento, a aprendizagem. Assim como um ser humano, a percepção e tomada de decisões decorre da sua exposição a fatos que serão internalizados e compreendidos. Em (RUSSELL; NORVIG, 2004) são apresentadas três divisões básicas para aprendizagem segundo a realimentação utilizada:

- **Aprendizado supervisionado**: ocorre aprendizado por meio de um mapeamento entrada-saída previamente definido, ou seja, os dados de entrada estão classificados com as saídas esperadas de forma que os seus parâmetros serão adequados para que proporcionem uma inferência similar;
- **Aprendizado não-supervisionado**: ocorre aprendizado sem mapeamento entrada-saída previamente definido, ou seja, o agente receberá apenas as entradas e deverá deduzir os padrões presentes nelas por conta própria;
- **Aprendizado por reforço**: também não apresenta um mapeamento entrada-saída, mas o aprendizado é baseado em recompensas, por meio de ações corretas serão retribuídas as recompensas para que o agente compreenda quando acertou, assim, iterativamente, converge ao comportamento adequado.

Desta maneira, como o objetivo do projeto é transferir a experiência dos práticos para a rede neural, será utilizado um aprendizado supervisionado em que as entradas são os parâmetros que definem o estado da embarcação em dado instante de tempo e as saídas serão os comandos selecionados pelo profissional. Portanto, o restante deste capítulo será focado nesta forma de aprendizado.

3.3 Redes neurais

A unidade básica do cérebro é o neurônio, célula que se interliga com outras de forma que em conjunto possam transmitir e processar os sinais elétricos. Analogamente, de forma que se possa reproduzir sua capacidade funcional, foram elaboradas as redes neurais artificiais que são compostas por unidades de neurônios artificiais interligados representados por modelos matemáticos.

A teoria a seguir está baseada em (NG, 2018) e também será o padrão de notação

utilizado. Cada unidade pode ser compreendida como uma função matemática de forma que recebe entradas e produz uma saída. Uma arquitetura de conexão entre essas unidades está representada na figura 1, no lado esquerdo estão representados os parâmetros de entrada que mais genericamente seriam dados por $x_j^{(i)}$, em que o subscrito representa o parâmetro de uma entrada e o sobrescrito entre parênteses o índice da entrada, por exemplo, no caso deste projeto, $x_1^{(1)}$ poderia ser a distância à linha de centro da simulação 1, $x_2^{(1)}$ poderia ser a distância ao final do canal da simulação 1, $x_1^{(2)}$ poderia ser a distância à linha de centro da simulação 2 e $x_2^{(2)}$ poderia ser a distância ao final do canal da simulação 2. Ao lado direito estão representadas as camadas ocultas, poderiam existir muitas outras camadas e cada uma com muitas outras unidades, neste caso, estão representadas 2 camadas com 4 e 3 unidades, respectivamente, da esquerda para direita. Finalmente, na extrema direita está a camada de saída em que $\hat{y}$ representa o valor inferido pela rede neural para a entrada (x_1, x_2, x_3). Embora tenha sido nomeado a entrada como uma camada, geralmente esta não é considerada nas numerações, portanto, considera-se que a arquitetura da figura possui apenas 3 camadas sendo 2 ocultas e 1 de saída. Em cada unidade está representado o resultado da ativação $a_j^{[i]}$ em que o subscrito representa o índice da unidade em uma camada e o sobrescrito entre colchetes o índice da camada.

Cada unidade da rede neural realiza os dois processos representados no diagrama da figura 2. Para um vetor de entrada $x \in \mathbb{R}^{n_x}$ com n_x sendo o número de parâmetros, a primeira unidade da camada 1 realiza uma ponderação linear por:

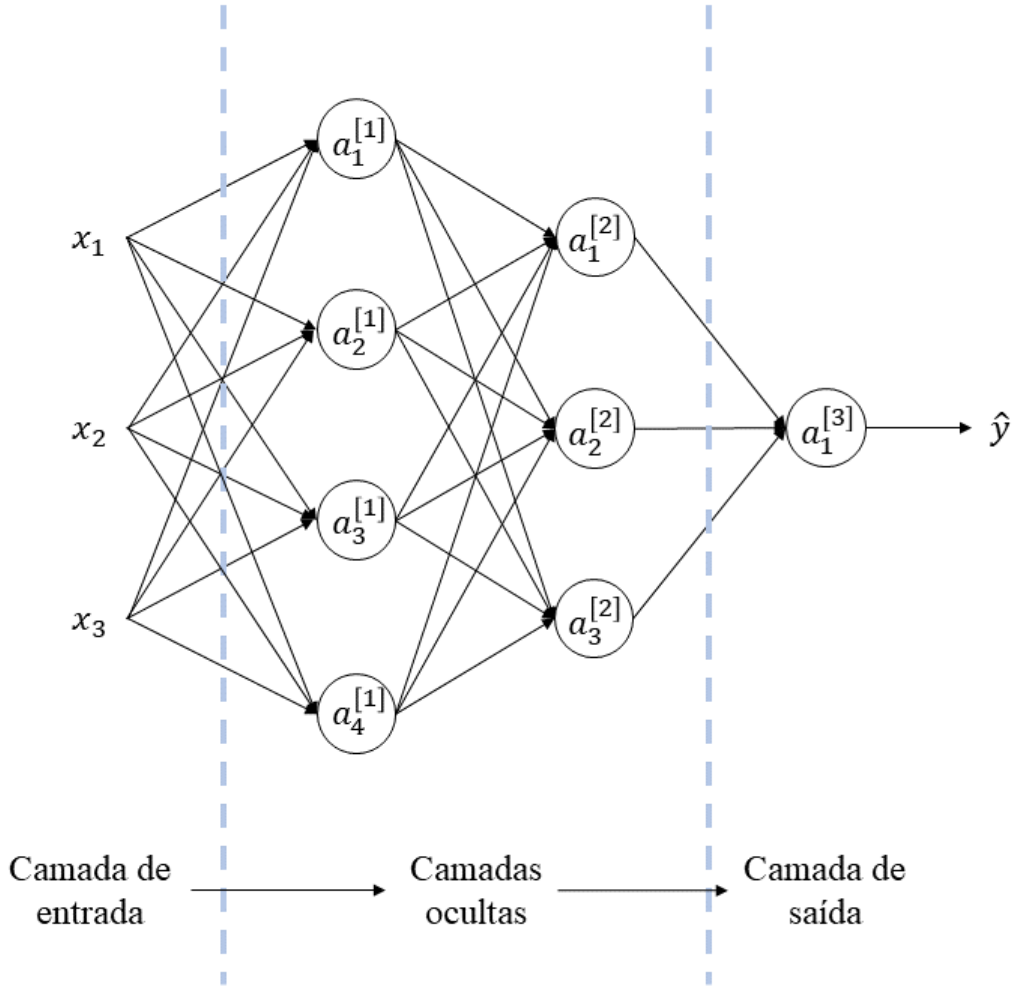
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]} \tag{4}$$

Em que $w_1^{[1]T}$ é o vetor de pesos transposto de $w_1^{[1]} \in \mathbb{R}^{n_x}$ e $b_1^{[1]}$ um valor de viés tal que $b_1^{[1]} \in \mathbb{R}$. Em seguida é utilizada uma função não linear $g : \mathbb{R} \to \mathbb{R}$ resultando no valor da ativação que é a sua própria saída:

$$\hat{y}_1^{[1]} = a_1^{[1]} = g(z_1^{[1]}) \tag{5}$$

A função $g(\cdot)$ é necessária pois, em sua ausência, todas as unidades gerariam resultados lineares, tornando o conjunto de todos os resultados da rede também linear, o que faria com que todas as unidades e camadas extras não proporcionassem efeitos adicionais, sendo assim, redes mais simples conseguiriam obter o mesmo resultado com menos processamento.

Figura 1 – Exemplo de arquitetura de rede neural de 3 parâmetros de entrada e 3 camadas sendo duas ocultas e uma saída



Fonte: Autores, baseado em (NG, 2018)

Alguns exemplos para essa função seriam a sigmóide, definida como:

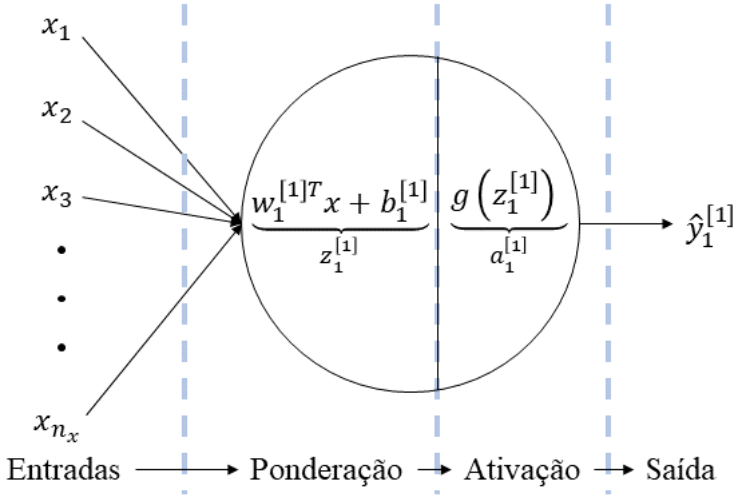
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

Ou a ReLU (*Rectified Linear Unit*), definida como:

$$g(z) = max(0, z) \tag{7}$$

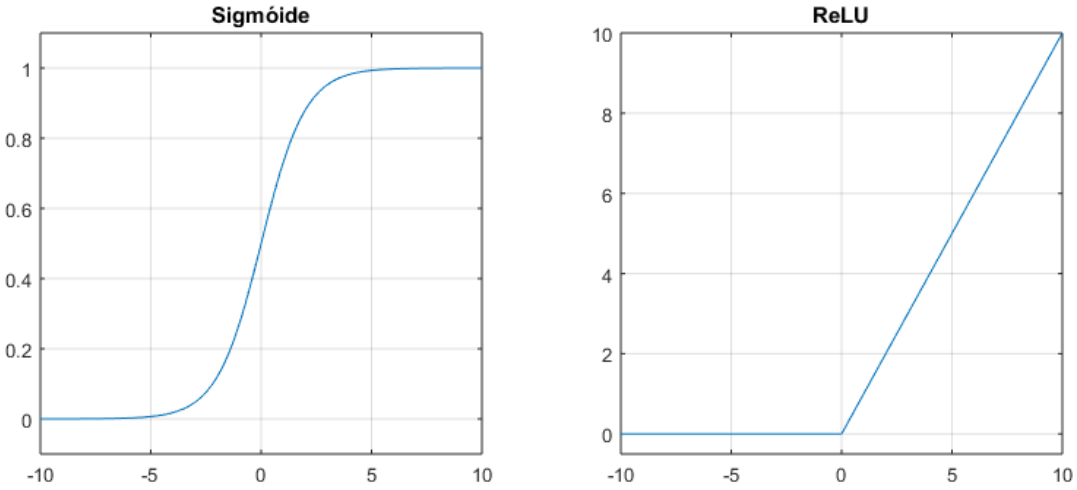
A escolha da função mais adequada depende do propósito da rede neural, por exemplo, em um classificador binário a saída deve ser binária (0 ou 1) então a sigmóide seria uma opção adequada. Mas sua derivada tende a 0 conforme $|z|$ aumenta, o que pode tornar a convergência mais lenta, por esta razão, a função ReLU é muito utilizada nas camadas ocultas, propriedades observáveis na figura 3.

Figura 2 – Diagrama de duas etapas representando a primeira unidade da rede neural na camada 1



Fonte: Autores, baseado em (NG, 2018)

Figura 3 – Plotagem da sigmóide, equação 6, e da ReLU, equação 7



Fonte: Autores

Pode-se generalizar o processo para toda a rede neural de forma a otimizar o processamento utilizando vetorização durante a implementação em cada camada. Considerando que os dados de treinamento possuem m entradas de n_x parâmetros cada, define-se a matriz $X \in \mathbb{R}^{n_x \times m}$ das entradas. Em uma camada l com $n^{[l]}$ unidades, define-se a matriz de pesos $W^{[l]} \in \mathbb{R}^{n^{[l-1]} \times n^{[l]}}$ e vetor de viés $b^{[l]} \in \mathbb{R}^{n^{[l]}}$ que pode ser expandido para $B^{[l]} \in \mathbb{R}^{n^{[l]} \times m}$ ao replicar seus valores já que todas as m entradas utilizam o mesmo viés. Pela aplicação da função de ativação $g^{[l]} : \mathbb{R}^{n^{[l]} \times m} \to \mathbb{R}^{n^{[l]} \times m}$ haverá a saída $A^{[l]} \in \mathbb{R}^{n^{[l]} \times m}$ e para simplificar

a notação, pode-se considerar $A^{[0]} = X$ e $n^{[0]} = n_x$, logo, para cada camada l:

$$\begin{aligned} Z^{[l]} &= W^{[l]T} A^{[l-1]} + B^{[l]} \\ A^{[l]} &= g^{[l]}(Z^{[l]}) \end{aligned} \tag{8}$$

Esse processo definido pelas equações em 8 é conhecido como propagação, ou seja, é a inferência que a rede neural realiza a partir das entradas.

3.4 Treinamento

De posse do mapeamento entrada-saída, o intuito de uma rede neural é inferir uma saída $\hat{y}$ que aproxime da saída mapeada y, ou seja, para cada entrada i se espera $\hat{y}^{(i)} \approx y^{(i)}$. Para simplificar a explicação, primeiro será considerado uma única unidade definida pelo peso w e viés b. De forma a mensurar a sua performance, pode-se definir uma função de erro, por exemplo, utilizando o erro quadrático dado por:

$$\mathscr{L}(\hat{y}, y) = (\hat{y} - y)^2 \tag{9}$$

Desta forma, pode ser definida a sua função de custo que mensura a performance em todo o conjunto de m entradas de treinamento:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}(\hat{y}^{(i)}, y^{(i)}) \tag{10}$$

Do qual se obtém o erro quadrático médio (*mean squared error - MSE*):

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 \tag{11}$$

Portanto, os resultados da unidade podem ser melhorados ao reduzir o valor da função de custo, pois isso significa que os valores inferidos estão próximos aos valores esperados. Em outras palavras, o objetivo é calibrar os valores do peso e do viés da unidade de forma que a função custo se aproxime do seu valor mínimo. Para encontrar esses valores adequados, utiliza-se um método chamado gradiente descendente, um método iterativo que por meio da derivada define a direção no espaço a ser seguido, ou seja:

$$\begin{aligned} w &:= w - \alpha \frac{\partial J(w, b)}{\partial w} \\ b &:= b - \alpha \frac{\partial J(w, b)}{\partial b} \end{aligned} \tag{12}$$

Em que o termo α é conhecido como a taxa de aprendizado, ou seja, o tamanho do passo a ser avançado na direção de decrescimento indicado pela derivada.

Em uma rede neural com L camadas, isso poderia ser generalizado para uma função custo $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, ..., W^{[L]}, b^{[L]})$ análoga:

$$J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, ..., W^{[L]}, b^{[L]}) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}(\hat{y}^{(i)}, y^{(i)}) \tag{13}$$

Portanto, para cada camada l é realizada a operação:

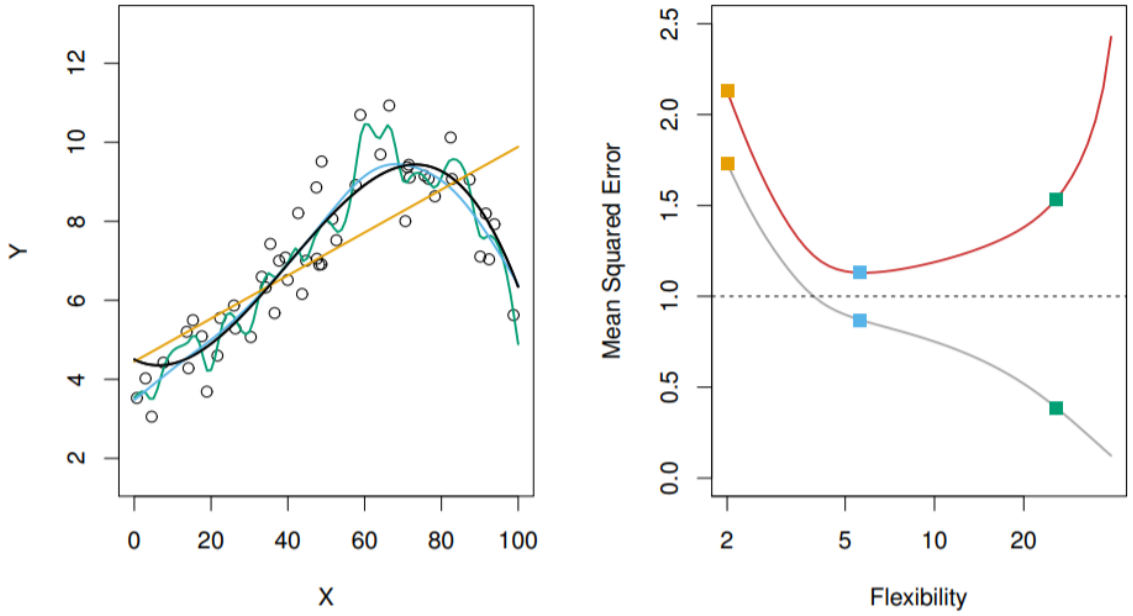
$$\begin{aligned} W^{[l]} &:= W^{[l]} - \alpha \frac{\partial J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, ..., W^{[L]}, b^{[L]})}{\partial W^{[l]}} \\ b^{[l]} &:= b^{[l]} - \alpha \frac{\partial J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, ..., W^{[L]}, b^{[L]})}{\partial b^{[l]}} \end{aligned} \tag{14}$$

Sendo o processo definido pelas equações em 14 conhecido como retropropagação, ou seja, o treinamento da rede neural a partir da minimização da função custo que significa uma aderência dos resultados inferidos aos valores conhecidos. Uma iteração completa, ou seja, a realização de uma propagação e uma retropropagação é conhecida como *epoch*.

O processo de treinamento pode gerar um problema importante conhecido como *overfitting* em que a rede neural é treinada em excesso e inclui em seu aprendizado os ruídos intrínsecos dos dados de acordo com a flexibilidade permitida ao modelo. A figura 4 ilustra o efeito de modelos que aderiram pouco e muito aos dados, nota-se que a curva do custo na base de treinamento é decrescente conforme a flexibilidade do modelo, mas a curva do custo na base de teste não, caracterizando na região ascendente a ocorrência de *overfitting*, ou seja, o modelo faz previsões muito boas para a base de treinamento, mas não em uma base diferente.

Existem métodos para evitar a ocorrência deste problema, um deles é conhecido como *dropout* em que unidades da rede são eliminadas aleatoriamente durante o treinamento de forma a reduzir o excesso de aderência do modelo.

Figura 4 – Do lado esquerdo estão representados os modelos treinados para aderir aos dados gerados a partir da curva preta com ruídos, nota-se que a curva amarela tem pouca flexibilidade e ocorrendo o oposto para a curva verde. Do lado direito estão representadas as curvas de custo na base de treinamento (cinza) e na base de teste (vermelho) com os pontos coloridos indicando os respectivos erros em relação aos modelos da esquerda, demonstrando o *overfitting* do modelo verde.



Fonte: (JAMES et al., 2014)

3.5 Redes neurais recorrentes

As redes neurais na estrutura apresentada até o momento são conhecidas como redes neurais sem realimentação, são redes mais simples que podem obter resultados muito bons em diversas aplicações, mas nem tanto em outras. Uma das desvantagens que podem ser observadas está em seu próprio nome: sem realimentação, ou seja, as entradas estão individualizadas e isoladas, uma não afeta o resultado da outra. Desta maneira, situações em que os dados de treinamento possuem uma ordem lógica não são detectados. Por exemplo, se o objetivo fosse elaborar um rede neural que detecte nomes, na frase "João estuda mecatrônica", cada palavra seria aprendida individualmente, mas a relação entre elas devido a ordem não seria detectada, logo, a informação implícita de que quem (João) realiza uma ação (estuda mecatrônica) possui alta chance de ser um nome, não seria considerada pela rede neural.

Esse problema é recorrente e relevante na área de aprendizado de máquinas, por exemplo, a detecção de sequências é importante em textos, em música, em sons e outros.

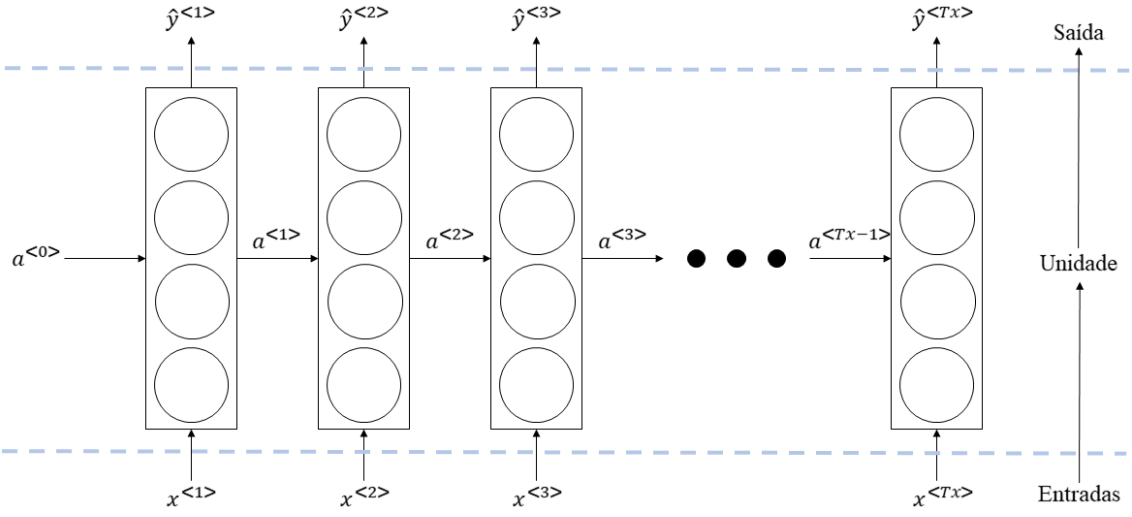
Para abordar essa questão, foram desenvolvidas as redes neurais recorrentes. De forma a explicar seu funcionamento, será necessário incrementar a notação já utilizada: pode-se definir $x^{<k>}$ como a k-ésima parte da sequência de x, por exemplo, sendo a frase anterior o valor de x então $x^{<1>}$ será "João", $x^{<2>}$ será "estuda" e $x^{<3>}$ será "mecatrônica".

A figura 5 apresenta um modelo esquemático do funcionamento de uma rede neural recorrente. Assim como na notação anterior, mas acrescido do sobrescrito entre os sinais de $<$ e $>$, x representa a entrada, a representa o resultado da função de ativação, y representa a saída ou o valor previsto para cada entrada e T_x representa o número de subpartes na entrada x. Algebricamente essa estrutura pode ser representada como:

$$\begin{aligned} a^{<k>} &= g(w_{aa}a^{<k-1>} + w_{ax}x^{<k>} + b_a) \\ \hat{y}^{<k>} &= g(w_{ya}a^{<k>} + b_y) \end{aligned} \tag{15}$$

Em que a notação w_{uv} indica o peso para qual função u está sendo calculada e sobre qual parâmetro v de entrada da camada, admitindo os valores a, x ou y. Por exemplo, w_{ax} é o peso para a entrada $x^{<k>}$ no cálculo da função de ativação a ser transmitido para a camada $k+1$. A notação b_u tem representação análoga, mas para o viés.

Figura 5 – Exemplo esquemático do funcionamento de uma rede neural recorrente



Fonte: Autores, baseado em (NG, 2018)

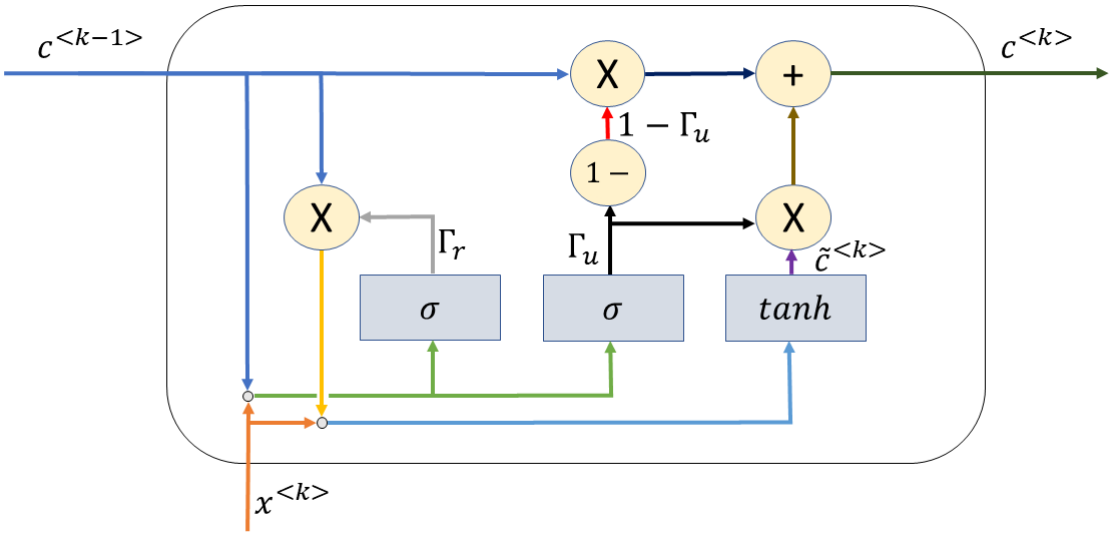
Detalhando seu funcionamento, a figura 5 está representando uma única camada da rede neural recorrente em que as entrada $x^{<k>}$ entram individualmente e são processadas gerando dois tipos de saída: $\hat{y}^{<k>}$ e $a^{<k>}$ sendo a primeira a saída prevista pela rede análogo ao da rede neural sem realimentação e a segunda é uma saída de comunicação

entre as entradas $< k >$, ou seja, é a realimentação de uma entrada anterior que será considerada ao realizar o novo processamento.

Desta maneira para realizar a definição e treinamento da rede como um todo podem ser utilizados os mesmos princípios já abordados como a escolha da taxa de aprendizagem, número de unidades, quantidade de camadas, função de custo, retropropagação e assim por diante.

O modelo apresentado é uma ideia geral de como funciona a retroalimentação, mas note que a informação principal advém da entrada anterior e não do conjunto todo, por essa razão outros modelo de unidade para redes neurais recorrentes foram desenvolvidos de forma a implementar uma espécie de "memória" que possa armazenar informações relevantes de mais entradas, por exemplo, a GRU (*Gated Recurrent Unit*) esquematizada na figura 6.

Figura 6 – Modelo esquemático de uma unidade do tipo GRU



Fonte: Autores, baseado em (NG, 2018) e (CHANGHAU, 2017)

Este tipo de unidade pode ser descrita segundo as equações em 16:

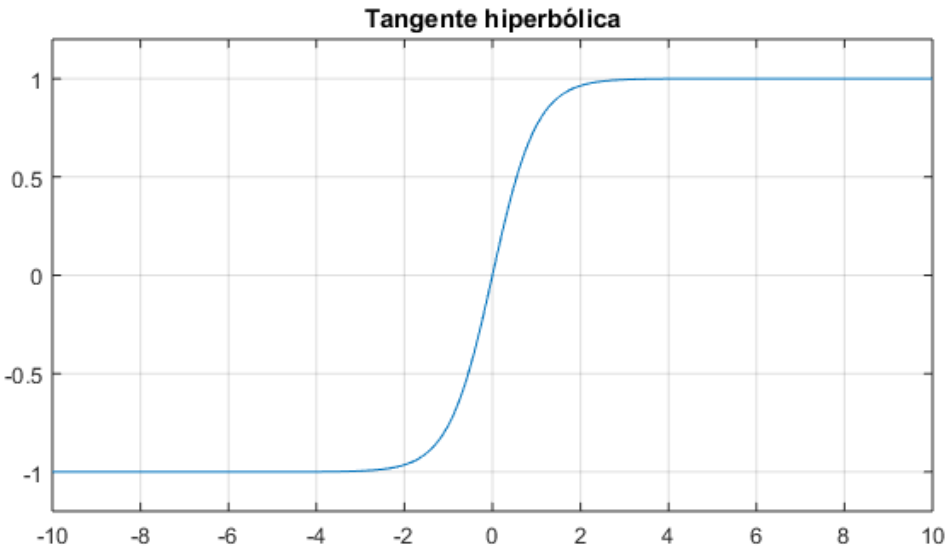
$$
\begin{aligned}
\widetilde{c}^{<k>} &= tanh(w_{cc}\Gamma_r c^{<k-1>} + w_{cx}x^{<k>} + b_c) \\
\Gamma_u &= \sigma(w_{uc}c^{<k-1>} + w_{ux}x^{<k>} + b_u) \\
\Gamma_r &= \sigma(w_{rc}c^{<k-1>} + w_{rx}x^{<k>} + b_r) \\
c^{<k>} &= \Gamma_u\widetilde{c}^{<k>} + (1-\Gamma_u)c^{<k-1>} \\
a^{<k>} &= c^{<k>}
\end{aligned}
\tag{16}
$$

A notação segue o mesmo padrão anterior acrescido de novas variáveis. A letra c é uma abreviação para *memory cell* ou célula de memória sendo sua função armazenar o estado de entradas passadas. Com base na memória passada $c^{<k-1>}$ e a nova entrada $x^{<k>}$ será realizado o cálculo para definir o possível novo valor de $c^{<k>}$ definido como $\widetilde{c}^{<k>}$. Esse novo valor pode ou não ser a saída $c^{<k>}$, isso dependerá de Γ_u. A variável Γ é o *gate* ou portão que designa o nome da célula (**G**RU). Existem dois tipos de *gates*, o Γ_u, u de *update*, responsável por definir se $c^{<k>}$ será igual a $c^{<k-1>}$ ou $\widetilde{c}^{<k>}$ e o Γ_r, r de *relevance*, responsável por definir qual a relevância de $c^{<k-1>}$ no cálculo de $\widetilde{c}^{<k>}$.

A função $\sigma(\cdot)$ representa a sigmóide como descrita na equação 6 e $tanh(\cdot)$ representa a tangente hiperbólica dada por:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{17}$$

Figura 7 – Plotagem da tangente hiperbólica, equação 17

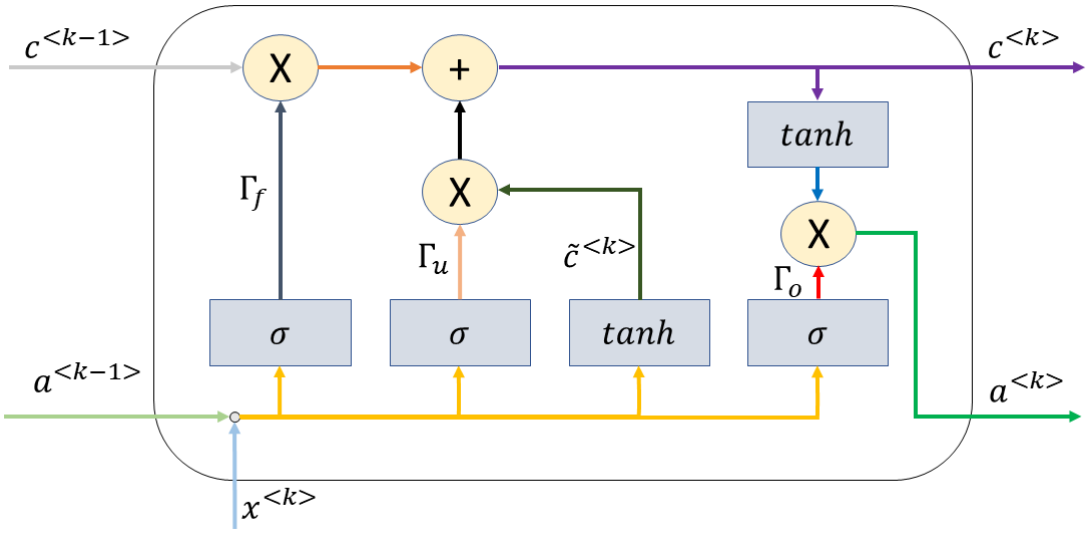


Fonte: Autores

Nota-se que a escolha de $\sigma(\cdot)$ para o cálculo de Γ_r se deve ao fato da mesma variar entre 0 e 1 representando um percentual de $c^{<k-1>}$ no novo valor de $\widetilde{c}^{<k>}$. Enquanto a $tanh(\cdot)$ varia de -1 a 1 e possui uma estreita zona de transição como pode ser observado na figura 7, desta maneira, na maioria dos casos a função retorna um valor próximo de -1 ou de 1 selecionando $c^{<k-1>}$ ou $\widetilde{c}^{<k>}$.

Outro modelo de unidade utilizado para redes neurais recorrentes é o LSTM (*Long Short-Term Memory*), semelhante ao modelo GRU, mas com *gates* adicionais como pode ser observado na figura 8. Suas equações estão em 18.

Figura 8 – Modelo esquemático de uma unidade do tipo LSTM



Fonte: Autores, baseado em (NG, 2018)

$$
\begin{aligned}
\tilde{c}^{<k>} &= tanh(w_{ca}a^{<k-1>} + w_{cx}x^{<k>} + b_c) \\
\Gamma_u &= \sigma(w_{ua}a^{<k-1>} + w_{ux}x^{<k>} + b_u) \\
\Gamma_f &= \sigma(w_{fa}a^{<k-1>} + w_{fx}x^{<k>} + b_f) \\
\Gamma_o &= \sigma(w_{oa}a^{<k-1>} + w_{ox}x^{<k>} + b_o) \\
c^{<k>} &= \Gamma_u\tilde{c}^{<k>} + \Gamma_f c^{<k-1>} \\
a^{<k>} &= \Gamma_o tanh(c^{<k>})
\end{aligned}
\tag{18}
$$

Percebe-se que existe uma semelhança entre as duas unidades, mas esta é mais complexa ao apresentar um *gate* Γ_u análogo ao anterior, mas também um *gate* Γ_f, f de *forget* ou esquecer, portanto, manter ou não o antigo valor $c^{<k-1>}$ depende de dois *gates* nesta célula ao invés de apenas Γ_u como na célula GRU. Adicionalmente, existe um Γ_o, o de *output* ou saída, que pondera a saída da ativação segundo o valor de $c^{<k>}$.

Não existe um consenso no meio acadêmico entre o uso de GRU e LSTM, em geral o LSTM é mais utilizado, mas observa-se pelo descrito que o GRU é mais simples, logo, pode apresentar menor complexidade e melhor performance em determinados casos.

4 Metodologia e ferramentas

A metodologia base para a solução do problema apresentado é a utilização de redes neurais, pois esta não requer a modelagem das não linearidades e pode ser utilizada como um análogo das tomadas de decisões dos práticos além de já ter apresentado resultados promissores como abordado no capítulo 2.

Como a intenção é utilizar os dados de simulações passadas, a forma mais adequada é a utilização de um aprendizado supervisionado como descrito em mais detalhes na seção 3.2. De forma a gerar os dados de treinamento, como detalhado na seção 5.4, foi necessária a utilização da linguagem *Python*, esta que também será utilizada no desenvolvimento da rede neural.

A linguagem *Python* foi uma escolha baseada em diversas premissas que entre elas podem ser citadas:

- Os autores já possuem experiência com a linguagem e *softwares* auxiliares de forma a facilitar a execução do projeto. Entre esses *softwares* está a *IDE* (*Integrated Development Environment*) denominada *PyCharm* da empresa *JetBrains* que auxilia o desenvolvimento do código fornecendo funções adicionais especialmente para realizar o *debug* do código;
- A linguagem também possui bibliotecas úteis como o *NumPy* que permite otimizar o código por vetorização ou o *Pandas* que facilita a manipulação dos dados de treinamento;
- É uma linguagem de programação muito utilizada por desenvolvedores de redes neurais portanto é possível encontrar uma comunidade muito ativa, facilitando a busca por soluções aos problemas deparados;
- Existem muitos *frameworks* de redes neurais que são compatíveis com a linguagem permitindo maior flexibilidade na seleção da mesma, por exemplo, *TensorFlow* da *Google Brain*, *PyTorch* hoje desenvolvido pelo *Facebook*, *Keras* de *François Chollet* entre outros.

Para o controle das versões foi utilizado o *GitHub*, serviço escolhido principalmente pela familiaridade dos autores e servindo aos propósitos básicos necessários (facilidade de uso, controle, compatibilidade com *PyCharm* e gratuidade).

O *framework* utilizado foi o *TensorFlow*, os outros também poderiam ser boas soluções, mas alguns pontos do *TensorFlow* ressaltaram sua escolha:

- É um *framework* popular na comunidade, desta forma, assim como o *Python*, possui diversas fontes para aprendizado e resolução de problemas;
- Utilização por grupos de pesquisa relevantes como o *DeepMind*, adesão por grandes empresas como o *Twitter* além de ser desenvolvido pelo *Google*, garantindo maior credibilidade para sua escolha;
- O *framework* possui suporte para diversos modelos de redes neurais, desta forma, foi possível testar várias implementações diferentes como redes neurais sem realimentação até redes neurais recorrentes;
- O *framework* possui suporte para utilização de *CUDA*, plataforma desenvolvida pela *Nvidia*, agilizando o processo de treinamento da rede neural;
- Uma das dificuldades no desenvolvimento de redes neurais é compreender o que está ocorrendo nela para conseguir reparar problemas ou otimizar, portanto, o *TensorBoard* é interessante para facilitar o desenvolvimento por ser uma ferramenta visual para avaliação.

Para facilitar o uso do *TensorFlow*, foi utilizada a implementação do *Keras* já incluso no pacote, este é executado por cima do primeiro e sua função é permitir que a criação de redes neurais seja mais intuitiva e prática, assim, agilizando o processo de desenvolvimento, correção e otimização da solução proposta.

A escolha da arquitetura ideal para a rede neural é complexa de ser definida, muitas vezes é realizada por teste e avaliação de performance. Desta maneira, iniciou-se utilizando uma rede neural sem realimentação baseada no trabalho de (AHMED; HASEGAWA, 2013) em que foi desenvolvido com o uso de mínimo erro quadrático e averiguado diferentes valores de unidades em cada camada. Nas 2 camadas ocultas, o melhor resultado para o comando de leme foi dado por 15 unidades na primeira camada e 10 na segunda enquanto para comando de máquina, 10 unidades na primeira e 5 unidades na segunda. Esta foi apenas uma base para a arquitetura inicial, ela foi treinada com os dados do TPN e averiguada quanto a performance.

Como esta rede neural não satisfez os requisitos, outras arquiteturas mais complexas também foram testadas. Por exemplo, o uso de redes neurais recorrentes, pois esta apresenta uma “memória” das entradas passadas o que possibilita compreender que existe uma série

temporal entre os comandos, permitindo uma maior complexidade para o seu aprendizado, logo, esperou-se melhor aderência aos dados.

Para o processo de seleção da solução mais adequada foi realizado o treino da rede neural com os dados de treinamento e validado no integrador numérico, *Dyna*. A literatura como em (NG, 2018) e outras fontes abordam a escolha por meio da divisão dos dados de treinamento em duas ou três partes, basicamente, 70% dos dados para treinamento e 30% para validação. O problema desta abordagem, no caso deste projeto, é que o sistema em questão é um sistema dinâmico, portanto, se a rede neural avaliar que é necessário um comando de leme ou de máquina diferente daquele realizado pelo prático nos dados de treinamento, a próxima entrada registrada não estará retratando fielmente o estado naquele momento, logo, estará inválido. Desta maneira, os dados de treinamento foram utilizados para verificar se, pelo menos, a rede estava aderindo ao resultado esperado, assim, a comparação era útil para realizar uma primeira verificação do resultado, mas não sendo considerada condição suficiente para invalidar a solução obtida.

A última ferramenta a ser utilizada é do Centro de Simulações Náuticas e Portuárias do TPN que conta com simuladores para estudo de manobrabilidade de embarcações. Segundo (TPN-USP, 2017), o centro é credenciado pelo *ITTC*(*International Towing Tank Conference*) e *IMSF*(*International Marine Simulator Forum*), o que garante confiabilidade na simulação e credibilidade ao validar a rede neural proposta. Para este projeto, será utilizado o integrador numérico do simulador, denominado *Dyna*, que permite processar numericamente as equações que regem os fenômenos físicos da navegação de uma embarcação. Durante a fase de otimização da rede neural, foi elaborada uma interface em *Python* que permite a comunicação entre a rede neural e o *Dyna*. Sua função é permitir a comunicação entre as duas aplicações pré processando os dados no formato adequado e controlar as condições de simulação.

5 Dados para treinamento e simulação

5.1 Características dos navios

Existem diversos navios com as mais diversas propriedades físicas. Essas são essenciais de serem consideradas no simulador numérico a ser utilizado na validação, pois são as propriedades que influenciam no desenvolvimento do sistema dinâmico segundo as entradas selecionadas e as perturbações.

Para o presente projeto, baseado nos dados de simulações dos práticos, foram considerados, principalmente, 2 modelos de navios: o Aframax e o Suezmax, cujas características físicas são descritas na tabela 1 e as velocidades de seus propulsores são apresentados na tabela 2. Ambos são navios-tanque, sendo o Aframax o maior na escala *Average Freight Rate Assessment (AFRA)* e capacidade de carregamento entre 80 mil e 120 mil toneladas, já o Suezmax é um navio petroleiro com capacidade de carregamento entre 140 e 175 mil toneladas, cujas dimensões são as máximas suportadas pelo canal de Suez, no Egito (TRANSPETRO, 2018).

Tabela 1 – Características dos navios Aframax e Suezmax (em metros)

	Boca	Pontal	Comprimento	Calado
Aframax	42.00	22.50	244.75	15.30
Suezmax	48.00	23.10	278.50	15.00

Fonte: Autores

Tabela 2 – Velocidades dos navios Aframax e Suezmax (em rpm)

	0) Parado	1) Muito Devagar	2) Devagar	3) Meia Força	4) Toda Força
Aframax	0	19.20	38.40	57.60	76.80
Suezmax	0	28.77	32.88	57.54	65.76

Fonte: Autores

5.2 Canal de Suape

Todos os dados descritos na seção 5.4 foram simulados em um modelo do canal do porto de Suape localizado no estado de Pernambuco, Brasil. Este complexo está interligado a mais de 160 portos no mundo sendo o porto mais estratégico da região nordeste (SUAPE, 2018b). O porto apresenta uma área total organizada com mais de 3 mil hectares, porto

interno com 1.6 quilômetros de cais e 5 berços. Na figura 9 é apresentado o canal de acesso ao porto que possui as seguintes características (SUAPE, 2018a):

- Extensão: 5 quilômetros;
- Largura: 300 metros;
- Profundidade: 16.5 metros.

Características essas essenciais para verificar a capacidade de acesso de uma determinada embarcação e referência tanto para o prático definir os comandos para a manobra quanto para a rede neural obter os parâmetros de entrada.

Figura 9 – Canal de acesso do porto de Suape

Fonte: Eicomnor Engenharia

No modelo do canal para simulação, as referências para os cálculos são dadas pelas boias ilustradas na próxima seção na figura 10, nestes a largura ao longo do canal é de aproximadamente 200 metros com a abertura final de até 400 metros com aproximadamente 6 quilômetros de extensão.

5.3 Condições ambientais

Como descrito por (AHMED; HASEGAWA, 2013), na manobra de entrada no canal, conforme a velocidade da embarcação é reduzida, as condições ambientais intensificam seus efeitos sobre o navio afetando a sua controlabilidade, por isso a importância de sua consideração e a dificuldade de realização da manobra.

Existem basicamente três agentes ambientais que requerem atenção do prático: a correnteza, as ondas e o vento, que exercem forças e momentos sobre o navio podendo desviar da rota prevista. Os três estão implementados no simulador do TPN e serão considerados para realizar a validação da solução obtida.

Entretanto, existem infinitas combinações possíveis de intensidade, direção e sentido desses três agentes, desta forma, para padronizar a validação foi selecionado um caso de intensidade média dada na tabela 3.

Tabela 3 – Condições ambientais para validação sendo N direção norte e SE direção sudeste

Correnteza		Vento		Onda		
Direção	Intensidade	Direção	Intensidade	Direção	Período	Altura
N	0.5 nós	SE	15.0 nós	SE	8 s	1.0 m

Fonte: Autores

5.4 Dados de treinamento

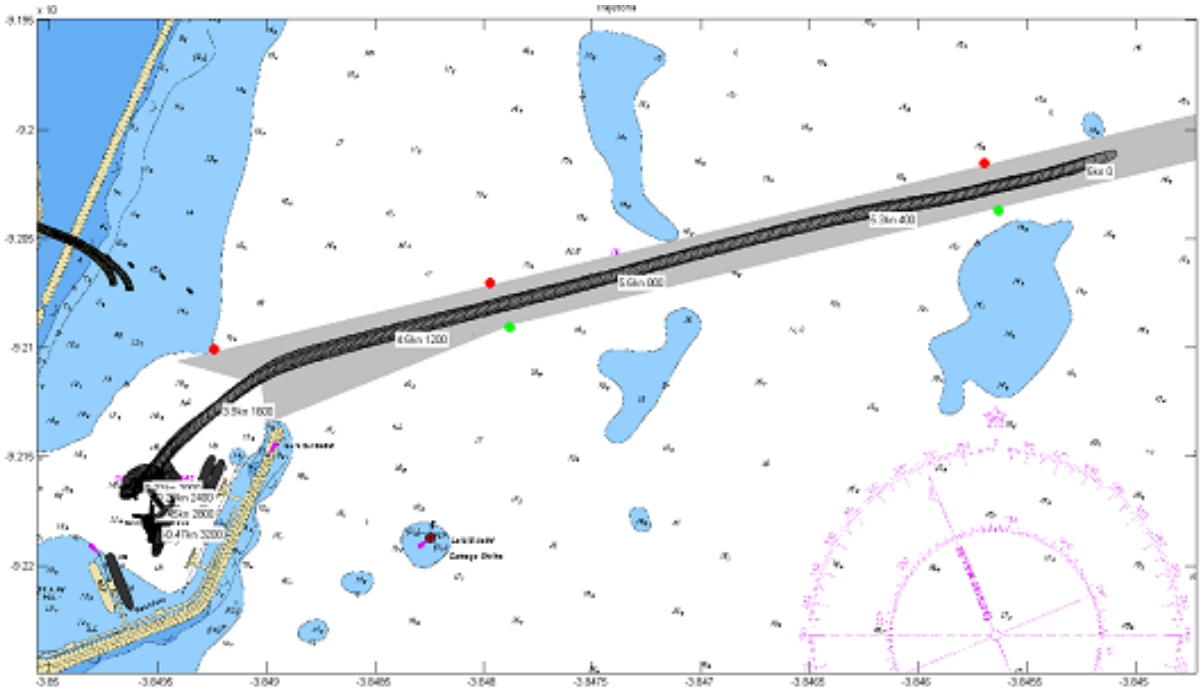
A primeira etapa do projeto foi o desenvolvimento dos dados de treinamento. Esta etapa visa processar os dados brutos que existem de simulações passadas realizadas por práticos no TPN de forma que resultem em dados de qualidade no formato necessário.

Inicialmente, foi obtido uma coleção de 113 simulações. Entretanto, nem todas eram válidas então foi necessário fazer uma análise individual dos vídeos das simulações. Na figura 10, denominado como caso 2 do Suape 2017, pode ser observado uma imagem com todos os estados representados em um desses vídeos, em cinza claro está o canal e em cinza escuro estão os estados da embarcação com o tempo. Os pontos vermelhos e verdes são as boias que definem as margens do canal. Por ser uma análise visual, foram adotados critérios menos precisos que depois seriam refinados por meio de algoritmo:

- A embarcação deve estar realizando uma manobra de entrada, ou seja, do mar para o berço;
- A embarcação deve permanecer dentro do canal durante toda a manobra.

Filtradas as simulações que satisfaçam essas condições, as mesmas foram processadas para gerar os arquivos de treinamento. Na tabela 4 pode ser visto o registro do primeiro segundo da simulação da figura 10, foram apresentadas apenas as colunas de interesse. Desta tabela, pode-se obter o instante de tempo em segundos de cada estado (*time_stamp*),

Figura 10 – Conjunto dos estados representados no vídeo da simulação do caso 2 do Suape 2017



Fonte: Orientador - TPN-USP

as coordenadas cartesianas em metro (x e y), o ângulo de aproamento em graus (zz), as velocidades locais em metros por segundo (vx e vy), a velocidade de guinada em graus por segundo (vzz), o comando de leme em radianos ($rudder\_demanded\_orientation\_0$) e a ordem de comando de máquina em rotações por minuto ($propeller\_demanded\_rpm\_0$).

Na tabela 5 está representada a saída ao se utilizar o algoritmo de processamento nos dados da tabela 4. De forma resumida, o algoritmo obtém os dados de simulação de planilhas auxiliares para formar o cabeçalho com o nome do navio, o tipo de cenário, o tipo de manobra executada e os dados de corrente, vento e onda. Posteriormente é realizado o processamento das distâncias de interesse mensurados em metros e representados na figura 11:

- Distância ao final do canal - '*distance_target*' (δ_t): relevante para que se saiba qual o comprimento de canal restante para redução de velocidade;
- Distância à linha de centro - '*distance_midline*' (δ_m): relevante para indicar o quanto a embarcação está distante da linha imaginária central do canal fornecendo informações para definir o comando de leme.

Tabela 4 – Primeiro segundo das colunas utilizadas do arquivo bruto com o conjunto dos estados representados no vídeo da simulação do caso 2 do Suape 2017

% number of rudders: 1												
% number of propellers: 8												
% time_stamp	x	y	...	zz	vx	vy	...	vzz	...	rudder_demanded_orientation_0	...	propeller_demanded_rpm_0
0.00	9605.72	4928.38	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.10	9605.47	4928.32	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.20	9605.22	4928.26	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.30	9604.97	4928.19	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.40	9604.72	4928.13	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.50	9604.47	4928.06	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.60	9604.22	4928.00	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.70	9603.96	4927.94	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.80	9603.71	4927.87	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
0.90	9603.46	4927.81	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
1.00	9603.21	4927.75	...	-160.00	2.57	-0.26	...	0.00	...	0.00	...	1.69
...	...	...	...	...	...	...	...	...	...	...	...	...

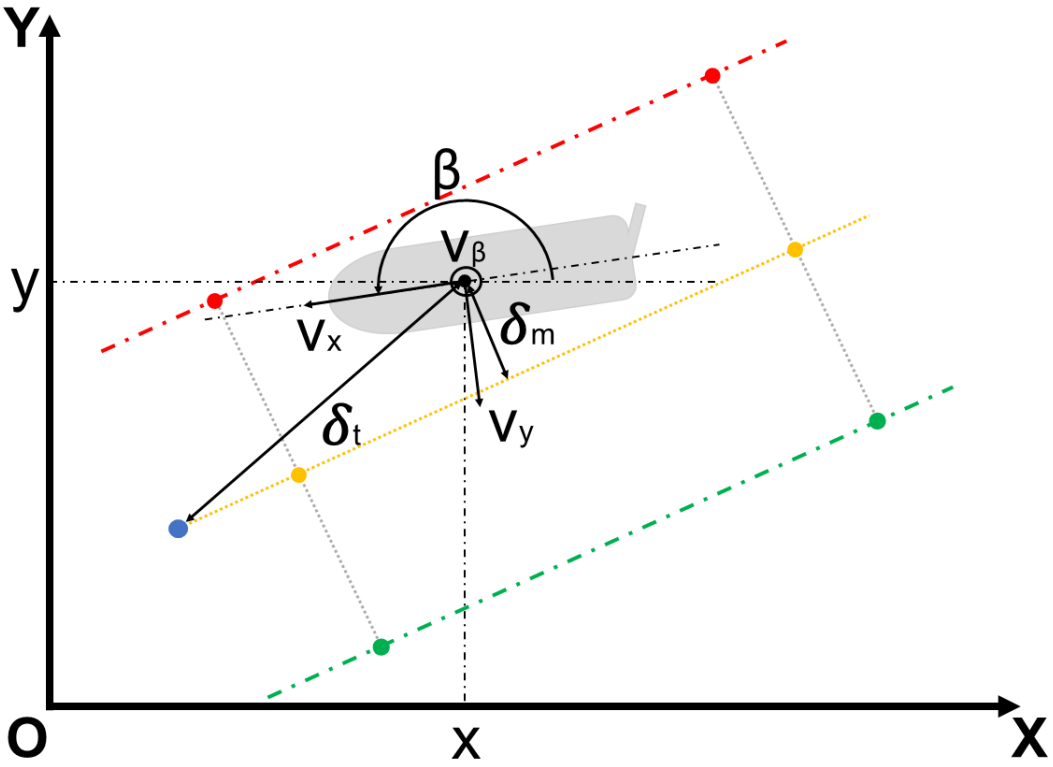
Fonte: Orientador - TPN-USP

Tabela 5 – Primeiro segundo das informações processadas do conjunto dos estados representados no vídeo da simulação do caso 2 do Suape 2017

Navio: Suezmax T150							
Cenário: Médio							
Manobra: Entrada completa							
Corrente: N 0.5nó							
Vento: SE 15nós							
Onda: 1.0m 8s SE							
zz	vx	vy	vzz	rudder_demanded	propeller_ demanded	distance_midline	distance_target
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.112	3940.353
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.111	3940.096
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.111	3939.839
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.12	3939.579
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.119	3939.322
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.129	3939.063
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.128	3938.805
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.125	3938.539
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.134	3938.279
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.134	3938.022
-160.0	2.57	-0.26	0.0	0.0	0.0	-33.133	3937.765
...	...	...	...	...	...	...	...

Fonte: Autores

Figura 11 – Diagrama esquemático dos parâmetros da embarcação que serão utilizados como entradas. As linhas ponto-tracejadas verde e vermelho representam as linhas imaginárias que são definidas pelas boias (pontos circulares). A linha tracejada cinza é a linha imaginária que liga as boias correspondentes para determinação dos pontos médios amarelos que definem a linha imaginária central também em amarelo.



Fonte: Autores

As coordenadas da boia e do ponto final do canal são obtidas manualmente para cada porto. De posse dessas coordenadas e da embarcação, foi utilizado geometria analítica para cálculo das distâncias (para o δ_t distância ponto a ponto do centro da embarcação até o objetivo e para δ_m distância ponto a reta do centro da embarcação até a linha imaginária central). A linha imaginária central é calculada com base nas boias, para cada par de boias (verde e vermelho) correspondentes é obtido o ponto médio (amarelo) e a sequência desses pontos definem a linha central, ressalta-se que esta é uma linha algébrica que não existe na realidade sendo a mesma observação válida ao ponto amarelo.

Por fim, no arquivo original, a velocidade do propulsor é dado em *rpm*, mas no caso de um prático, o comando é dado em ordem de máquina, ou seja, *stop* (parado), *dead slow* (muito devagar), *slow* (devagar), *half* (meia força) e *full* (toda força) representado por valores inteiros de 0 a 4, respectivamente. Para os valores positivos estão representadas as ordens *ahead* (à vante) e em negativo as ordens *astern* (à ré). Na tabela 2 pode ser

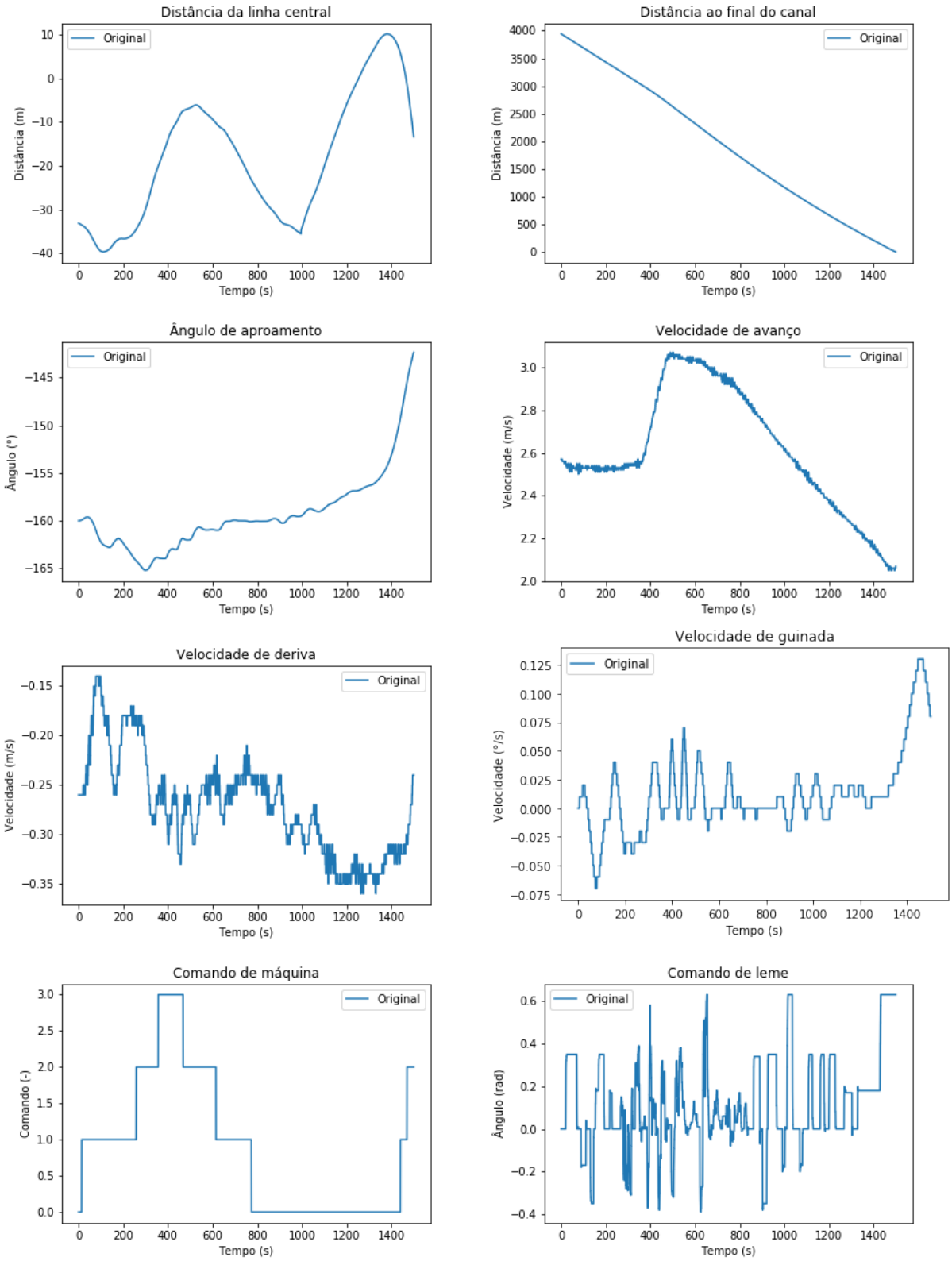
observada a conversão entre os valores para o Aframax e o Suezmax.

Portanto, por meio do cabeçalho gerado, parte dos dados originais e alguns parâmetros calculados, são obtidos os dados processados que serão utilizados no treinamento. O último detalhe é filtrar para que apenas os dados úteis sejam apresentados no arquivo de saída, isso é necessário, pois embora a simulação esteja boa para ser utilizada, ela pode conter partes que não são de interesse, por exemplo, após a manobra de entrada no canal pode existir uma manobra com rebocadores que fazem o atracamento efetivo, manobra esta fora do escopo deste projeto.

De forma a exemplificar os resultados, a figura 12 consolida o resultado desse processamento no caso 2 do Suape 2017. Nota-se que os resultados estão coerentes, as aproximações e afastamentos da linha imaginária central ocorrem em sincronia com os movimentos correspondentes da figura 10, a embarcação manobra em direção ao ponto final (distância ao final do canal tende a zero), a velocidade de avanço é reduzida conforme o requisito e as ordens de máquina estão discretizadas.

Para finalizar, um panorama geral sobre esse processo é que foram examinadas 113 simulações, mas apenas 45 foram aprovados para efetivamente serem utilizados no treinamento da rede neural. Embora esse número aparente ser pequeno, vale notar que cada arquivo possui dezenas de milhares de estados, desta forma, ao contabilizar todos os estados que serão utilizados, a soma se aproxima de quase 650 mil.

Figura 12 – Evolução dos estados na simulação do caso 2 do Suape 2017



Fonte: Autores

6 Arquitetura da rede neural

De forma que a rede neural possa funcionar como esperado é necessário que ela possua uma arquitetura definida.

O primeiro passo é realizar a escolha adequada dos parâmetros de entrada e de saída. Como o objetivo da rede neural é o controle da embarcação, seu movimento é definido por dois comandos básicos:

- Comando de leme (θ): define a direção da embarcação e é mensurado em radianos;
- Comando de máquina (ω): define a rotação do propulsor e é adimensional variando de -4 a 4.

Para as entradas, foram escolhidos os parâmetros mais relevantes para a tomada de decisão dos práticos:

- Distância ao final do canal - '*target*' (δ_t): mensurado em metros;
- Distância à linha imaginária central - '*midline*' (δ_m): mensurado em metros;
- Velocidade local (v_x, v_y): mensurado em metros por segundo;
- Ângulo de aproamento (β): mensurado em graus;
- Velocidade de guinada (v_β): mensurado em graus por segundo.

O detalhamento para apurar esses valores estão descritos na seção 5.4. Conforme o desenvolvimento da rede neural é possível apurar sua performance e realizar ajustes. Por exemplo, inicialmente a rede neural possuía como entrada as distâncias às margens definidas pelas boias a bombordo (δ_p) e a boreste (δ_s), mas os resultados não estavam satisfatórios e foi realizada uma alteração para substituir as duas medidas por uma única que é a distância à linha imaginária central como utilizada por (AHMED; HASEGAWA, 2013). Essa substituição é vantajosa porque simplifica os parâmetros que a rede neural precisa averiguar para aderir e reduz a possibilidade das entradas não serem linearmente independentes, pois as variáveis δ_p, δ_s e β podem possuir uma dependência. Considerando tais variáveis, foram estabelecidas as seguintes relações entre elas:

$$\theta = f(v_x, v_y, \beta, v_\beta, \delta_m) \tag{19}$$

$$\omega = g(v_x, v_y, \beta, v_\beta, \delta_m, \delta_t) \tag{20}$$

Como explicado no capítulo 3, existem diversas formas de estruturar a rede neural (representação das funções $f(\cdot)$ e $g(\cdot)$), a tarefa de definir os hiperparâmetros mais adequados nem sempre possuem metodologias exatas e muitas vezes são definidos por meio de tentativa e erro. No presente trabalho foram realizadas diversas alterações que eram validadas conforme a metodologia apresentada no capítulo 4.

As primeiras tentativas de rede neural foram as redes sem realimentação, que como discutido em mais detalhes na seção 8.1, não apresentaram resultados satisfatórios. Por essa razão, a solução por redes neurais recorrentes foi cogitada na expectativa de um melhor aprendizado. Sua utilização poderia ser mais adequada devido à natureza do problema a ser solucionado: os dados de treinamento são uma discretização no tempo, intervalos de 0.1 segundo, de uma sequência de estados da embarcação, portanto, uma abordagem que possa compreender a existência de uma ordem entre os estados poderia ser mais eficiente. Desta maneira, foram realizados testes com unidades de GRU (*Gated Recurrent Unit*) e LSTM (*Long Short-Term Memory*). Diversas variantes foram testadas e o trabalho versará sobre as que obtiveram os melhores resultados. Estas estão descritas na tabela 6 e com resultados na seção 8.3.

Tabela 6 – Parâmetro das redes neurais recorrentes sendo *MSE* referente a equação 11

Rede	Taxa de aprendizado	Camadas (unidades)	Dropout (%)	Função de ativação	Função de custo
1	0.001	GRU(128)/GRU(128)	0	*tanh*	MSE
2	0.001	Dense(128)/GRU(128)	20	*sigmoid*	MSE
3	0.001	GRU(128)/LSTM(128)	0	*sigmoid*	MSE
4	0.001	GRU(256)/GRU(128)/GRU(32)	10	*sigmoid*	MSE

Fonte: Autores

7 Validação

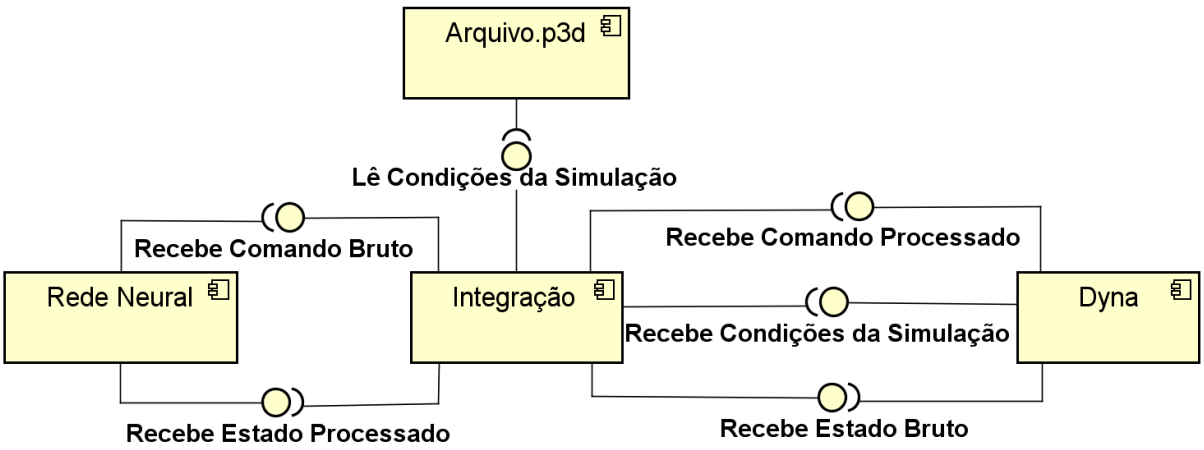
7.1 Módulo de integração

Como explicado no capítulo 4, o teste e validação da rede neural pela metodologia tradicional não é a mais adequada devido ao sistema controlado ser um sistema dinâmico. Desta maneira, a abordagem escolhida para realizar os testes e, de fato, definir se a solução está controlando a embarcação de forma adequada é o uso de simulação.

Para possibilitar agilidade no desenvolvimento do controlador e sua validação, foi necessário elaborar um sistema que integrasse o treinamento da rede neural com o processo de simulação, logo, o ciclo estaria automatizado permitindo que os autores não necessitassem interferir manualmente entre uma tarefa e outra. Essa ferramenta é de utilidade porque, dependendo da estrutura da rede neural, o seu treinamento demanda tempo, desta maneira, automatizar o ciclo de validação permite executar todo o processo de forma contínua.

A estrutura básica para elaboração da integração pode ser observada no diagrama de componentes da figura 13. O componente que este capítulo aborda é o denominado "Integração". Sua função principal é ser uma interface de comunicação entre a rede neural e o *Dyna*.

Figura 13 – Diagrama de componentes demonstrando a funcionalidade da interface de integração entre a rede neural e o *Dyna*



Fonte: Autores

O *Dyna* é o integrador numérico desenvolvido no TPN que permite simular a navegação das embarcações, as equações que regem os fenômenos que afetam a embarcação

estão implementados para serem resolvidos numericamente. Existem diversas condições que podem afetar a simulação, por exemplo, as características físicas da embarcação, o relevo submarino da região, a direção e a intensidade do vento, a direção e a intensidade da maré e muitos outros parâmetros. Para definição dessas influências, o integrador busca no arquivo do tipo "P3D" os valores a serem considerados e envia para o *Dyna* construir o ambiente de simulação.

Para fazer a manobra da embarcação, existem basicamente dois comandos que podem ser enviados para o integrador: o comando de máquina e o comando de leme. A rede neural desenvolvida busca simular de forma coerente com o que os práticos realizam, assim, os comandos de máquina variam entre *stop*, *dead slow*, *slow*, *half* e *full* nas direções *astern* e *ahead* enquanto o *Dyna* recebe o percentual da velocidade máxima das características da embarcação simulada. De maneira análoga funciona o comando de leme, a rede neural define comandos em radianos e o *Dyna* recebe em percentual do máximo de leme. Portanto, o módulo de integração também serve para realizar essa compatibilização de comunicação entre os componentes.

Também faz parte deste módulo o cálculo de δ_m e δ_t que são parâmetros de entrada da rede neural a serem enviados em conjunto com os dados do estado que são obtidos pelo retorno do *Dyna*.

A partir da execução da simulação, o *Dyna* gera um arquivo como o da tabela 4 que poderá ser utilizado posteriormente para gerar os gráficos semelhantes ao da figura 12. Para facilitar a visualização da simulação, uma plotagem dos estados é realizada em tempo de execução para verificar sua trajetória como apresentado na figura 22. É importante observar que a embarcação não está representada em escala e em formato preciso ao modelo simulado, pois esta é apenas uma ferramenta de acompanhamento, a análise mais precisa e correta deve ser realizada diretamente nos dados de saída da simulação.

Esse módulo foi desenvolvido sobre outro módulo pré-existente do aluno de mestrado José Amendola Netto Andrade de quem são os direitos autorais do código. Por parte dos autores foram realizadas apenas adaptações para considerar o modelo de rede neural proposto.

7.2 Condições iniciais

Um aspecto importante para realizar a simulação é a definição da condição inicial do navio, ou seja, quais os valores dos parâmetros que definem o estado do navio no momento de partida. Para tal, o *Dyna* requer os seguintes valores:

- Coordenada x global (x): mensurado em metros;
- Coordenada y global (y): mensurado em metros;
- Ângulo de aproamento (β): mensurado em graus;
- Velocidade de avanço (v_x): mensurado em metros por segundo;
- Velocidade de deriva (v_y): mensurado em metros por segundo;
- Velocidade de guinada (v_β): mensurado em graus por segundo.

Na seção 8.3 serão discutidos os resultados das simulações para diferentes condições iniciais de forma a analisar a influência de cada parâmetro no desempenho da rede neural para a execução da manobra.

A tabela 7 apresenta as condições referentes a influência do posicionamento inicial mais próximo da margem bombordo (1), exatamente na linha média (2) ou mais próximo da margem boreste (3) com resultados descritos na subseção 8.3.2.

Tabela 7 – Parâmetros do navio para as condições iniciais no teste de posicionamento

Condição inicial	x **(m)**	y **(m)**	β (°)	v_x (m/s)	v_y (m/s)	v_β (°/s)
1	**11611.97**	**5404.08**	-166.45	4.00	0.00	0.00
2	**11601.97**	**5445.60**	-166.45	4.00	0.00	0.00
3	**11591.96**	**5487.15**	-166.45	4.00	0.00	0.00

Fonte: Autores

A tabela 8 apresenta as condições referentes a influência do ângulo de aproamento inicial direcionando a embarcação para a margem bombordo (4), alinhado ao canal (5) ou para a margem boreste (6) com resultados descritos na subseção 8.3.3.

Tabela 8 – Parâmetros do navio para as condições iniciais no teste de ângulo de aproamento, sendo que a condição 5 é a mesma que a condição 2 da tabela 7

Condição inicial	x (m)	y (m)	β **(°)**	v_x (m/s)	v_y (m/s)	v_β (°/s)
4	11601.97	5445.60	**-151.45**	4.00	0.00	0.00
5 (2)	11601.97	5445.60	**-166.45**	4.00	0.00	0.00
6	11601.97	5445.60	**-181.45**	4.00	0.00	0.00

Fonte: Autores

A tabela 9 apresenta as condições referentes a influência da velocidade de guinada inicial no sentido anti-horário (7), sem rotação (8) ou sentido horário (9) com resultados descritos na subseção 8.3.4.

Tabela 9 – Parâmetros do navio para as condições iniciais no teste de velocidade de guinada, sendo que a condição 8 é a mesma que a condição 2 da tabela 7

Condição inicial	x (m)	y (m)	β (°)	v_x (m/s)	v_y (m/s)	v_β **(°/s)**
7	11601.97	5445.60	-166.45	4.00	0.00	**-0.11**
8 (2)	11601.97	5445.60	-166.45	4.00	0.00	**0.00**
9	11601.97	5445.60	-166.45	4.00	0.00	**0.11**

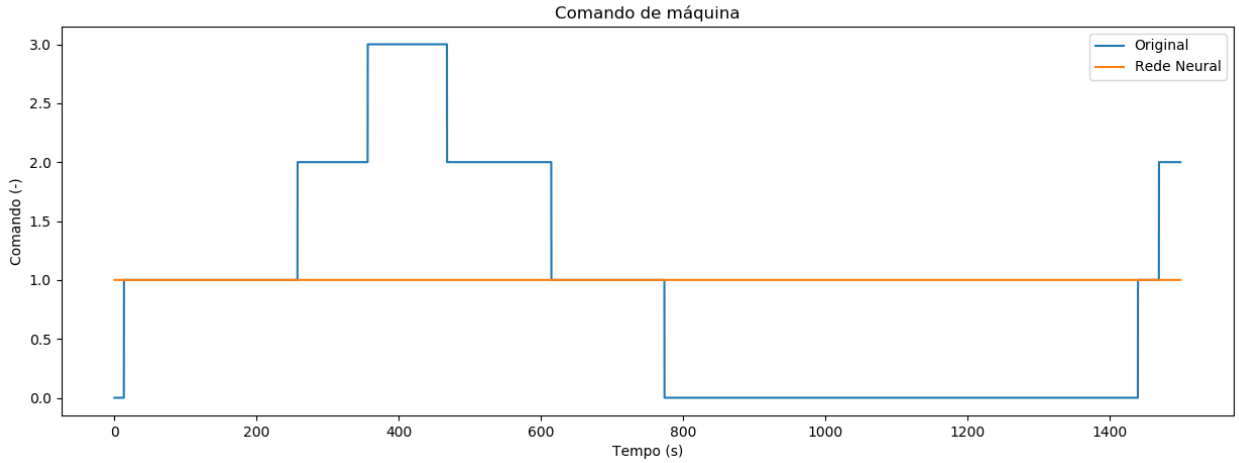
Fonte: Autores

8 Resultados

8.1 Resultados preliminares

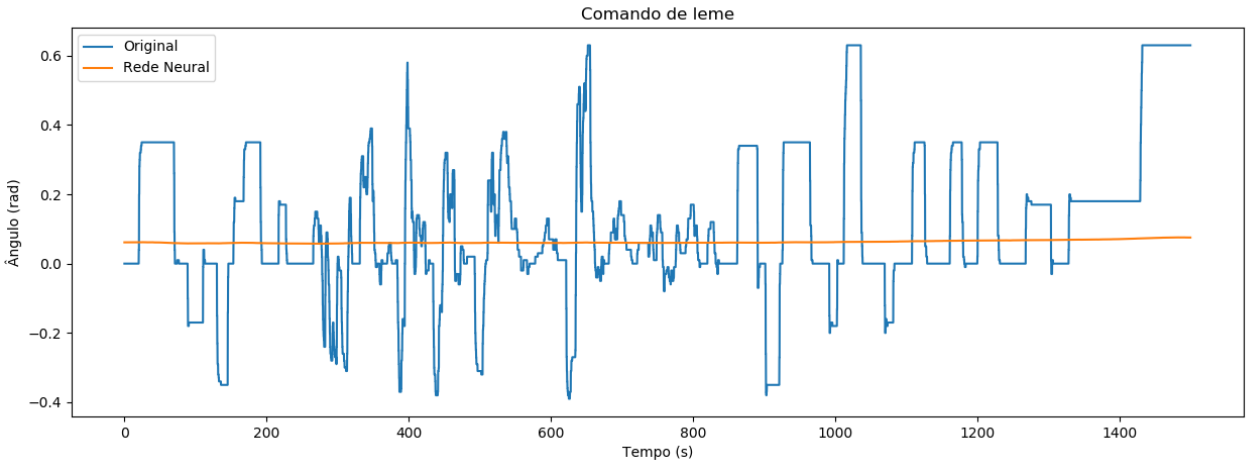
Iniciou-se o processo de apuração pelas redes sem realimentação que não estavam aderindo apropriadamente aos dados de treinamento, muitas vezes apresentando saídas constantes ou com variação mínima como apresentado nas figuras 14 e 15 que simularam os mesmos estados da figura 12.

Figura 14 – Comando de máquina uniforme obtido com a rede neural sem realimentação no caso 2 do Suape 2017 demonstrando não estar controlando adequadamente



Fonte: Autores

Figura 15 – Comando de leme uniforme obtido com a rede neural sem realimentação no caso 2 do Suape 2017 demonstrando não estar controlando adequadamente

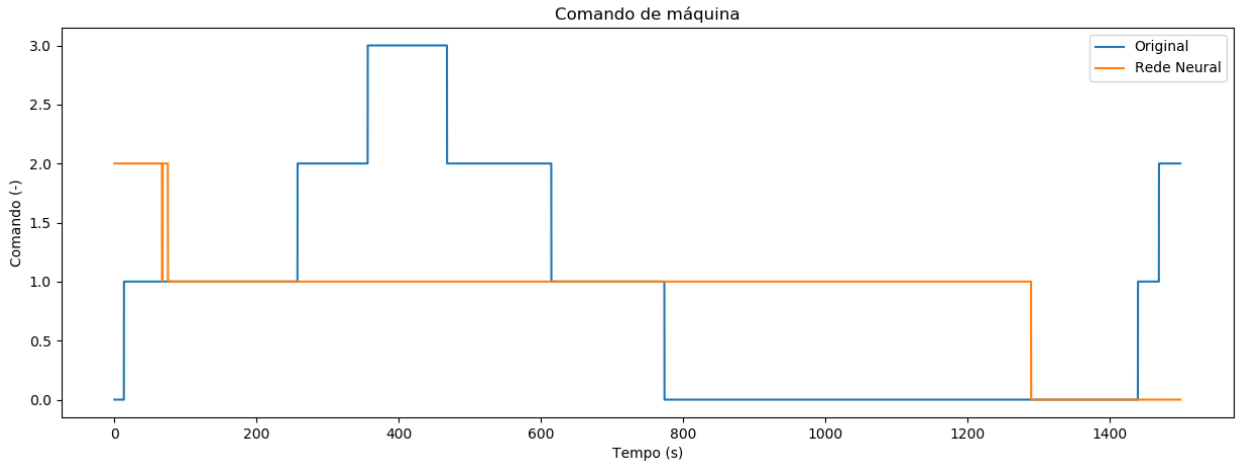


Fonte: Autores

O modelo baseado em (AHMED; HASEGAWA, 2013), descrito no capítulo 4,

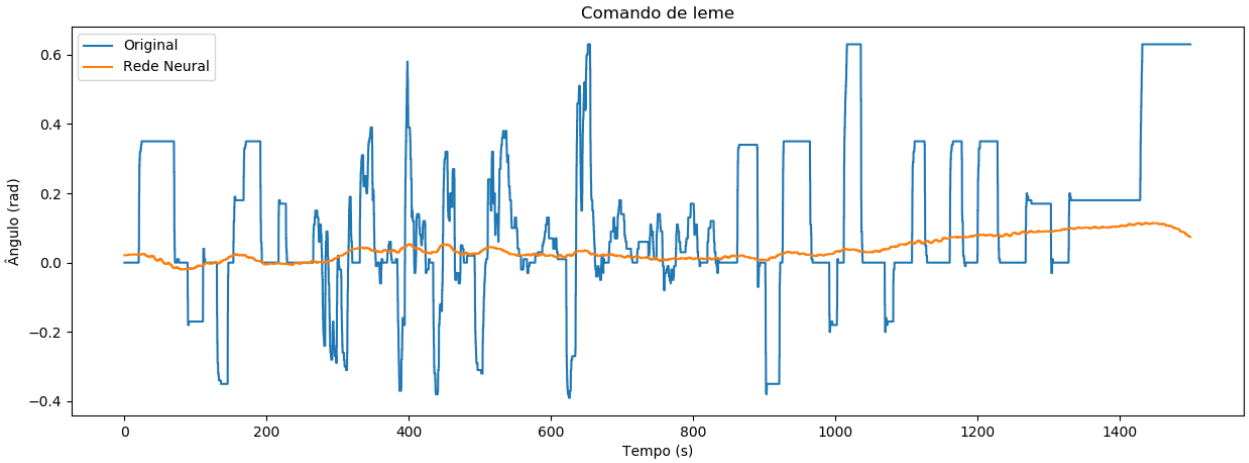
apresentou uma melhora nos resultados como pode ser observado nas figuras 16 e 17. É possível verificar uma maior oscilação em relação às figuras 14 e 15, mas ainda com baixa amplitude sendo insuficiente para controle em situações que requerem comandos mais abruptos.

Figura 16 – Comando de máquina com a rede neural de arquitetura baseada em (AHMED; HASEGAWA, 2013) no caso 2 do Suape 2017



Fonte: Autores

Figura 17 – Comando de leme com a rede neural de arquitetura baseada em (AHMED; HASEGAWA, 2013) no caso 2 do Suape 2017



Fonte: Autores

Desta maneira, foram requeridas duas etapas para contornar a não funcionalidade destas primeiras tentativas: primeiramente uma análise mais criteriosa dos dados a fim de verificar a viabilidade de sua utilização como descrito em detalhes na seção 8.2 e

posteriormente a utilização de redes neurais recorrentes como solução final detalhado na seção 8.3.

8.2 Análise de dados

Um aspecto importante em qualquer aprendizado de máquina é a qualidade dos dados disponíveis para uso, ou seja, o algoritmo desenvolvido pode apresentar um desempenho indesejado de acordo com a base de dados que foi utilizada para treinar o mesmo. Nesse caso realizou-se a análise dos parâmetros de entrada com os de saída.

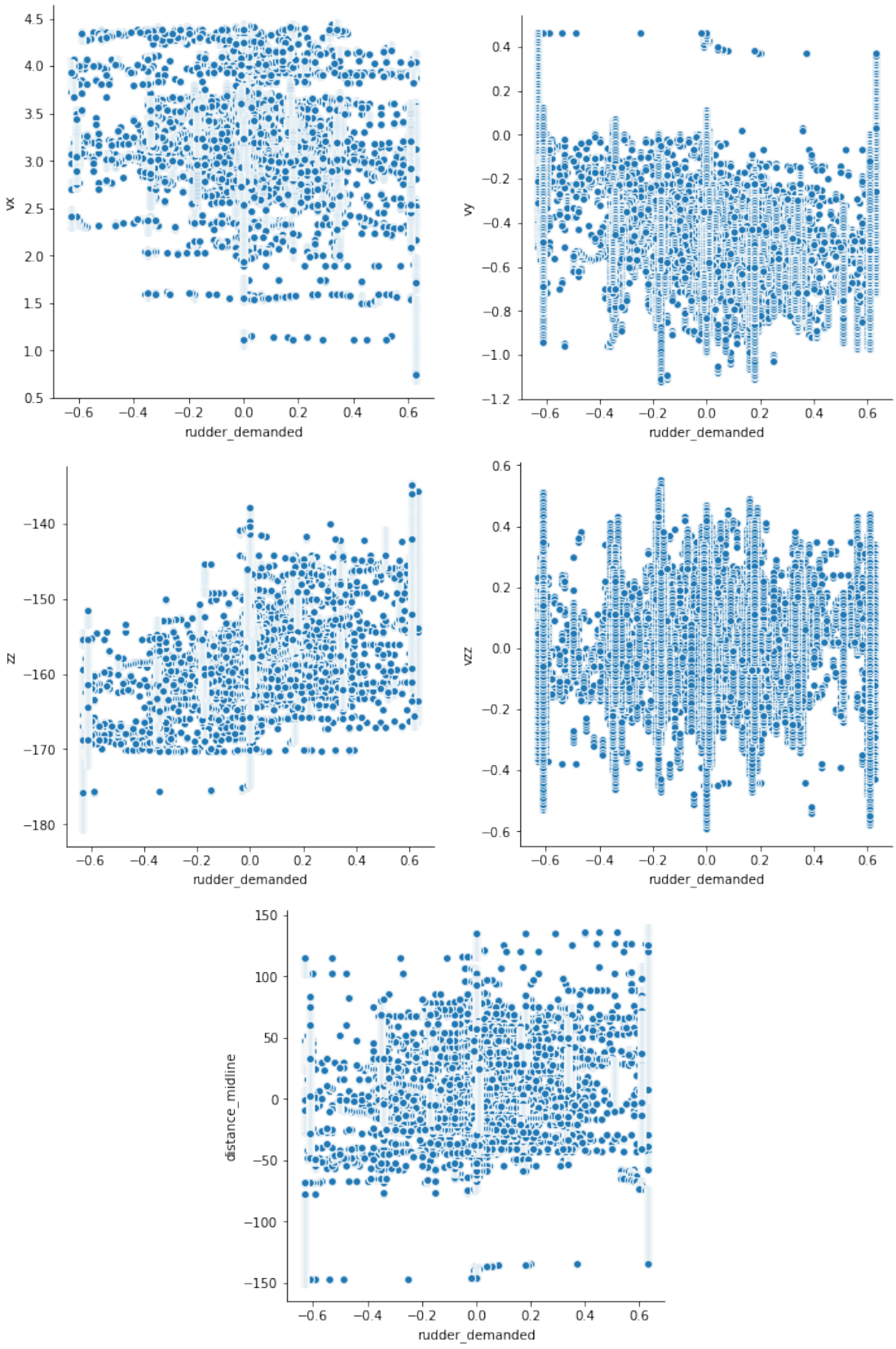
As figuras 18 e 19 apresentam a distribuição de cada parâmetro de entrada descrito nas equações 19 e 20, respectivamente, para cada saída. De forma geral é difícil identificar padrões nessas imagens embora a expectativa fosse de que elas existissem claramente. Por exemplo, uma expectativa era de que no gráfico δ_m por θ fosse verificado uma concentração maior em δ_m negativo com θ positivo e também em δ_m positivo com θ negativo, pois são os comandos que direcionariam a embarcação para a linha de centro.

Entretanto, para realizar a manobra da embarcação é necessário considerar o conjunto todo desses parâmetros e não individualmente. Por essa razão existe a dificuldade em identificar os padrões, por exemplo, se v_β estiver ascendente rapidamente, para o prático é mais importante reduzir essa taxa de aumento do que realizar o alinhamento da embarcação na linha de centro, pois a embarcação pode se tornar muito instável. Esta é uma das razões para a existência de pontos em todo o espaço de δ_m por θ, explicando a quebra da expectativa apresentada.

No caso do comando de máquina, figura 19, o aparente padrão entre as variáveis se deve ao fato da discretização dos valores do comando de máquina, assim, não existe o preenchimento do espaço todo, o que torna a aparência menos caótica.

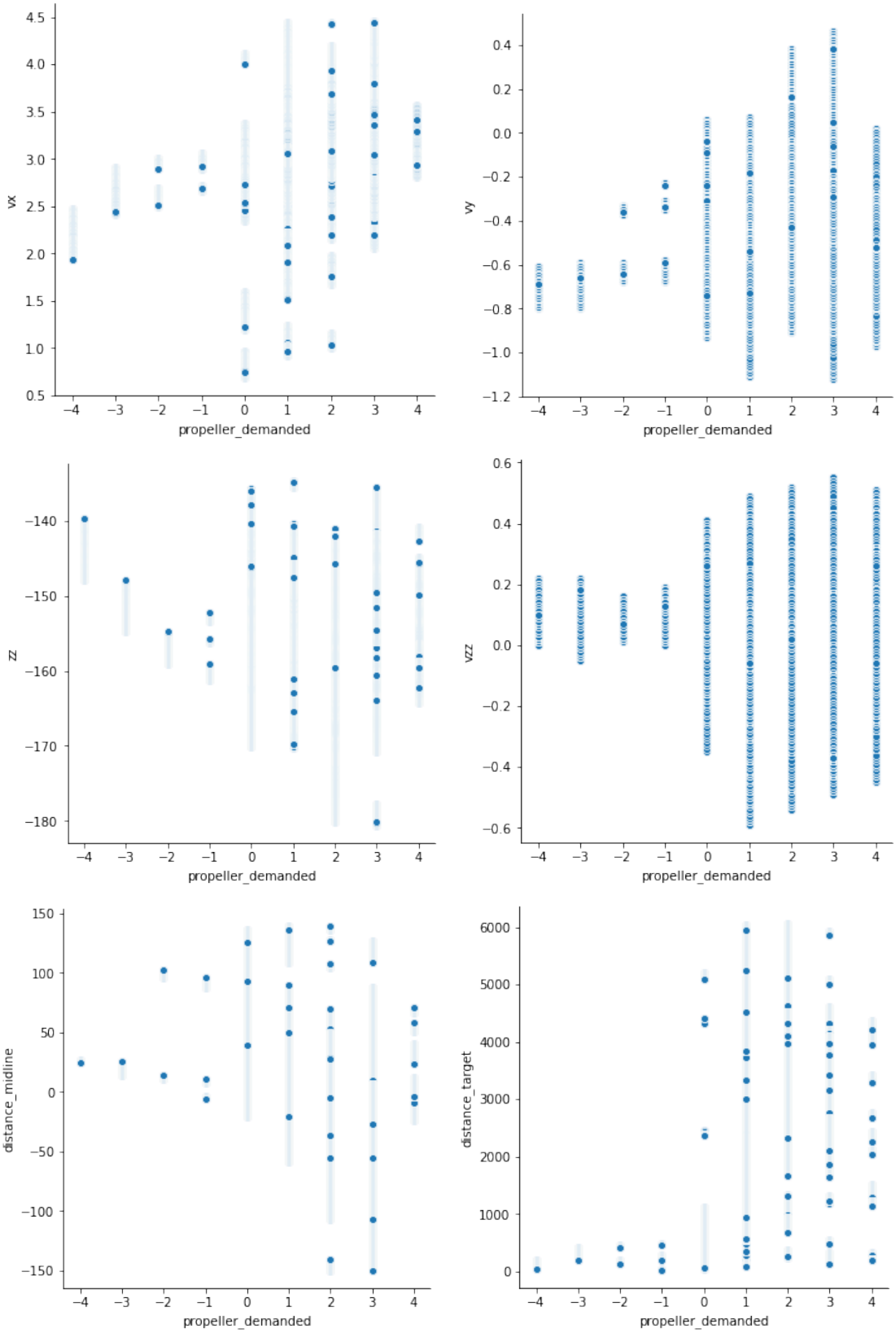
De maneira a realizar uma análise mais precisa, foi realizada uma verificação por meio de correlação. Na tabela 10 está representada a correlação dos parâmetros com a saída no caso 2 do Suape 2017 apresentado na seção 5.4. Pode-se observar que poucas variáveis apresentaram um índice de correlação acima de 0.5 e nenhum acima de 0.7 que seria um valor interessante para garantir um melhor aprendizado por parte da rede neural. Conforme a base de dados é aumentada com os demais casos, tabela 11, é possível perceber que alguns parâmetros melhoram os resultados da correlação como v_β com ω enquanto

Figura 18 – Plotagem dos parâmetros em relação ao comando de leme (θ)



Fonte: Autores

Figura 19 – Plotagem dos parâmetros em relação ao comando de máquina (ω)



Fonte: Autores

outros pioram como β com θ.

A análise das correlações individuais e conjunta demonstra a irregularidade que existe na base devido a complexidade em se realizar este tipo de manobra que pondera diversos fatores simultaneamente. A maneira que a base foi gerada também influencia nos resultados observados, ela foi produzida por práticos diferentes que possuem experiência e técnicas de manobra diferentes, existem embarcações com propriedades físicas distintas e condições ambientais que variam entre os casos. Essa diversidade torna a base rica em situações que auxiliam a obter uma rede neural mais genérica, entretanto aumenta a complexidade para obter os padrões de controle e gera a aparência caótica das figuras 18 e 19.

Tabela 10 – Análise de correlação das variáveis para o caso 2 do Suape 2017

Variáveis	Comando de leme	Comando de máquina
v_x	-0,32	0,38
v_y	-0,29	0,36
β	0,54	-0,39
v_β	0,51	-0,01
δ_m	0,21	-0,01
δ_t	-	0,59

Fonte: Autores

Tabela 11 – Análise de correlação das variáveis para a base inteira de dados

Variáveis	Comando de leme	Comando de máquina
v_x	-0,18	0,40
v_y	-0,25	0,04
β	0,05	-0,05
v_β	0,06	-0,10
δ_m	0,15	-0,11
δ_t	-	0,48

Fonte: Autores

8.3 Resultados finais

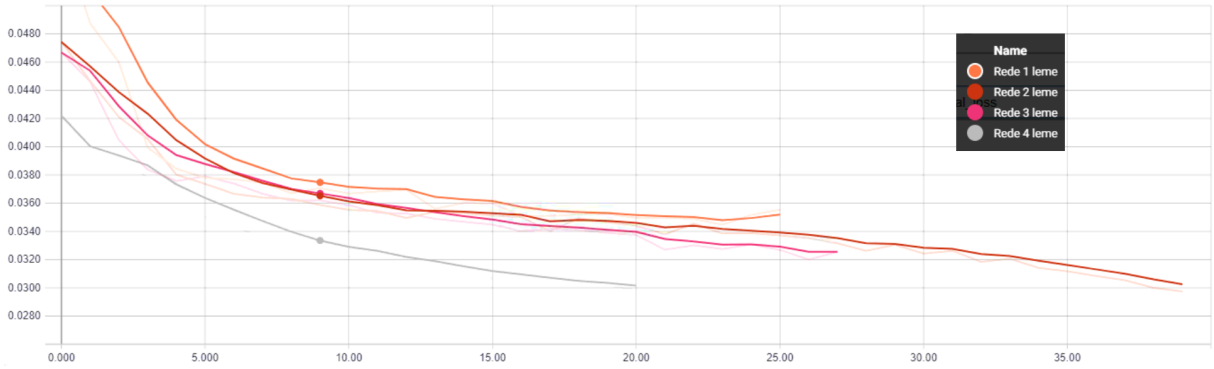
8.3.1 Arquitetura e treinamento

Existem diversos parâmetros que podem ser definidos nas redes neurais para obter resultados mais apropriados a cada situação. Portanto, definida a escolha da utilização de redes neurais recorrentes, foram desenvolvidos diversos modelos pela alteração da arquitetura e dos hiperparâmetros, alguns com resultados melhores e outros piores. Para

a primeira filtragem foram realizadas análises mais simples como o uso do conjunto de estados de simulações passadas não utilizadas no treinamento a fim de verificar o erro entre os comandos registrados e os previstos pelo controlador, semelhante ao realizado na seção 8.1.

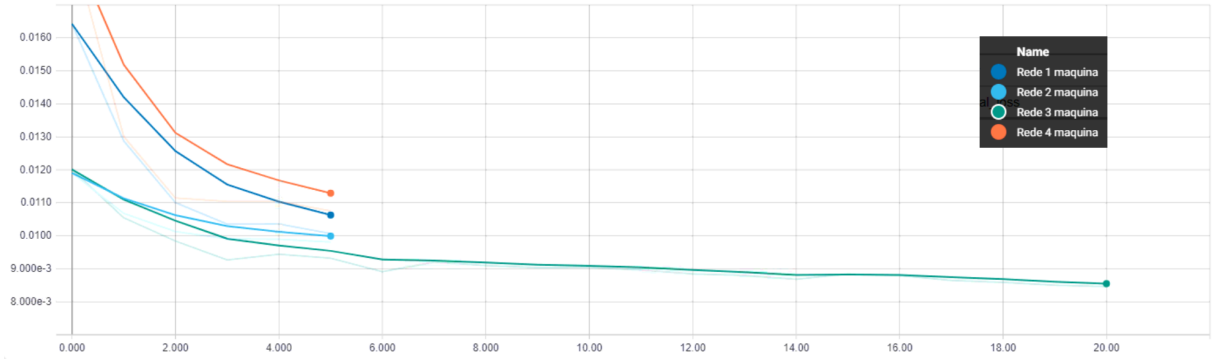
De posse dos resultados, os melhores foram selecionadas para realização de algumas simulações preliminares com o *Dyna* para averiguar se existe realmente um indício de que a rede resulta em comandos de controle coerentes. Desses, foram selecionados os 4 modelos que apresentaram os resultados mais satisfatórios (tabela 6) para serem utilizados no processo de validação completo descrito no capítulo 7.

Figura 20 – Valor da função de custo para modelos de rede neural de controle de leme obtida no *Tensorboard* em função da quantidade de *epoch*



Fonte: Autores

Figura 21 – Valor da função de custo para modelos de rede neural de controle do comando de máquina obtida no *Tensorboard* em função da quantidade de *epoch*



Fonte: Autores

O *Tensorboard* permite observar o desenvolvimento da rede neural durante a fase de treinamento. Nas figuras 20 e 21 é possível observar a variação da função de custo

com o decorrer desta etapa nos controladores selecionados para comando de leme e de máquina, respectivamente. Nota-se, portanto, que o processo de treinamento dos modelos selecionados foi realizado com sucesso, uma vez que a função custo mede o erro da saída prevista em relação aos valores originais de teste. Pelos gráficos, detecta-se que a função custo é minimizada durante o processo de aprendizagem, ou seja, a rede passa a aderir melhor aos dados conforme os *epochs*.

Verificado a adequação do processo de treinamento destes modelos, em seguida foi realizado o processo de validação utilizando a interface de integração, figura 13, para realizar a simulação via *Dyna*. Nas simulações, utilizou-se o modelo de navio Aframax, tabelas 1 e 2, com os parâmetros de condições iniciais das tabelas 7, 8 e 9.

8.3.2 Variação de posicionamento

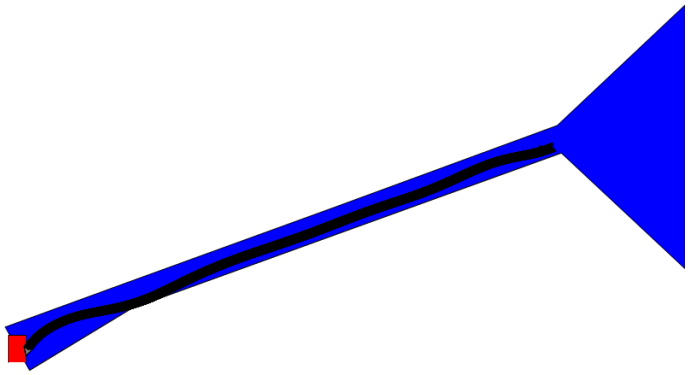
Como descrito na seção 7.2, foi realizado a simulação para os casos da tabela 7, obtendo os resultados descritos na tabela 12. A figura 22 apresenta um caso de simulação realizado com sucesso, ou seja, não há ocorência de colisões durante o trajeto. Os demais resultados das simulações podem ser observados no apêndice A.

Tabela 12 – Resultados do teste de posicionamento ✓: sem colisão ✗: com colisão

Condição inicial	Rede 1	Rede 2	Rede 3	Rede 4
1	✓	✗	✓	✓
2	✓	✗	✗	✓
3	✓	✗	✗	✓

Fonte: Autores

Figura 22 – Simulação do controle da embarcação através da rede 1 na condição 1



Fonte: Autores

8.3.3 Variação de ângulo de aproamento

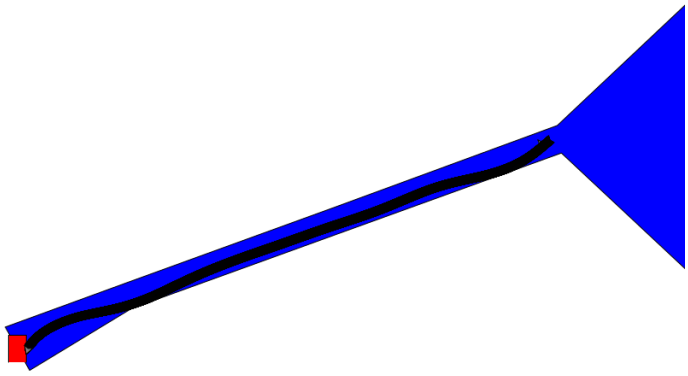
Como descrito na seção 7.2, foi realizado a simulação para os casos da tabela 8, obtendo os resultados descritos na tabela 13. A figura 23 apresenta um caso de simulação realizado com sucesso, ou seja, não há ocorência de colisões durante o trajeto. Os demais resultados das simulações podem ser observados no apêndice A.

Tabela 13 – Resultados do teste de ângulo de aproamento ✓: sem colisão ✗: com colisão

Condição inicial	Rede 1	Rede 2	Rede 3	Rede 4
4	✓	✗	✓	✓
5 (2)	✓	✗	✗	✓
6	✗	✗	✗	✗

Fonte: Autores

Figura 23 – Simulação do controle da embarcação através da rede 1 na condição 4



Fonte: Autores

8.3.4 Variação de velocidade de guinada

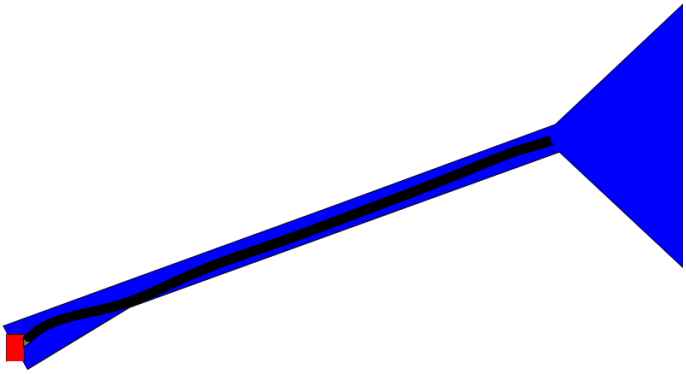
Como descrito na seção 7.2, foi realizado a simulação para os casos da tabela 9, obtendo os resultados descritos na tabela 14. A figura 24 apresenta um caso de simulação realizado com sucesso, ou seja, não há ocorência de colisões durante o trajeto. Os demais resultados das simulações podem ser observados no apêndice A.

Tabela 14 – Resultados do teste de velocidade de guinada ✓: sem colisão ✗: com colisão

Condição inicial	Rede 1	Rede 2	Rede 3	Rede 4
7	✓	✓	✗	✓
8 (2)	✓	✗	✗	✓
9	✓	✗	✗	✓

Fonte: Autores

Figura 24 – Simulação do controle da embarcação através da rede 1 na condição 7



Fonte: Autores

9 Discussão

9.1 Análise de desempenho

A partir dos resultados das simulações obtidas, tabelas 12, 13 e 14, podemos avaliar o desempenho das redes neurais durante as 7 simulações de acordo com o requisito primário que é a condução do navio até o final do canal sem colisão. Vale notar que no teste 6, tabela 13, nenhuma rede conseguiu controlar a embarcação até o final, tais resultados podem ser justificados pela instabilidade do navio ser amplificada devido às condições iniciais nessa simulação, impossibilitando o controle pela rede neural.

Em uma primeira análise é possível identificar um padrão de comportamento em três regiões distintas do canal, como pode ser observado nas figuras 22, 23 e 24:

- Acima de 5 km distante do final do canal (*região 1*): no início a rede procura se estabilizar da condição inicial aproximando e alinhando ao máximo possível da linha central;
- Entre 5 km e 2 km distante do final do canal (*região 2*): nesse intervalo a rede busca se manter a um valor constante de distância da linha central conforme a aproximação que ela conseguiu no trecho anterior;
- Abaixo de 2 km distante do final do canal (*região 3*): a rede inicia uma manobra de entrada no trecho final do canal, que possui um aumento de largura, portanto, não é mais necessário manter o navio tão próximo da linha central. O objetivo principal se torna reduzir a velocidade para conseguir atracar com segurança e preparar a embarcação para a próxima manobra.

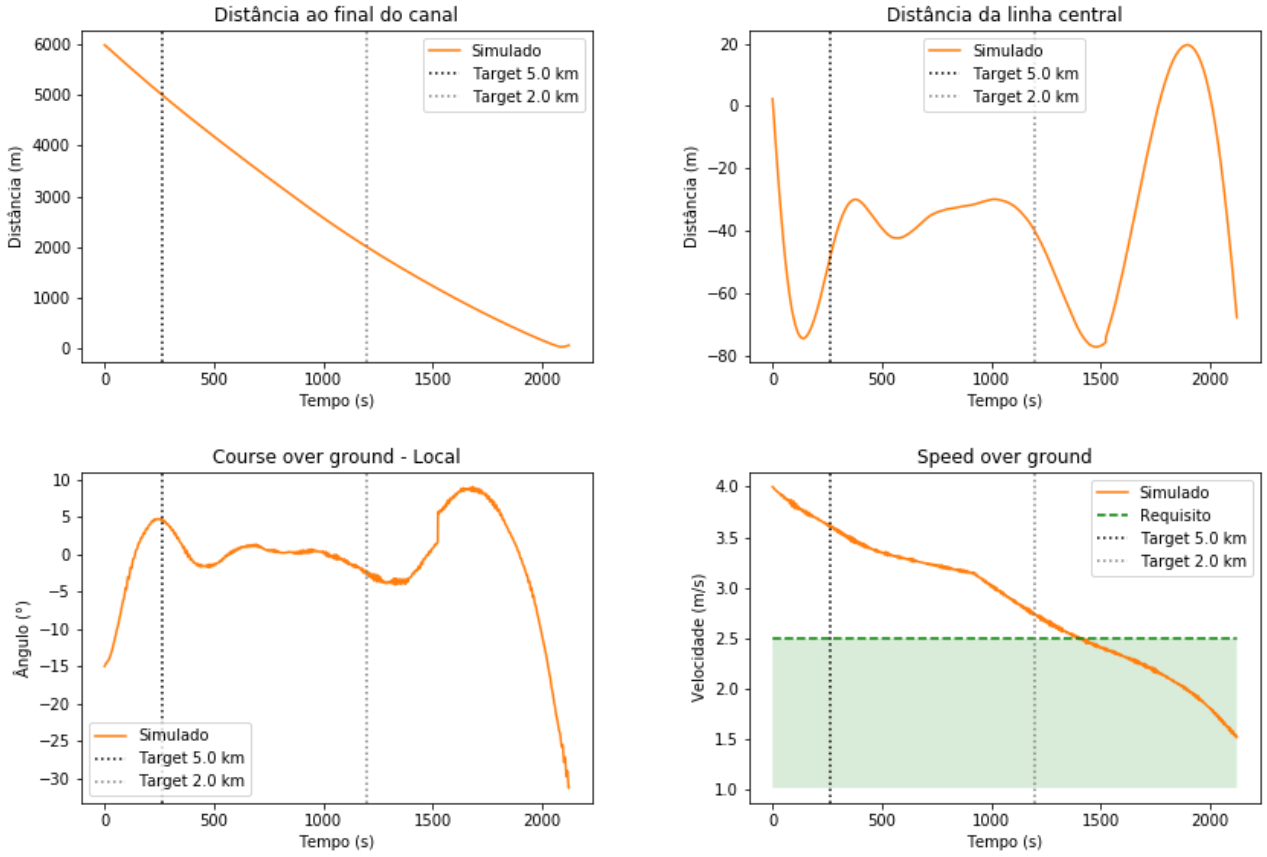
Por exemplo, utilizando a rede 1 na condição inicial 4 podem ser apresentados alguns gráficos de interesse como o da figura 25. As linhas verticais delimitam as regiões citadas anteriormente para facilitar a análise e podem ser verificadas no gráfico da distância ao final do canal. Para observar o alinhamento da embarcação, o parâmetro mais relevante é o *course over ground* (COG) local, ou seja, o ângulo do vetor velocidade resultante, sendo seu módulo denominado *speed over ground* (SOG), em relação a direção do canal, pois isso é o que garante a trajetória alinhada. Os padrões anteriores podem ser observados:

- *Região 1*: a embarcação se afasta inicialmente da linha central devido a condição inicial, mas procura se aproximar logo em seguida. O COG local apresenta um desvio

inicial que é corrigido tendendo a zero realizando o alinhamento;

- *Região 2*: a distância a linha central se mantém mais estável em aproximadamente 40 metros bombordo e o mesmo ocorre ao *COG* local próximo de 0 demonstrando o alinhamento;
- *Região 3*: ocorre o desalinhamento do *COG* local, uma explicação pode ser o fato de estar realizando uma manobra parecida com o da figura 10 em que é necessário realizar uma curva para entrada no berço voltado a bombordo. O *SOG* é reduzido para garantir a baixa velocidade dentro dos requisitos.

Figura 25 – Parâmetros de distância ao final do canal, da linha central, *couse over ground* local e *speed over ground* da simulação de controle da embarcação através da rede 1 na condição 4 onde o navio começa a simulação com um ângulo de aproamento não alinhado ao canal



Fonte: Autores

Em seguida, analisa-se o desempenho das redes nas simulações realizadas. Primeiramente, filtra-se os resultados entre a ocorrência ou não de colisão durante o trajeto. Segundo os dados da tabela 15, é possível observar que as redes 1 e 4 apresentaram um desempenho satisfatório ao passo que cada uma conseguiu conduzir a embarcação sem

colisão em 6 das 7 simulações realizadas. A partir desses resultados, a análise dos requisitos pode ser focada nos casos de sucesso.

Tabela 15 – Resultados gerais das simulações

	Rede 1	Rede 2	Rede 3	Rede 4
Sucesso	6	1	2	6
% sucesso	86%	14%	28%	86%

Fonte: Autores

9.2 Análise de requisitos

Pelos dados das simulações e segundo os requisitos da tabela 16 foram elaborados os gráficos das figuras 26, 27, 28 e 29 para a simulação das redes 1 e 4 na condição inicial 1. As informações para os demais casos podem ser observados no apêndice A.

Para avaliação do primeiro requisito, as figuras 26 e 28 apresentam uma região vermelha indicando um limite inferior de distância em relação à margem, 21 metros segundo a boca do Aframax, que não deve ser ultrapassado devido a iminência de uma colisão com a delimitação do canal.

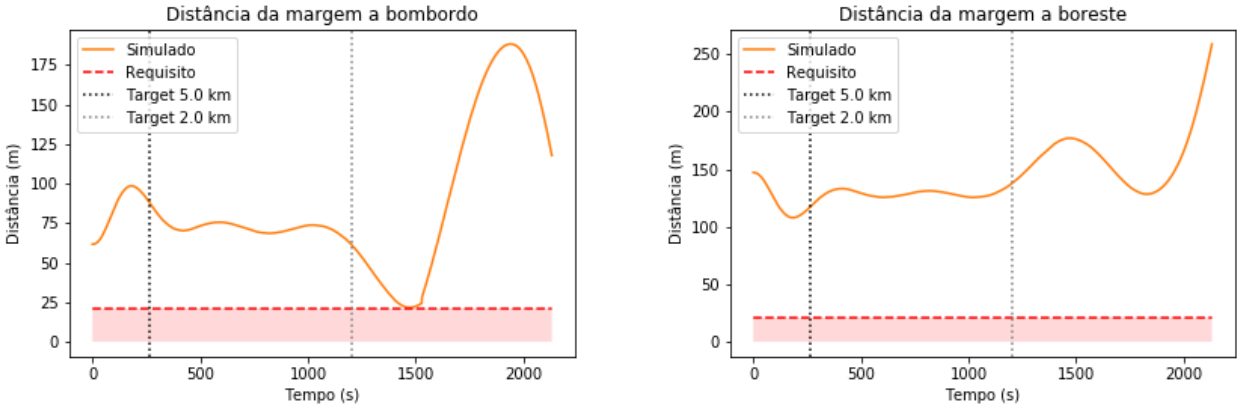
Enquanto as figuras 27 e 29 avaliam o segundo requisito apresentando uma região verde que determina a velocidade final que deve ser atingida pelo navio ao término da manobra para possibilitar seu atracamento em segurança.

Tabela 16 – Requisitos de projeto baseado na seção 1.2 e nas características do Aframax da tabela 1

Distância mínima às margens	Velocidade máxima ao final do canal
21 m	5.0 nós (2.5 m/s)

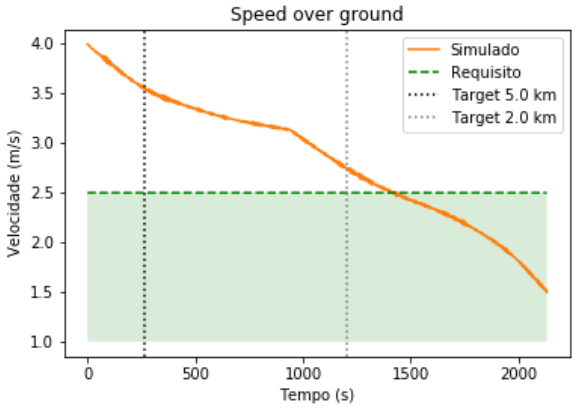
Fonte: Autores

Figura 26 – Gráfico das distâncias às margens durante a simulação da rede 1 na condição inicial 1. A linha vermelha representa o requisito de distância mínima (21 m)



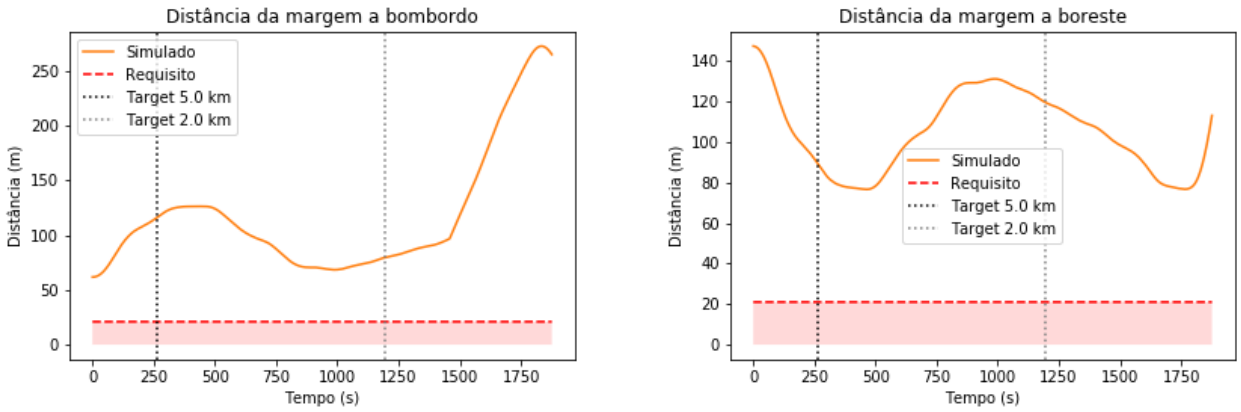
Fonte: Autores

Figura 27 – Gráfico de velocidade durante a simulação da rede 1 na condição inicial 1. A linha verde representa o limite máximo da velocidade no final do canal (2.5 m/s)



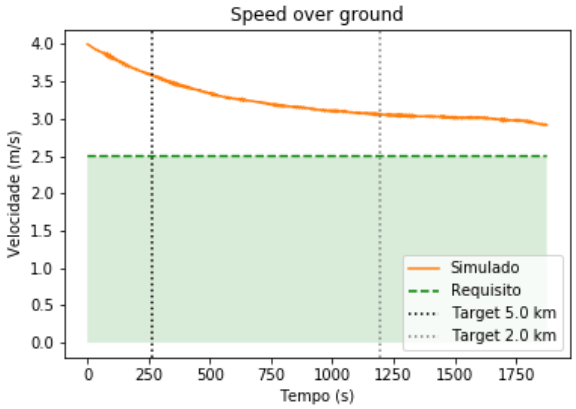
Fonte: Autores

Figura 28 – Gráfico das distâncias às margens durante a simulação da rede 4 na condição inicial 1. A linha vermelha representa o requisito de distância mínima (21 m)



Fonte: Autores

Figura 29 – Gráfico de velocidade durante a simulação da rede 4 na condição inicial 1. A linha verde representa o limite máximo da velocidade no final do canal (2.5 m/s)



Fonte: Autores

Da análise destas figuras e outras apresentadas no apêndice A, podem ser elaboradas as tabelas de consolidação dos resultados 17 e 18 que apresentam a adequação das redes neurais 1 e 4, respectivamente, aos requisitos de projeto.

Tabela 17 – Desempenho da rede 1 de acordo com os requisitos de distância às margens e velocidade

Rede 1	Requisitos		
Cond. Inicial	Distância	Velocidade	Final
1	✓	✓	✓
2	✓	✓	✓
3	✓	✓	✓
4	✓	✓	✓
7	✓	✓	✓
9	✓	✓	✓

Fonte: Autores

Tabela 18 – Desempenho da rede 4 de acordo com os requisitos de distância às margens e velocidade

Rede 4	Requisitos		
Cond. Inicial	Distância	Velocidade	Final
1	✓	✗	✗
2	✓	✓	✓
3	✓	✓	✓
4	✓	✗	✗
7	✓	✓	✓
9	✓	✓	✓

Fonte: Autores

Apesar de ambas as redes conseguirem conduzir a embarcação até o final do canal sem colisão, analisando os desempenhos das mesmas perante os requisitos, nota-se que a rede 1 atende aos requisitos em todas as simulações, enquanto a rede 4 não atende o requisito de velocidade máxima ao final do canal em duas condições iniciais (1 e 4), ou seja, a embarcação chega no final do canal com velocidade acima de 5.0 nós (2.5 m/s). Para exemplificar tal diferença, pode-se ver pelas figuras 27 e 29 que no final do percurso a rede 1 consegue entrar na área verde que significa que a mesma atende o requisito de velocidade, enquanto a rede 4 fica acima da mesma área. Quanto ao requisito de distâncias às margens, ambas as redes conseguem manter a embarcação dentro da região proposta pelo requisito (figuras 26 e 28). Com essas informações disponíveis, é possível afirmar que a rede neural 1 é a que apresenta os melhores resultados considerando os requisitos estabelecidos.

9.3 Próximos passos

Para uma aplicação real de um controlador de embarcação qualquer, não apenas por meio de redes neurais, é muito importante que ela possa ser utilizada em situações genéricas, seja em canais com estruturas diferentes ou mesmo condições ambientais mais severas. Desta maneira, listamos a seguir algumas tarefas que podem ser realizadas para melhorar a performance e ampliar o escopo de aplicação da solução obtida:

- Generalização do controlador: ampliar a base de dados para simulações de outros portos e considerar a condição ambiental e os parâmetros físicos da embarcação como parâmetros de entrada;
- Melhoria de performance: utilizar outros modelos de arquitetura, hiperparâmetros e metodologias para definir a rede neural;
- Integração completa com os simuladores do TPN: isso permitiria mais ferramentas de avaliação como a interface gráfica de visão 360º, sugestões de comandos aos práticos e aprendizado contínuo toda vez que fosse realizada uma simulação.

10 Conclusão

Nesse trabalho foi possível aprender sobre o desenvolvimento de redes neurais e as peculiaridades envolvidas para garantir uma boa performance, por exemplo, a importância de realizar um tratamento e uma análise prévia da base de dados a fim de garantir sua qualidade e, consequentemente, uma boa aderência.

O processo de definição da melhor arquitetura e dos melhores hiperparâmetros depende muito da experiência e do conhecimento técnico do projetista, mas também da realização de testes e validações devido ao caráter iterativo de seu desenvolvimento. Para o problema proposto de navegação em um canal de acesso, os dados de treinamento são sequências temporais, e os melhores resultados foram obtidos por meio do uso de redes neurais recorrentes, fato este explicado devido ao modelo possuir uma memória que guarda a relação entre os estados.

O trabalho também abordou um processo de validação pelo uso de simulação por meio do integrador numérico *Dyna* que inclui a implementação de diversos fatores que influenciam a manobrabilidade da embarcação e torna a solução obtida muito mais compatível com a realidade.

Em suma, os resultados obtidos neste trabalho demonstram que as redes neurais recorrentes são soluções interessantes para automatizar o processo de condução de um navio em um canal de acesso ao apresentar diversas simulações sem colisão e, inclusive, dentro dos requisitos propostos. Entretanto, ainda são necessários trabalhos futuros de melhoria para que possam ser generalizadas e aplicáveis em embarcações reais.

Referências

AHMED, Y. A.; HASEGAWA, K. Automatic ship berthing using artificial neural network trained by consistent teaching data using nonlinear programming method. Engineering Applications of Artificial Intelligence, v. 26, n. 10, p. 2287–2304, 2013. ISSN 0952-1976. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0952197613001668⟩. 8, 21, 35, 38, 46, 52, 53

ALTHOFF, M.; MERGEL, A. Comparison of Markov Chain Abstraction and Monte Carlo Simulation for the Safety Assessment of Autonomous Cars. IEEE Transactions on Intelligent Transportation Systems, v. 12, n. 4, p. 1237–1247, 12 2011. ISSN 1524-9050. Disponível em: ⟨http://ieeexplore.ieee.org/document/5875884/⟩. 18

BARBER, D. Bayesian Reasoning and Machine Learning. [S.l.]: Cambridge University Press, 2012. 22

BEHESHTI, I.; DEMIREL, H.; MATSUDA, H. Classification of Alzheimer's disease and prediction of mild cognitive impairment-to-Alzheimer's conversion from structural magnetic resource imaging using feature ranking and a genetic algorithm. Computers in Biology and Medicine, v. 83, p. 109–119, 2017. ISSN 0010-4825. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0010482517300483⟩. 20

BELCHER, P. A sociological interpretation of the COLREGS. Journal of Navigation, Cambridge University Press, v. 55, n. 02, p. 213–224, 5 2002. ISSN 0373-4633. Disponível em: ⟨http://www.journals.cambridge.org/abstract_S0373463302001686⟩. 19

CHANGHAU, I. LSTM and GRU – Formula Summary. 2017. Disponível em: ⟨https://isaacchanghau.github.io/post/lstm-gru-formula/⟩. 31

DEVORE, J. L. Probability and statistics for engineering and the sciences. 8th. ed. [S.l.: s.n.], 2012. 22

FOSSEN, T. I. A survey on Nonlinear Ship Control: from Theory to Practice. IFAC Proceedings Volumes, Elsevier, v. 33, n. 21, p. 1–16, 8 2000. ISSN 1474-6670. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1474667017370441⟩. 19

IM, N.-K.; NGUYEN, V.-S. Artificial neural network controller for automatic ship berthing using head-up coordinate system. International Journal of Naval Architecture and Ocean Engineering, 2017. ISSN 2092-6782. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S2092678216304368⟩. 21

JAMES, G. et al. An Introduction to Statistical Learning: With Applications in R. [S.l.]: Springer Publishing Company, Incorporated, 2014. ISBN 1461471370, 9781461471370. 29

KRUTHIVENTI, S. S. S.; AYUSH, K.; BABU, R. V. DeepFix: A Fully Convolutional Neural Network for Predicting Human Eye Fixations. IEEE Transactions on Image Processing, v. 26, n. 9, p. 4446–4456, 2017. ISSN 1057-7149. 20

LEE, S.-M.; KWON, K.-Y.; JOH, J.
A fuzzy logic for autonomous navigation of marine vehicle satisfying COLREG guidelines. [S.l.: s.n.], 2004. v. 2. 20

LEE, Y.-i.; KIM, Y.-G. A Collision Avoidance System for Autonomous Ship Using Fuzzy Relational Products and COLREGs. In: YANG, Z. R.; YIN, H.; EVERSON, R. M. (Ed.). Intelligent Data Engineering and Automated Learning – IDEAL 2004. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 247–252. ISBN 978-3-540-28651-6. 20

LIN, P. Why Ethics Matters for Autonomous Cars. Springer, Berlin, Heidelberg, p. 69–85, 2016. Disponível em: ⟨https://link.springer.com/chapter/10.1007/978-3-662-48847-8_4⟩. 18

MAKRIDAKIS, S. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. Futures, v. 90, p. 46–60, 2017. ISSN 0016-3287. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0016328717300046⟩. 20

NG, A. Deep Learning Specialization. 2018. Disponível em: ⟨https://www.coursera.org/specializations/deep-learning⟩. 23, 25, 26, 30, 31, 33, 36

PADEN, B. et al. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. IEEE Transactions on Intelligent Vehicles, v. 1, n. 1, p. 33–55, 3 2016. ISSN 2379-8904. Disponível em: ⟨http://ieeexplore.ieee.org/document/7490340/⟩. 18

PERERA, L.; CARVALHO, J.; SOARES, C. G. Decision making system for the collision avoidance of marine vessel navigation based on COLREGs rules and regulations. 10 2009. 19

ROBERTS, G. et al. Intelligent ship autopilots—A historical perspective. Mechatronics, Pergamon, v. 13, n. 10, p. 1091–1103, 12 2003. ISSN 0957-4158. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0957415803000448?via%3Dihub⟩. 20

RUSSELL, S.; NORVIG, P. Inteligencia Artificial. [S.l.: s.n.], 2004. ISBN 85-352-1177-2. 23

SATO, Y.; ISHII, H. Study of a collision-avoidance system for ships. Control Engineering Practice, Pergamon, v. 6, n. 9, p. 1141–1149, 9 1998. ISSN 0967-0661. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0967066198001075⟩. 19

SHAHIN, M. A. State-of-the-art review of some artificial intelligence applications in pile foundations. Geoscience Frontiers, v. 7, n. 1, p. 33–44, 2016. ISSN 1674-9871. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1674987114001327⟩. 20

SOUZA JR., J. R. de et al. Development and application of a ship manoeuvring digital simulator for restricted waters. IFAC Proceedings Volumes, Elsevier, v. 42, n. 18, p. 32–37, 1 2009. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1474667016318675⟩. 21

STATHEROS, T.; HOWELLS, G.; MAIER, K. M. Autonomous Ship Collision Avoidance Navigation Concepts, Technologies and Techniques. The Journal of Navigation, Cambridge University Press, v. 61, n. 1, p. 129–142, 2008. ISSN 1469-7785. Disponível em: ⟨https://www.cambridge.org/core/journals/journal-of-navigation/article/autonomous-ship-collision-avoidance-navigation-concepts-technologies-and-techniques/3F3ED13DEFF7B84B5B0A758BEBF47ADD⟩. 19

SUAPE. Guia Portuário. 2018. Disponível em: ⟨http://www.suape.pe.gov.br/pt/porto/infraestrutura-portuaria/guia-portuario⟩. 38

SUAPE. O que é Suape. 2018. Disponível em: ⟨http://www.suape.pe.gov.br/pt/institucional/o-que-e-suape⟩. 37

TANNURI, E. A. et al. Dynamic positioning systems: An experimental analysis of sliding mode control. Control Engineering Practice, v. 18, n. 10, p. 1121–1132, 2010. ISSN 0967-0661. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0967066110001486⟩. 20

TPN-USP. Centro de Simulações. 2017. Disponível em: ⟨http://tpn.usp.br/simuladores/⟩. 36

TRANSPETRO. Tipos de navios. 2018. Disponível em: ⟨http://www.transpetro.com.br/pt_br/areas-de-negocios/transporte-maritimo/tipos-de-navios.html⟩. 37

WAN, Y.; SI, Y.-W. Adaptive neuro fuzzy inference system for chart pattern matching in financial time series. Applied Soft Computing, v. 57, p. 1–18, 2017. ISSN 1568-4946. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1568494617301473⟩. 20

Apêndice A – Resultados dos testes

Neste apêndice serão apresentados os resultados obtidos nos modelos das redes neurais recorrentes da tabela 6 para as condições iniciais das tabelas 7, 8 e 9. Foi com base nestes resultados que as observações da seção 8.3 foram elaboradas.

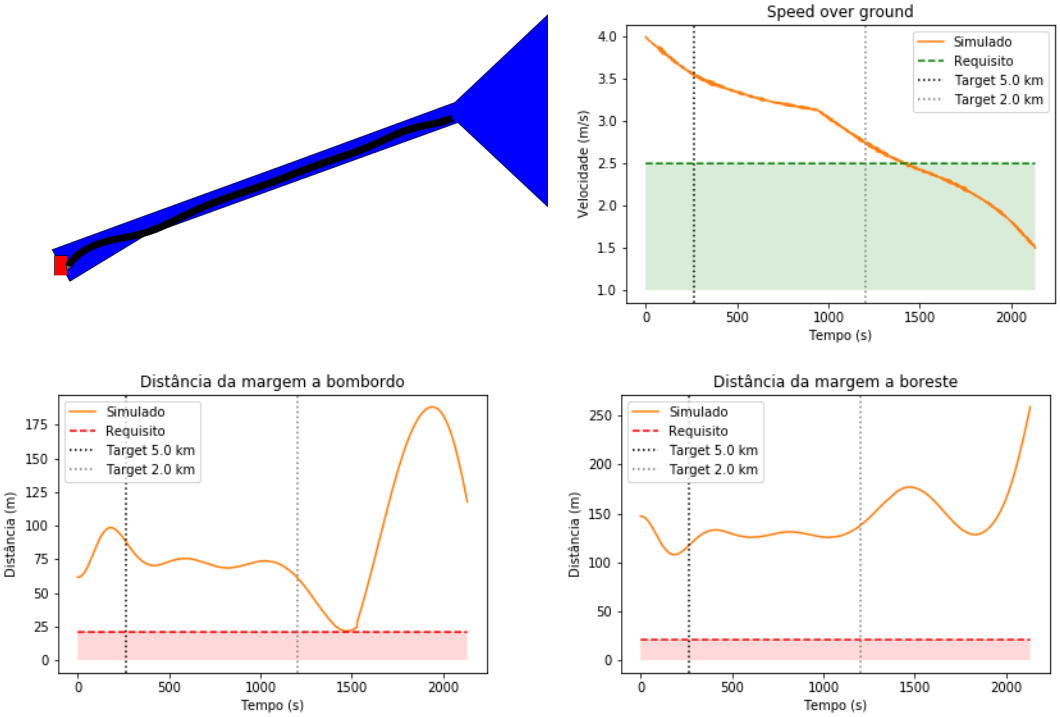
Em cada simulação serão apresentados 2 conjuntos de dados: a trajetória da embarcação no canal (superior esquerdo) e os parâmetros para avaliação dos requisitos. Esses parâmetro são compostos pelo o módulo da velocidade da embarcação também denominado *speed over ground* (superior direito) e as distâncias da embarcação às margens bombordo e boreste (inferior esquerdo e direito, respectivamente).

Vale ressaltar algumas observações importantes. Primeiramente, na imagem da trajetória, a embarcação não está na mesma escala e com o mesmo formato que o seu modelo real, pois esta figura serve apenas para acompanhar seu percurso no canal durante a simulação.

Portanto, a avaliação mais criteriosa dos resultados deve ser realizada nos parâmetros apresentados. No gráfico do *speed over ground*, a região verde representa o valor aceitável ao final da manobra. Nos gráficos das distâncias, a região vermelha representa um valor que nunca deve ocorrer durante a simulação, pois, caso seja ultrapassado, a embarcação estaria na iminência de uma saída do canal. A forma como esses valores foram definidos está descrito em mais detalhes na seção 1.2. Finalmente, as linhas verticais, quando presentes, indicam o valor de δ_t, em preto o valor de 5 quilômetros e em cinza de 2 quilômetros, isso é útil para identificar em qual trecho do canal o navio estava em cada instante de tempo, auxiliando na análise da manobra.

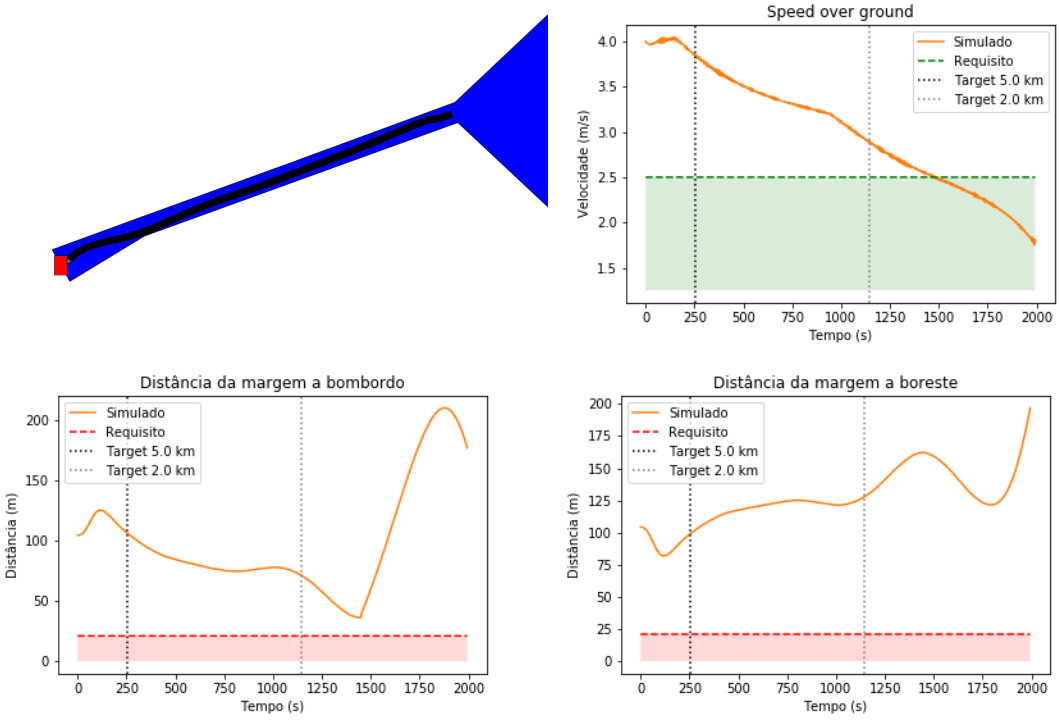
A.1 Rede 1

Figura 30 – Manobra realizada pela rede 1 na condição inicial 1



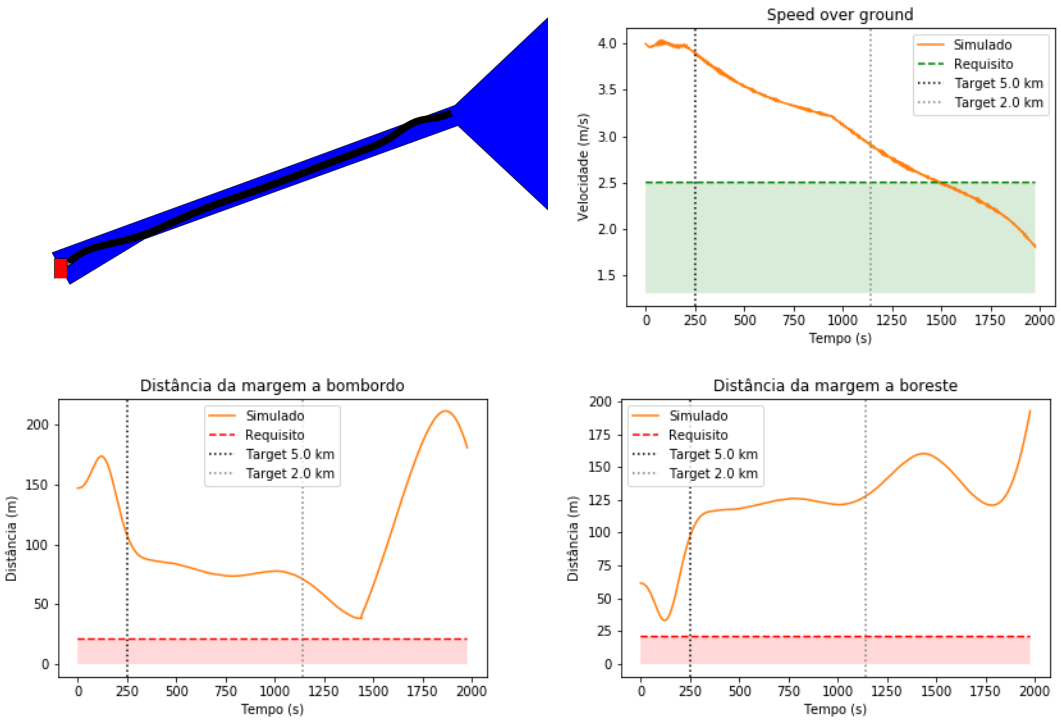
Fonte: Autores

Figura 31 – Manobra realizada pela rede 1 na condição inicial 2



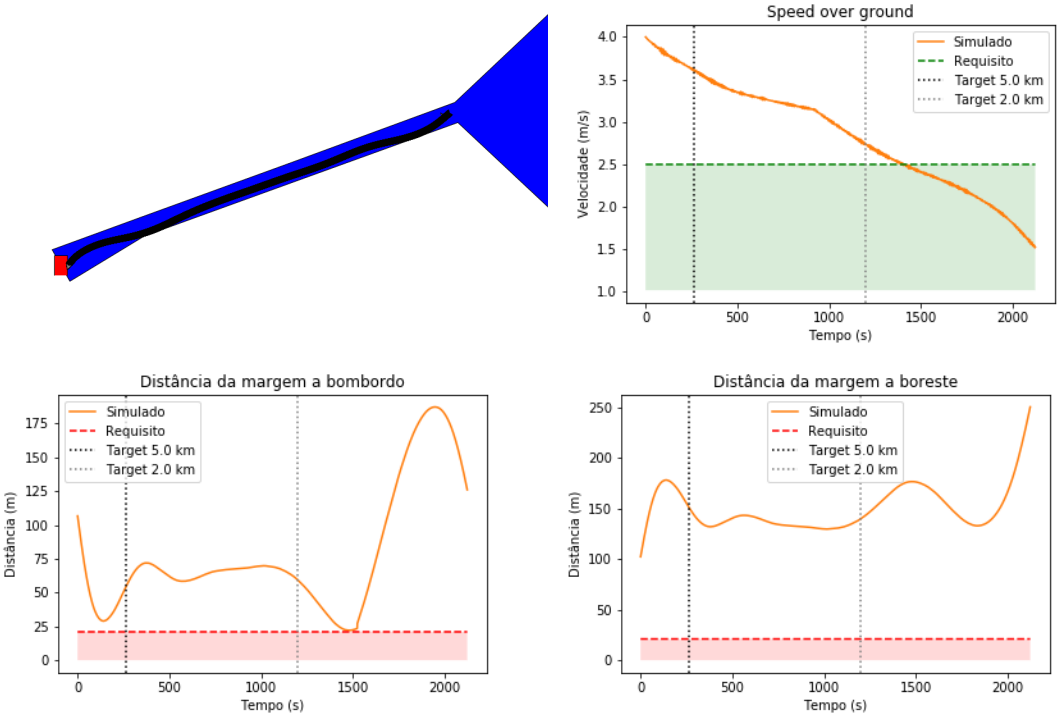
Fonte: Autores

Figura 32 – Manobra realizada pela rede 1 na condição inicial 3



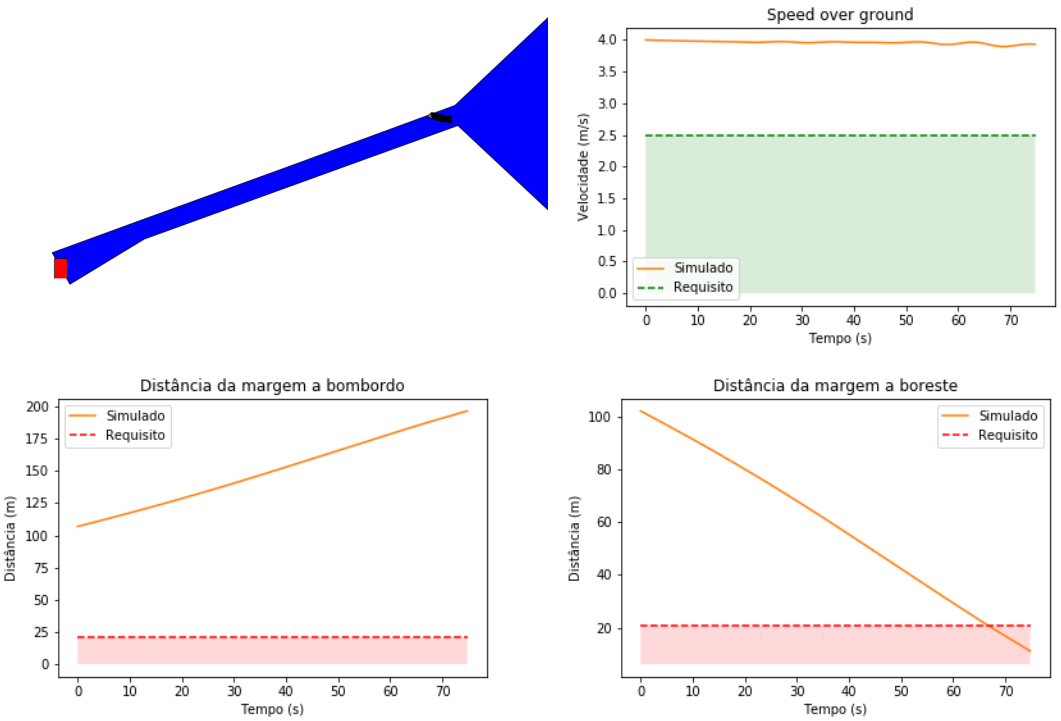
Fonte: Autores

Figura 33 – Manobra realizada pela rede 1 na condição inicial 4



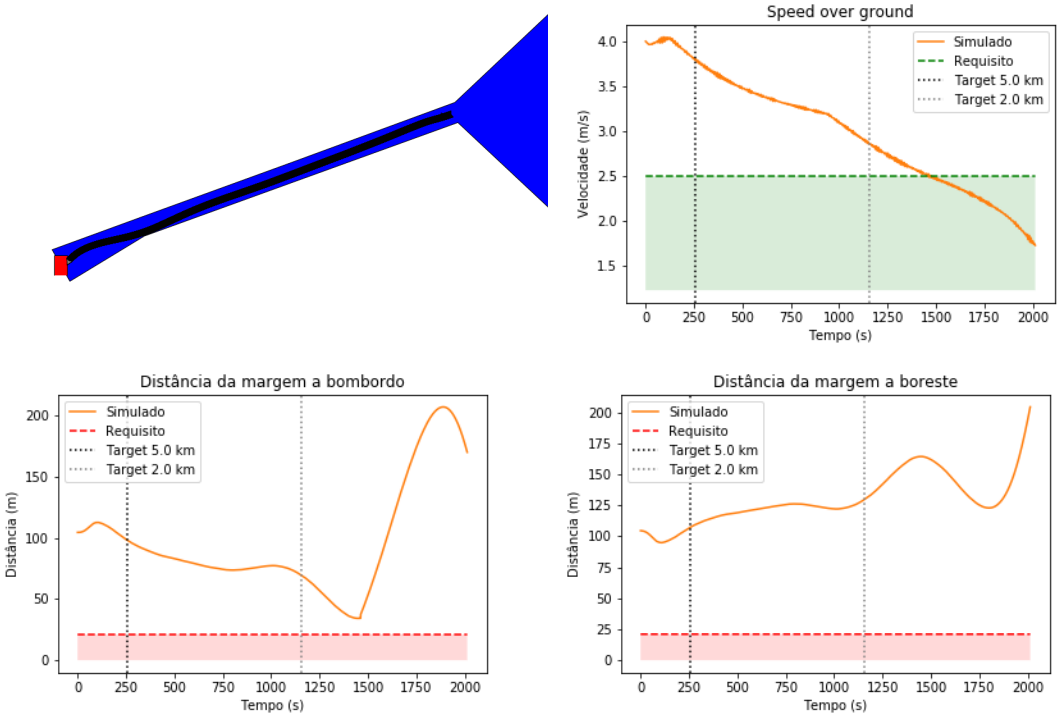
Fonte: Autores

Figura 34 – Manobra realizada pela rede 1 na condição inicial 6



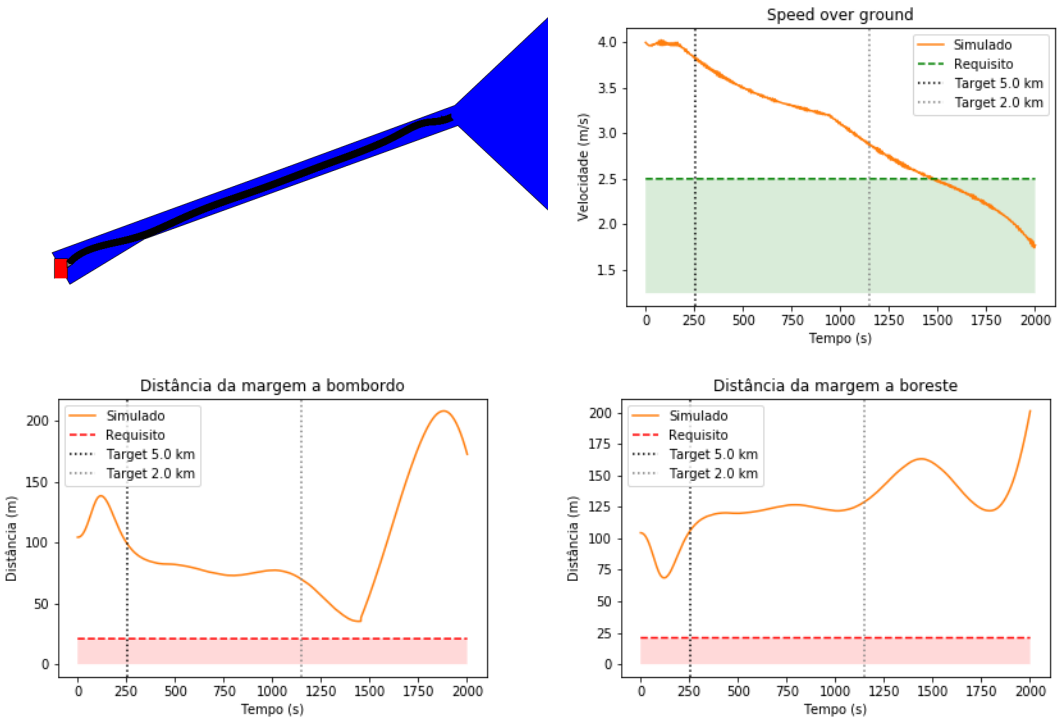
Fonte: Autores

Figura 35 – Manobra realizada pela rede 1 na condição inicial 7



Fonte: Autores

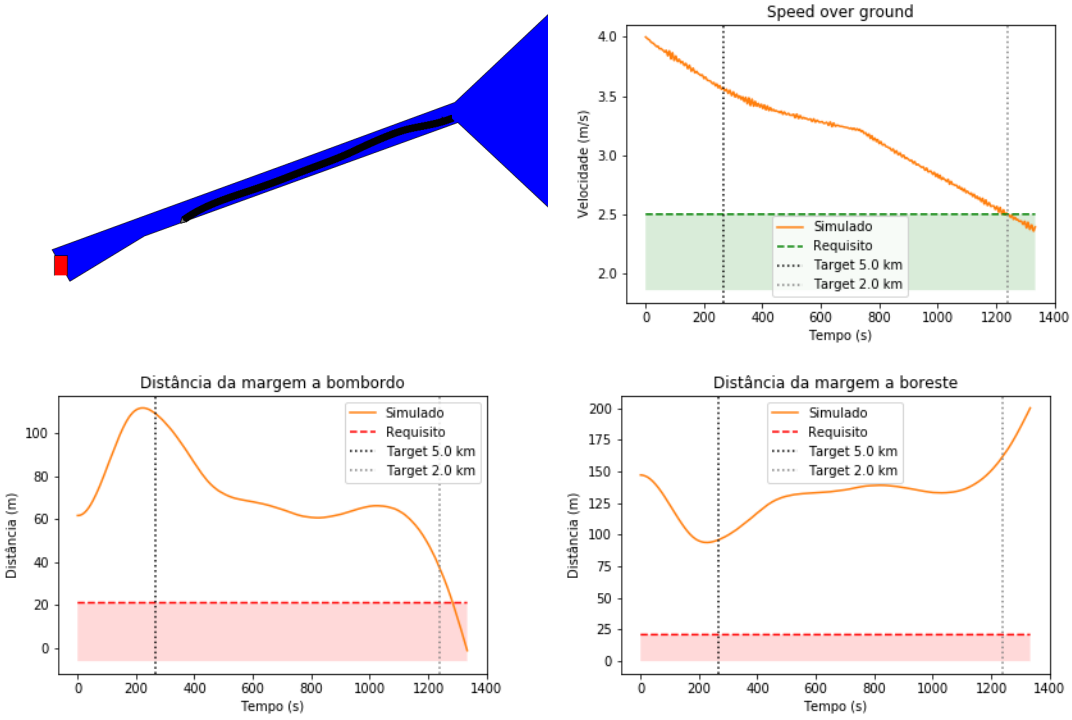
Figura 36 – Manobra realizada pela rede 1 na condição inicial 9



Fonte: Autores

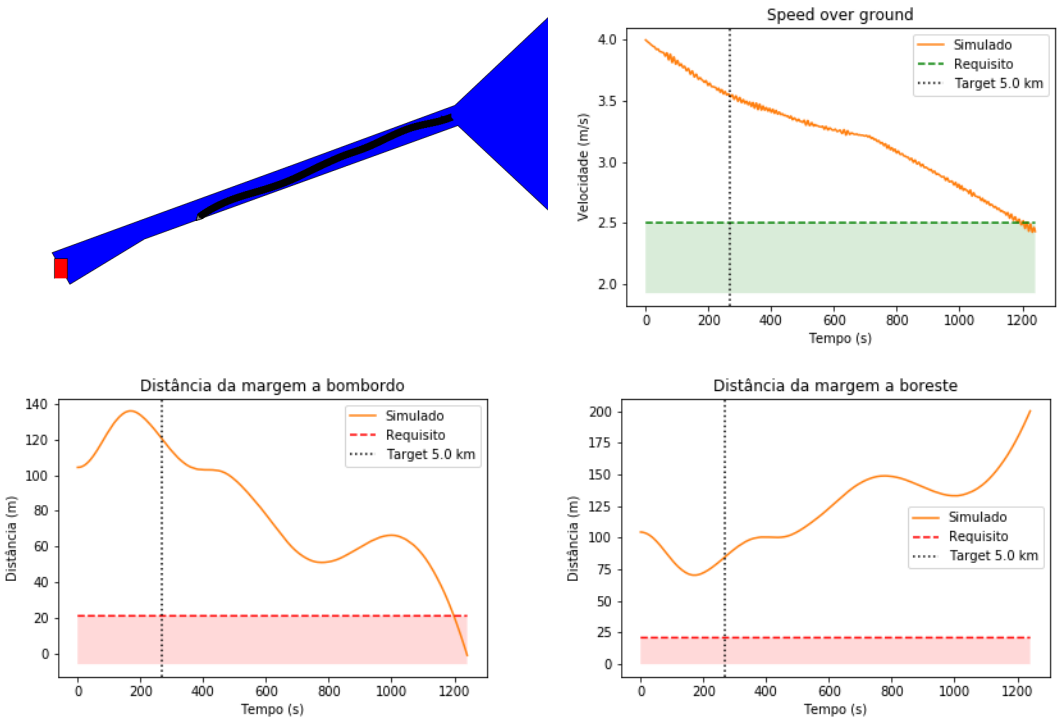
A.2 Rede 2

Figura 37 – Manobra realizada pela rede 2 na condição inicial 1



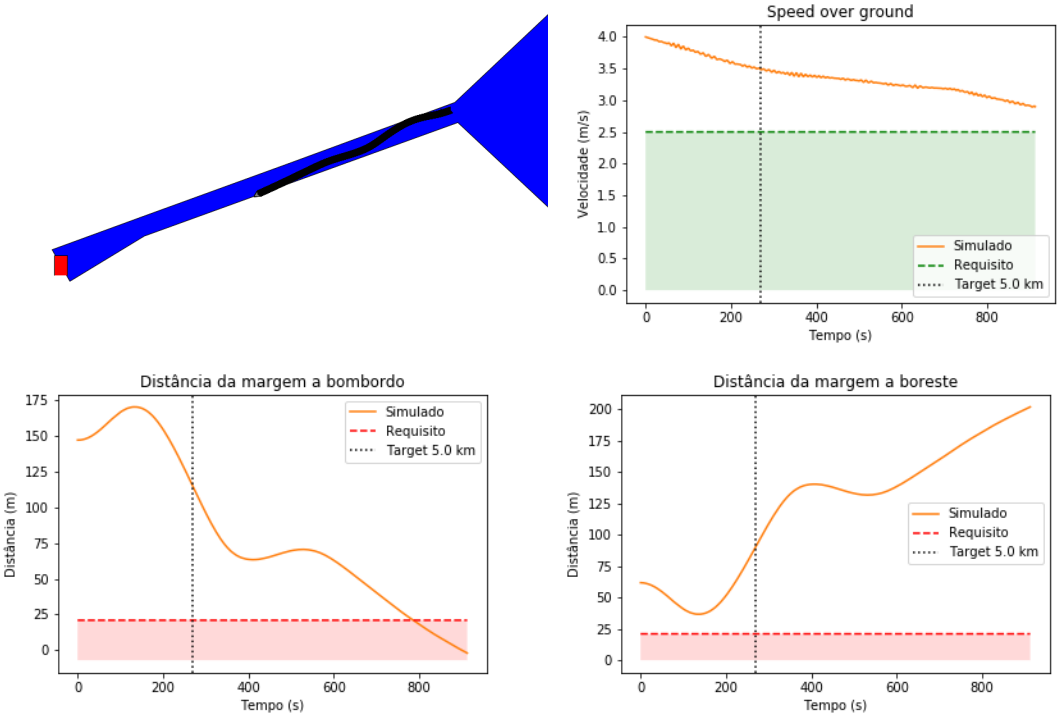
Fonte: Autores

Figura 38 – Manobra realizada pela rede 2 na condição inicial 2



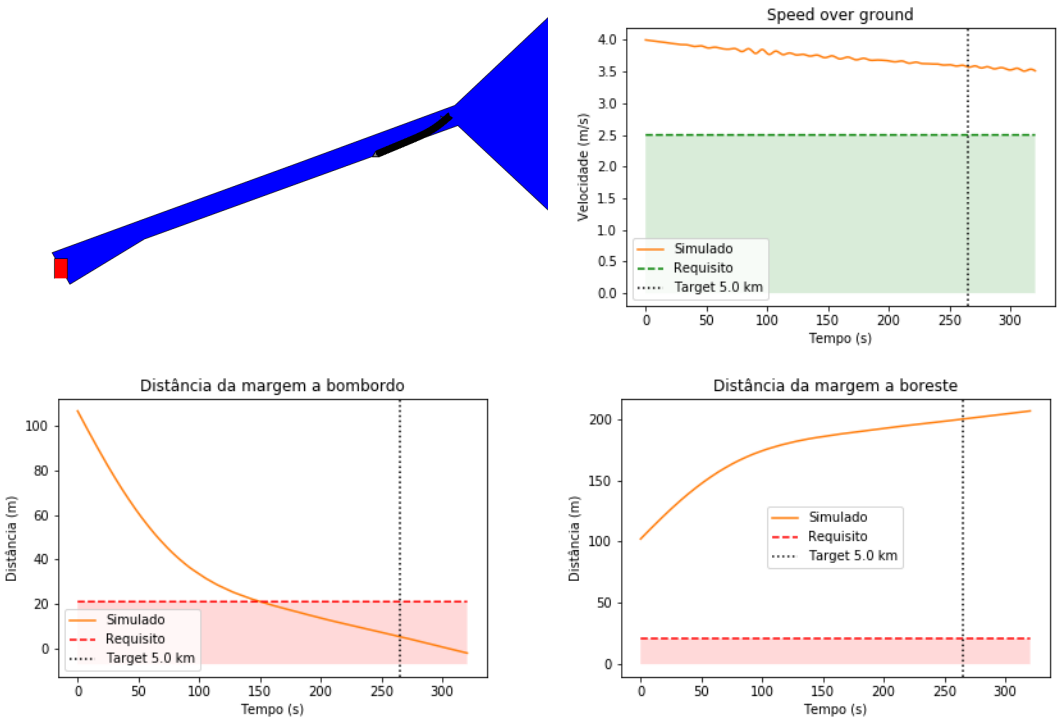
Fonte: Autores

Figura 39 – Manobra realizada pela rede 2 na condição inicial 3



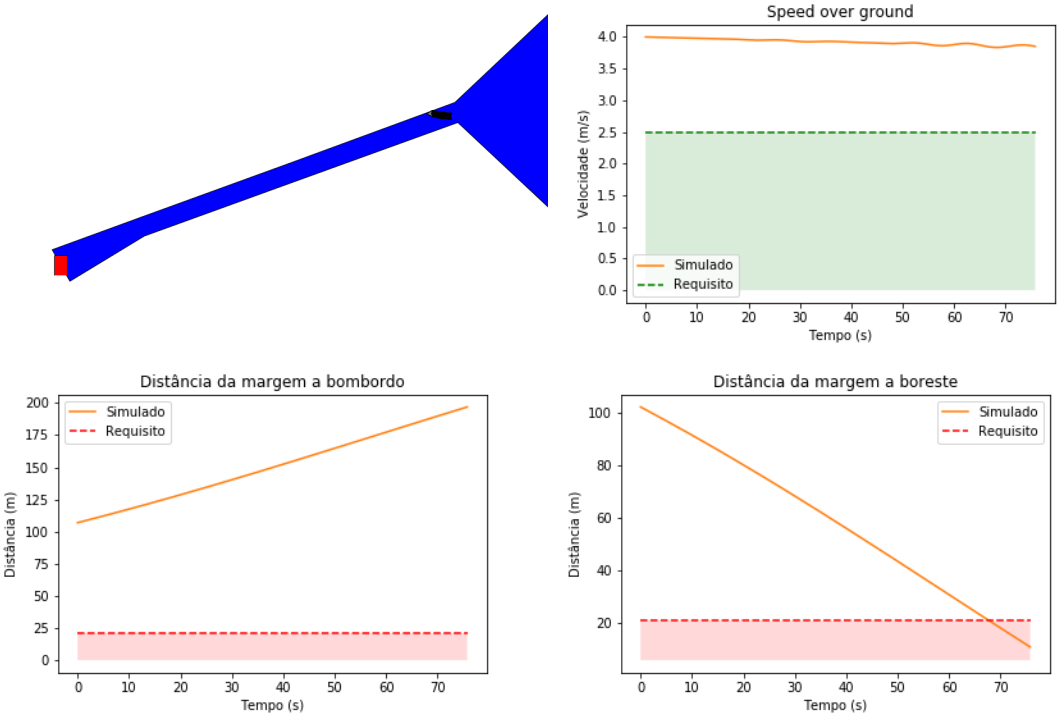
Fonte: Autores

Figura 40 – Manobra realizada pela rede 2 na condição inicial 4



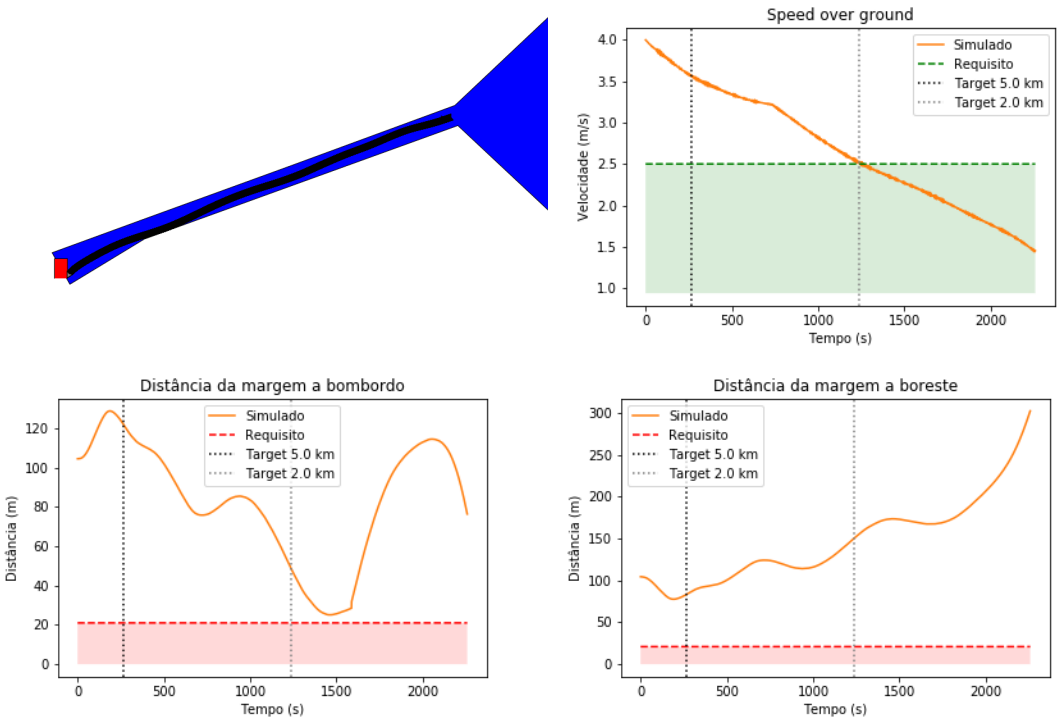
Fonte: Autores

Figura 41 – Manobra realizada pela rede 2 na condição inicial 6



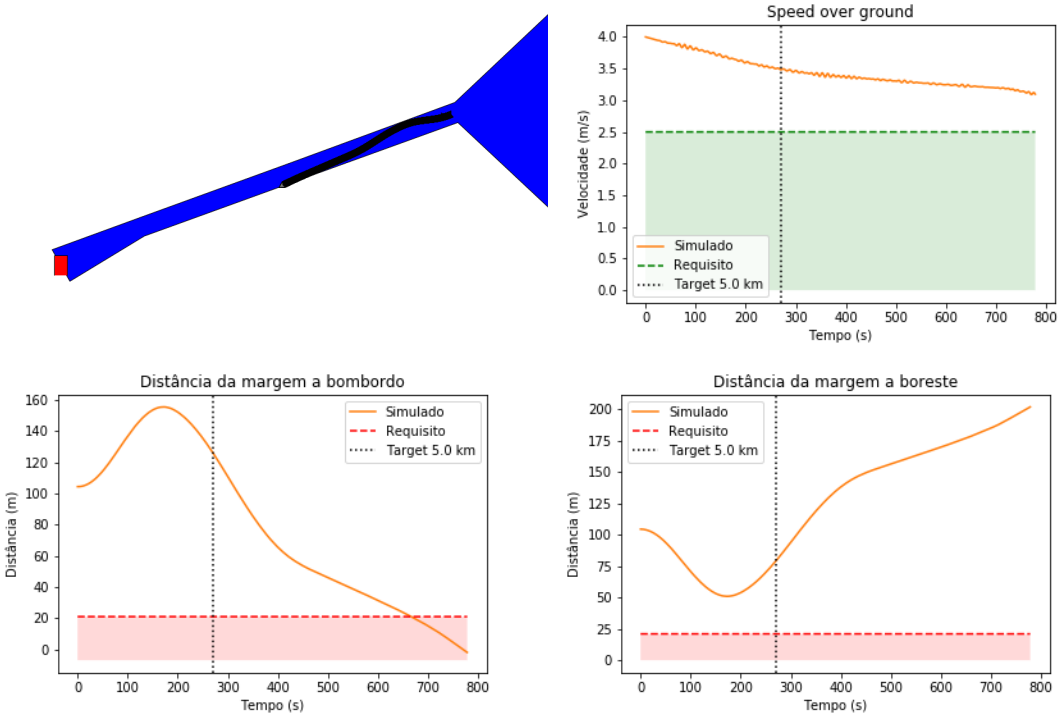
Fonte: Autores

Figura 42 – Manobra realizada pela rede 2 na condição inicial 7



Fonte: Autores

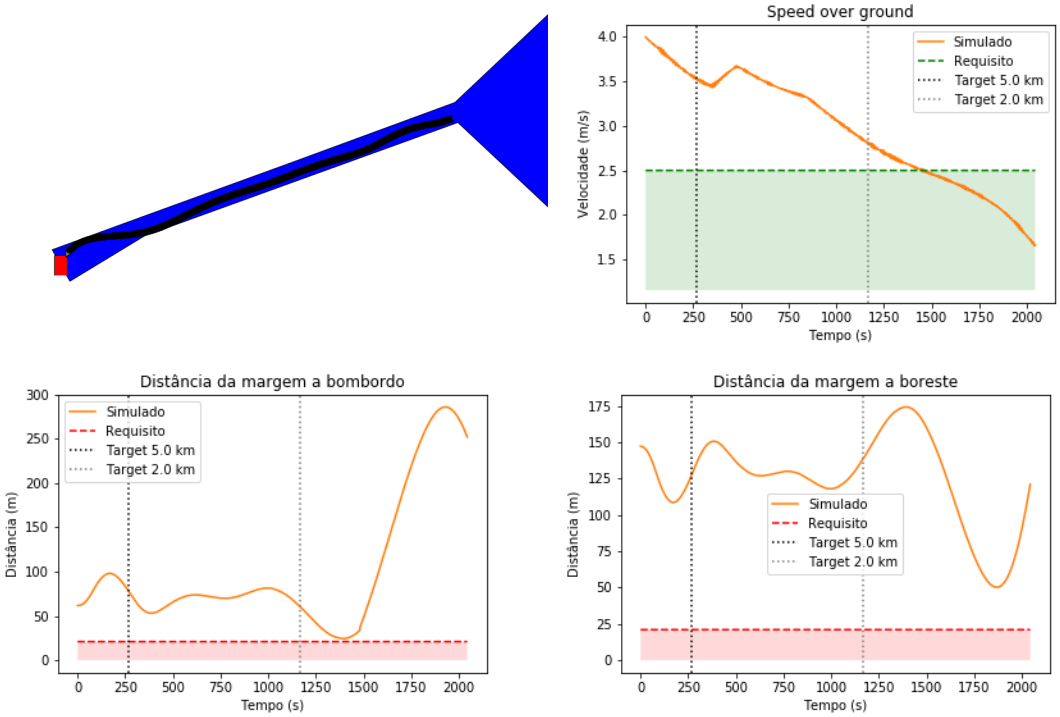
Figura 43 – Manobra realizada pela rede 2 na condição inicial 9



Fonte: Autores

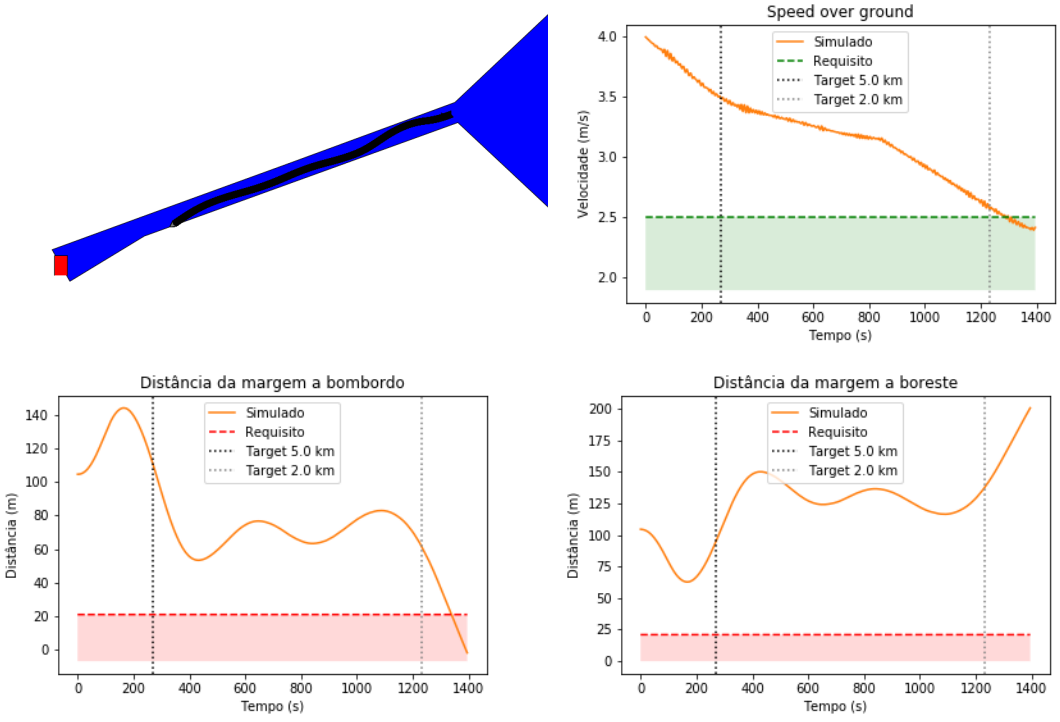
A.3 Rede 3

Figura 44 – Manobra realizada pela rede 3 na condição inicial 1



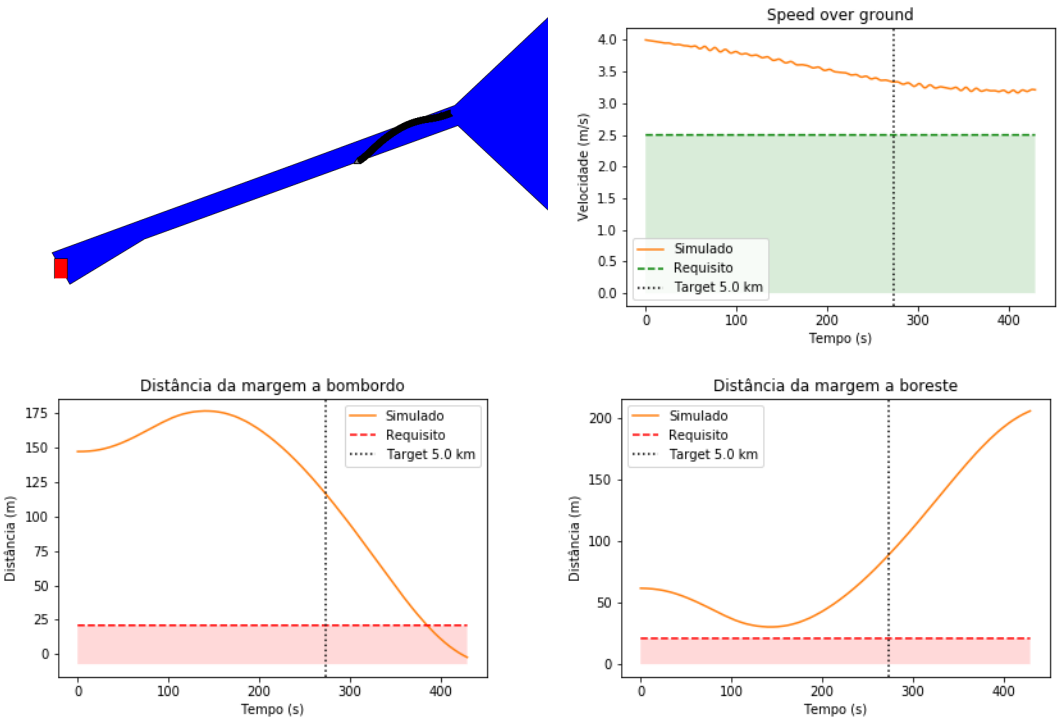
Fonte: Autores

Figura 45 – Manobra realizada pela rede 3 na condição inicial 2



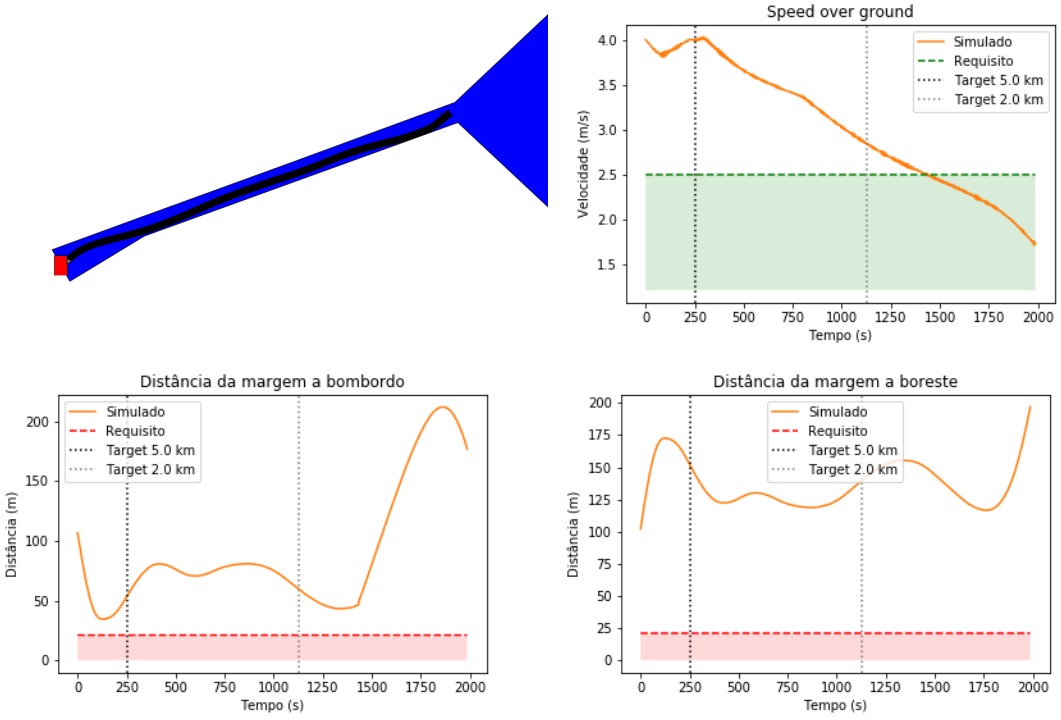
Fonte: Autores

Figura 46 – Manobra realizada pela rede 3 na condição inicial 3



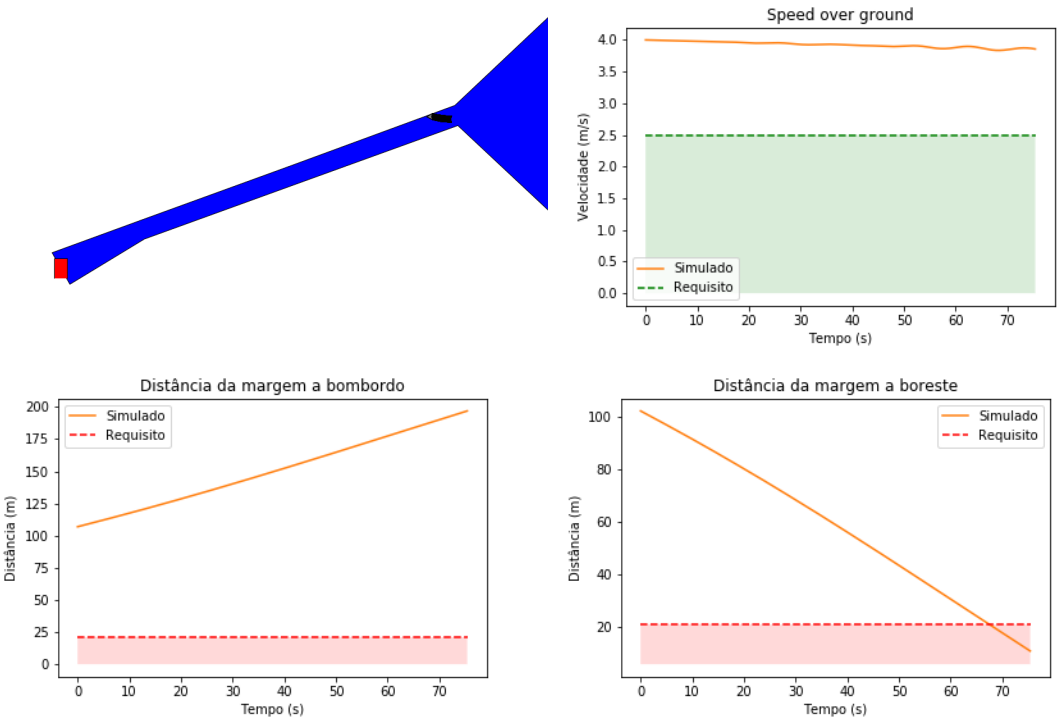
Fonte: Autores

Figura 47 – Manobra realizada pela rede 3 na condição inicial 4



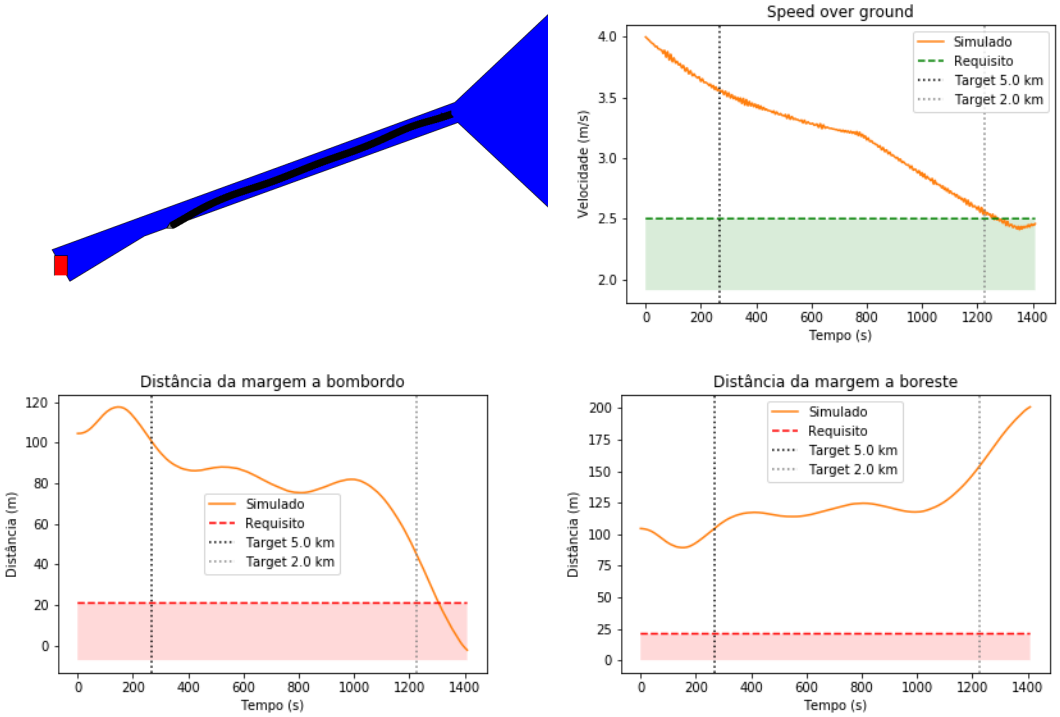
Fonte: Autores

Figura 48 – Manobra realizada pela rede 3 na condição inicial 6



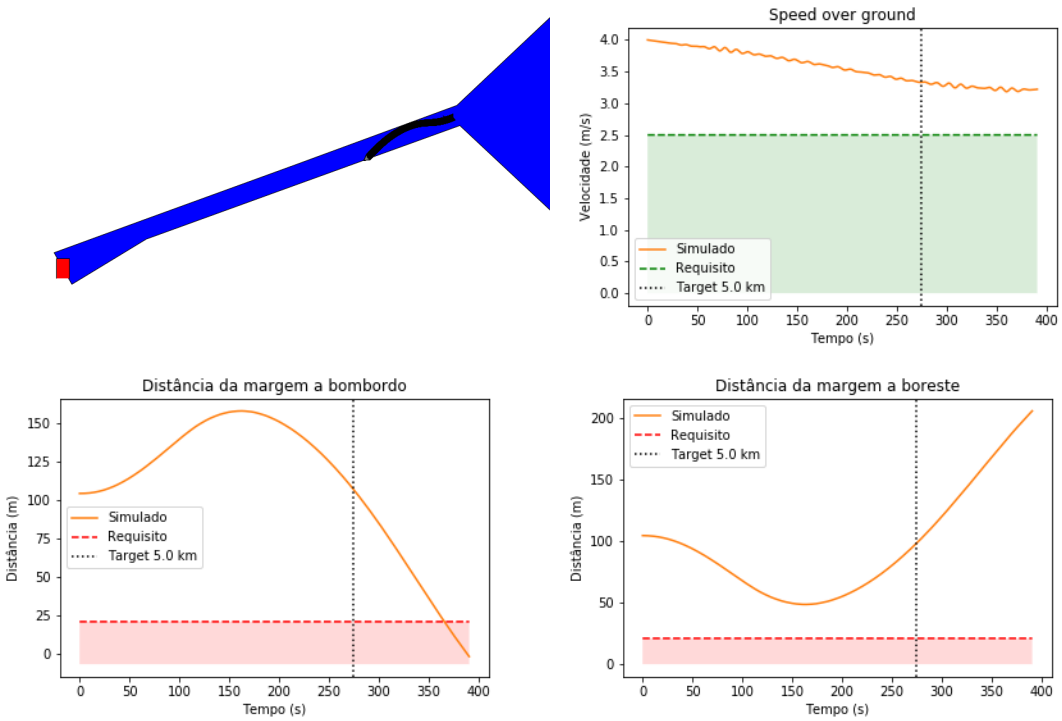
Fonte: Autores

Figura 49 – Manobra realizada pela rede 3 na condição inicial 7



Fonte: Autores

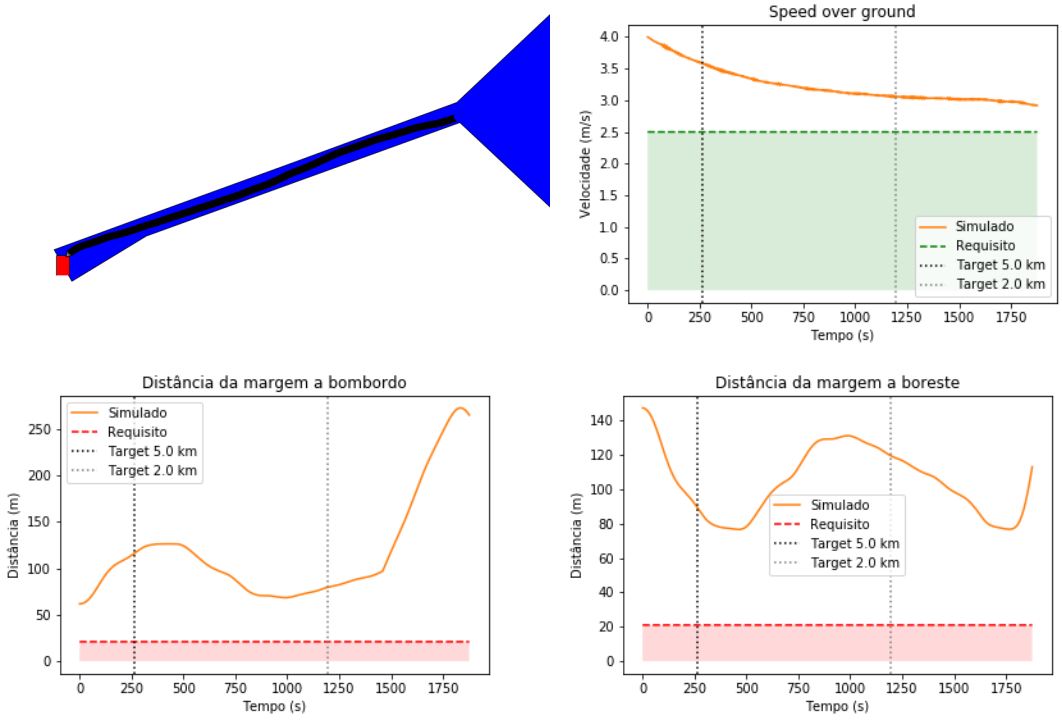
Figura 50 – Manobra realizada pela rede 3 na condição inicial 9



Fonte: Autores

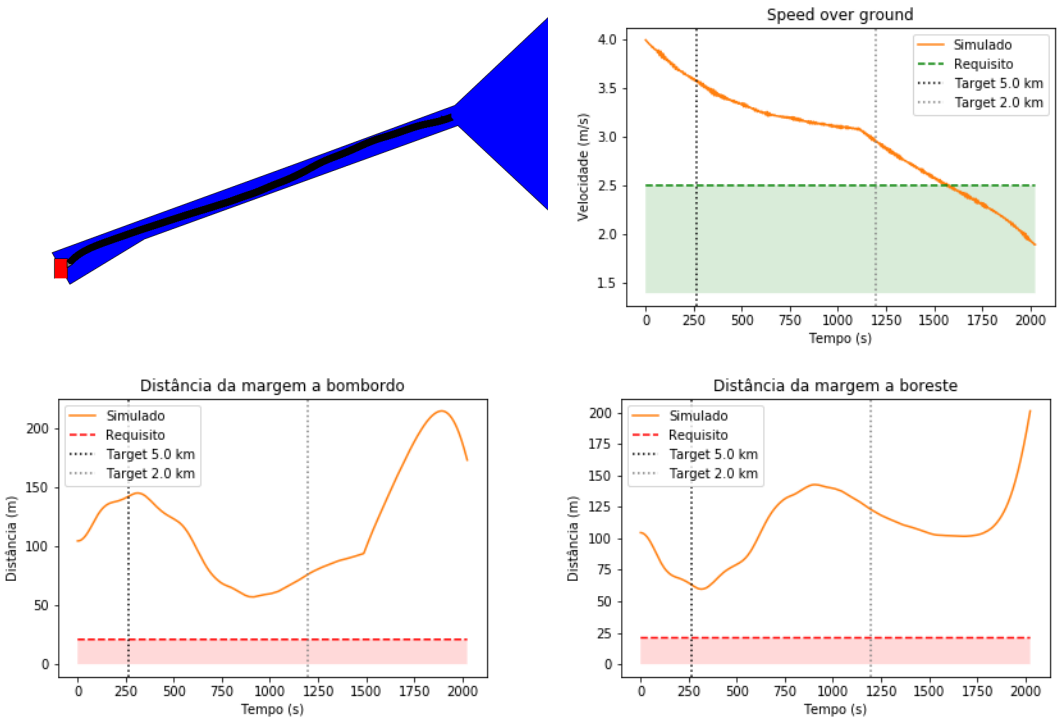
A.4 Rede 4

Figura 51 – Manobra realizada pela rede 4 na condição inicial 1



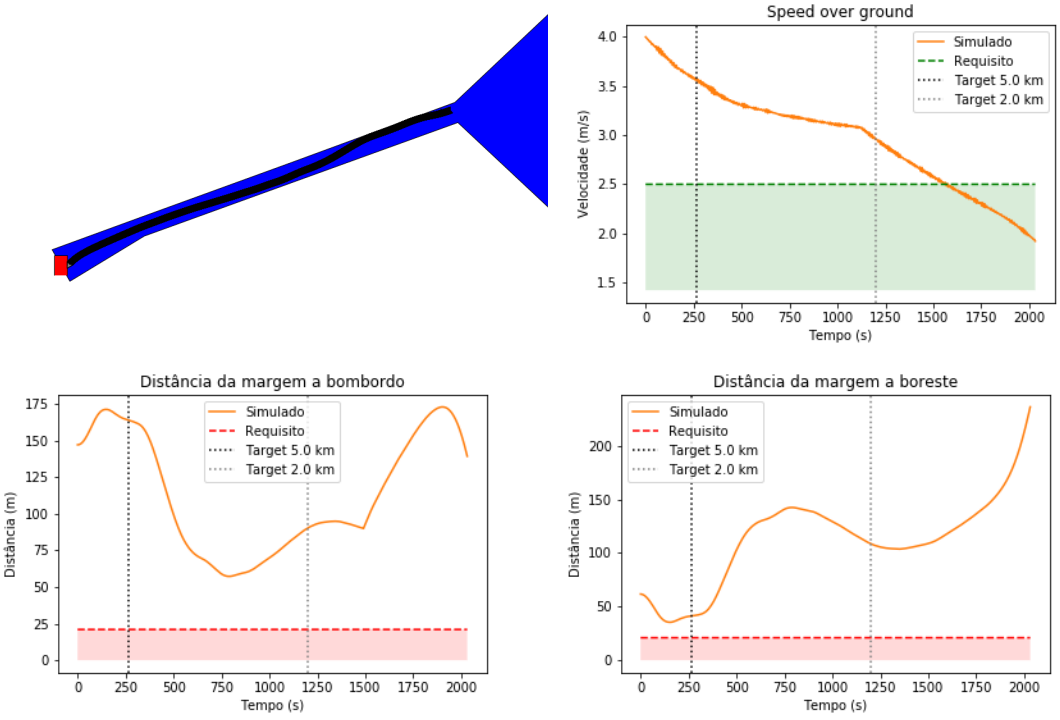
Fonte: Autores

Figura 52 – Manobra realizada pela rede 4 na condição inicial 2



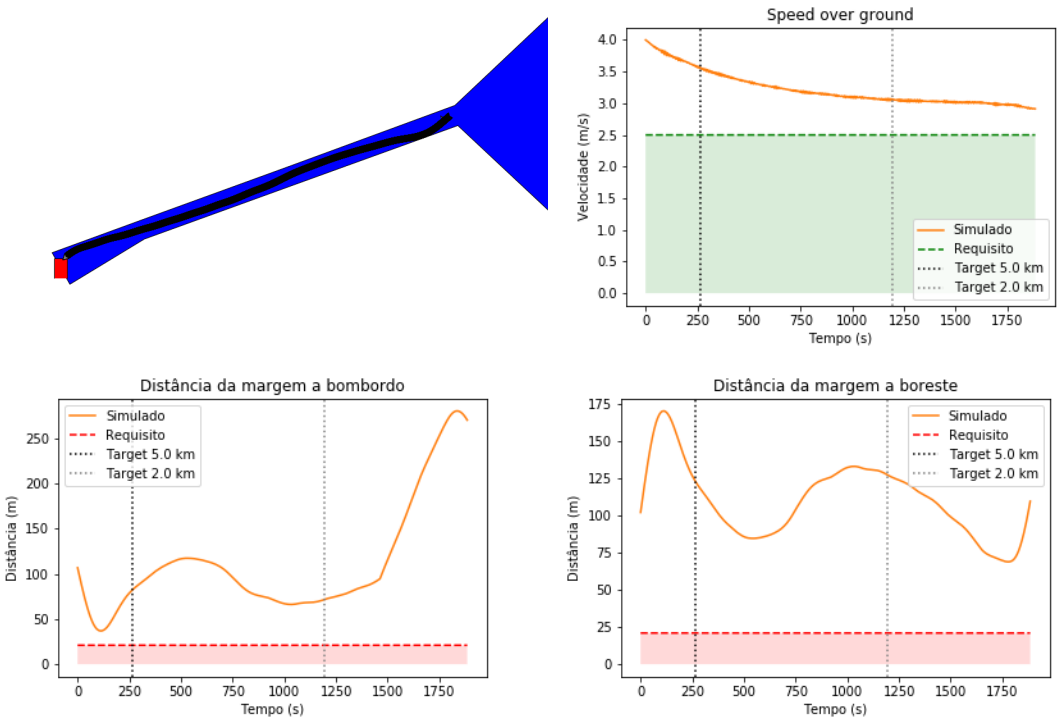
Fonte: Autores

Figura 53 – Manobra realizada pela rede 4 na condição inicial 3



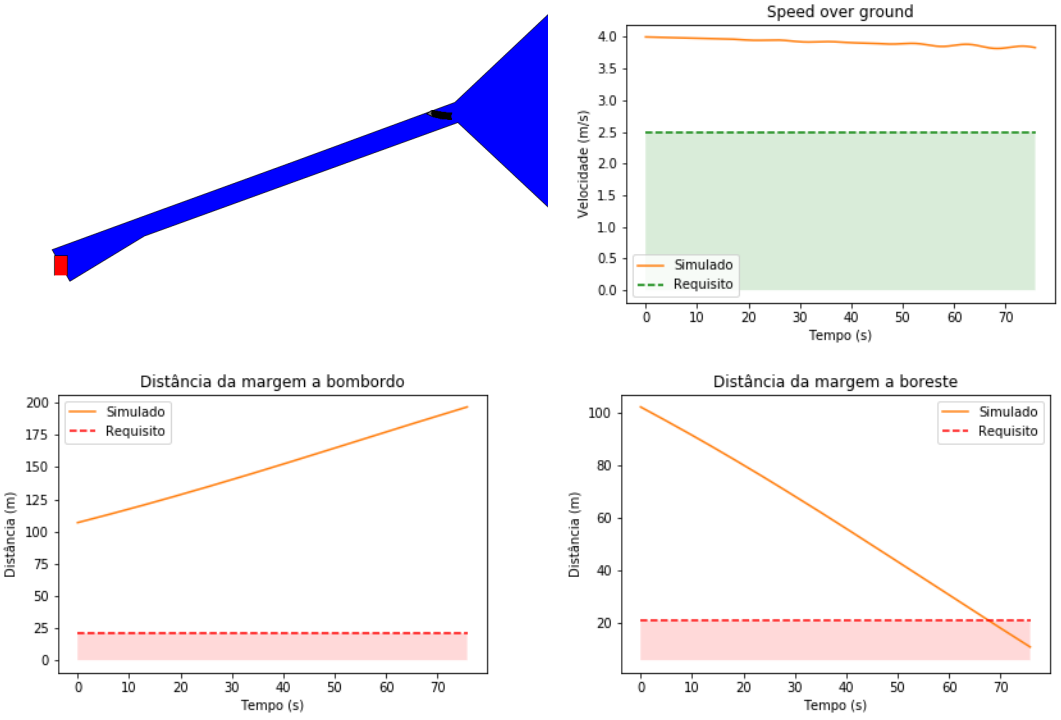
Fonte: Autores

Figura 54 – Manobra realizada pela rede 4 na condição inicial 4



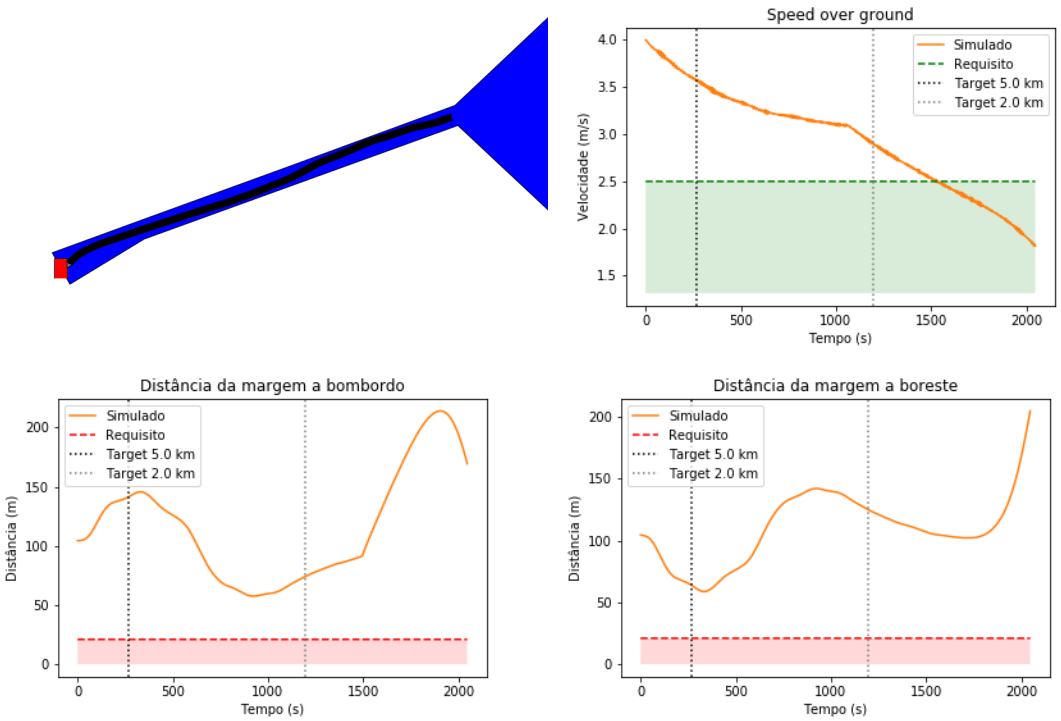
Fonte: Autores

Figura 55 – Manobra realizada pela rede 4 na condição inicial 6



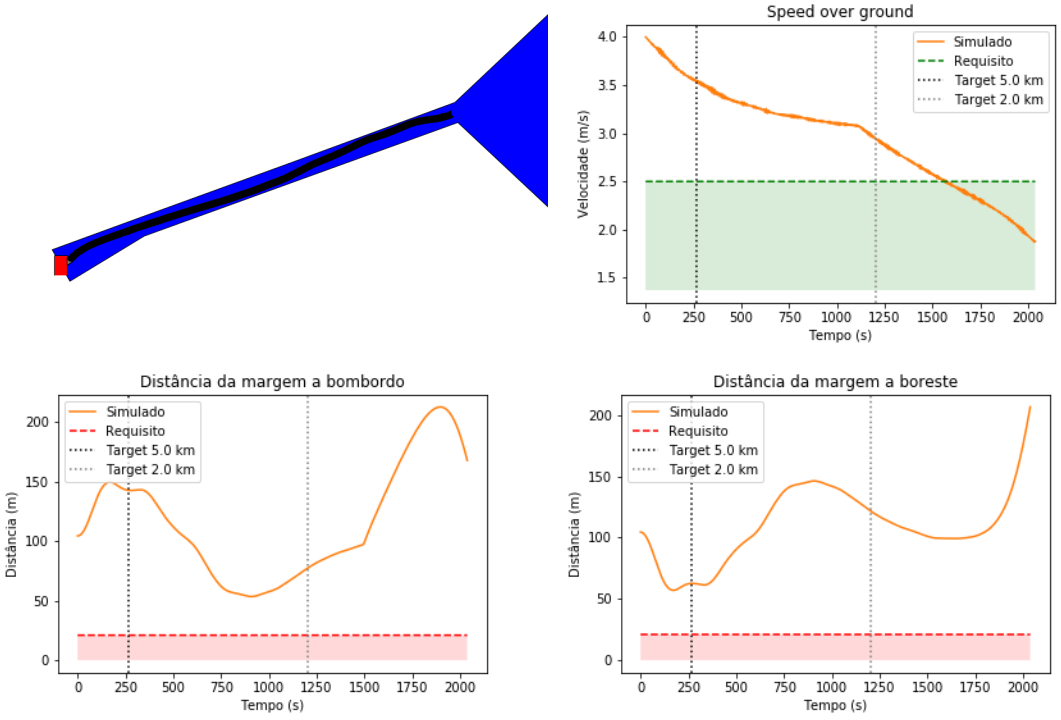
Fonte: Autores

Figura 56 – Manobra realizada pela rede 4 na condição inicial 7



Fonte: Autores

Figura 57 – Manobra realizada pela rede 4 na condição inicial 9



Fonte: Autores