UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS - FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

Lucas Teixeira Oliveira

Eficácia e acurácia de emuladores em modelos eletrofisiológicos cardíacos simulados

Juiz de Fora

2025

Lucas Teixeira Oliveira

Eficácia e acurácia de emuladores em modelos eletrofisiológicos cardíacos simulados

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Modelagem Computacional. Área de concentração: Modelagem Computacional.

Orientador: Prof. Dr. Rodrigo Weber dos Santos

Coorientador: Prof. Dr. Joventino de Oliveira Campos

Juiz de Fora

2025

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Teixeira Oliveira, Lucas.
Eficácia e acurácia de emuladores em modelos eletrofisiológicos cardíacos simulados / Lucas Teixeira Oliveira. -- 2025.
120 p.

Orientador: Rodrigo Weber dos Santos
Coorientador: Joventino de Oliveira Campos
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós-Graduação em Modelagem Computacional, 2025.

1. Modelo eletrofisiológico cardíaco . 2. Emulador. 3. Processo Gaussiano. 4. Expansão em Caos Polinomial. 5. Redes Neurais . I. Weber dos Santos, Rodrigo, orient. II. de Oliveira Campos, Joventino, coorient. III. Título.

Lucas Teixeira Oliveira

Eficácia e acurácia de emuladores em modelos eletrofisiológicos cardíacos simulados

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Modelagem Computacional. Área de concentração: Modelagem Computacional.

Aprovada em 09 de setembro de 2025.

BANCA EXAMINADORA

Prof. Dr. Rodrigo Weber dos Santos - Orientador

Universidade Federal de Juiz de Fora

Prof. Dr. Joventino de Oliveira Campos

Universidade Federal de Juiz de Fora

Prof. Dr. Bernardo Martins Rocha

Universidade Federal de Juiz de Fora

Prof. Dr. Rafael Sachetto Oliveira

Universidade Federal de São João del Rei

Juiz de Fora, 08/09/2025.

Documento assinado eletronicamente por **Rodrigo Weber dos Santos**, **Professor(a)**, em 19/09/2025, às 15:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rafael Sachetto Oliveira**, **Usuário Externo**, em 15/10/2025, às 11:44, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Bernardo Martins Rocha**, **Professor(a)**, em 15/10/2025, às 12:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Joventino de Oliveira Campos**, **Professor(a)**, em 15/10/2025, às 19:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **2606163** e o código CRC **3E80CF46**.

Aos meus pais, José Emílio e Maria Imaculada, à minha irmã Alice e à minha namorada Lara.

AGRADECIMENTOS

A Deus, pai de infinita bondade, por me guiar e proteger.

Ao Santo Antônio, pela proteção, guarda e devoção.

Aos meus pais, José Emílio Zanzirolani de Oliveira e Maria Imaculada Evangelista Teixeira Oliveira e minha irmã, Alice Teixeira Oliveira, pelos ensinamentos, pelo incentivo, pela paciência, pelo apoio e pelo amor, vocês sempre supriram minhas necessidades e com quem divido essa conquista.

À Lara dos Reis Gamonal Gonçalves, companheira de tantas jornadas e tempos memoráveis, por me observar e seguir com passos firmes ao meu lado.

Aos meus entes queridos, em especial, tios, tias, primos, primas, avós, sogro, sogra, por fazerem parte de minha genealogia e refletirem em mim esperanças.

Aos meus amigos, pela convivência, pelas risadas, pelos futebóis, pelas corridas e pelas pedaladas, em especial ao Davi Boratto. Foi bom ter irmãos.

Aos professores do curso, especialmente ao orientador Rodrigo Weber dos Santos e ao coorientador Joventino de Oliveira Campos, pelos ensinamentos, pelo apoio e por acreditarem que daria certo.

Aos colegas do Programa de Pós-graduação em Modelagem Computacional, em especial ao Alexsandro, Mayra e Yan pela prontidão, auxílio e disponibilidade.

À Universidade Federal de Juiz de Fora, pelo ensino público, gratuito, de qualidade e pela oportunidade formativa concedida e experiências incríveis.

Ao Programa de Pós-graduação Computacional e ao Laboratório de Fisiologia Computacional pela oportunidade de me qualificar.

À Levty, por escolher apoiar e permitir adquirir e empregar conhecimentos.

Às agências brasileiras CAPES, CNPq, FAPEMIG, FINEP e à UFJF, pelo suporte financeiro concedido, e ao LNCC, pelo apoio com a utilização do supercomputador Santos Dumont.

A vida é uma sucessão de eventos aleatórios e o acaso determina o caminho. (O autor, 2025)

RESUMO

Doenças cardíacas são comuns nos dias atuais, e a necessidade de estudos sobre a atividade eletrofisiológica tem sido atendida por meio de simulações computacionais que auxiliam no diagnóstico médico. O uso de ferramentas estatísticas, como a Quantificação de Incertezas (UQ) e a Análise de Sensibilidade (SA), orienta o desenvolvimento de modelos computacionais ao identificar os parâmetros mais influentes. Entretanto, essas ferramentas demandam uma grande quantidade de dados para produzir resultados confiáveis. Devido ao elevado custo computacional das simulações, emuladores surgem como alternativa para gerar dados suficientes e viabilizar a aplicação de UQ e SA. O objetivo deste trabalho foi analisar três emuladores, Expansão em Caos Polinomial (PCE), Processos Gaussianos (GP) e Redes Neurais (NN), quanto à sua eficácia e acurácia ao treinar e predizer valores de dois modelos eletrofisiológicos baseados no modelo de ten Tusscher (2004). Inicialmente, investigaram-se diferentes kernels para os GPs, com o intuito de selecionar os mais adequados e compará-los aos demais emuladores. Posteriormente, os métodos foram avaliados quanto ao erro quadrático médio (MSE) e aos tempos de treinamento e de inferência, sendo atribuída uma nota normalizada obtida pela média dessas métricas. De modo geral, os resultados mostram que as Redes Neurais apresentaram o melhor equilíbrio entre tempo de inferência e acurácia, sobretudo em cenários com alta dimensionalidade, enquanto os PCEs destacaram-se pela rapidez de treinamento e bom desempenho em variáveis específicas. Já os Processos Gaussianos mostraram desempenho competitivo, especialmente em duas quantidades de interesse, nas quais obtiveram os menores erros quadráticos médios entre todos os métodos.

Palavras-chave: modelo eletrofisiológico cardíaco; emulador; processo Gaussiano; expansão em caos polinomial; redes neurais.

ABSTRACT

Heart diseases are common nowadays, and the need for studies on electrophysiological activity has been met through computational simulations that assist in medical diagnosis. The use of statistical tools such as Uncertainty Quantification (UQ) and Sensitivity Analysis (SA) guides the development of computational models by identifying the most influential parameters. However, these tools require a large amount of data to produce reliable results. Due to the high computational cost of simulations, emulators emerge as an alternative to generate sufficient data and enable the application of UQ and SA. The objective of this work was to analyze three emulators, Polynomial Chaos Expansion (PCE), Gaussian Processes (GP), and Neural Networks (NN), regarding their efficiency and accuracy in training and predicting values of two electrophysiological models based on the ten Tusscher (2004) model. Initially, different kernels were investigated for the GPs to select the most suitable ones and compare them with the other emulators. Subsequently, the methods were evaluated based on the mean squared error (MSE) and on training and inference times, with a normalized score obtained by averaging these metrics. In general, the results show that Neural Networks presented the best balance between inference time and accuracy, especially in high-dimensional scenarios, while PCEs stood out for their fast training and good performance in specific variables. Gaussian Processes, in turn, showed competitive performance, especially in two quantities of interest, in which they achieved the lowest mean squared errors among all methods.

Keywords: cardiac electrophysiological model; emulator; Gaussian process; polynomial chaos expansion; neural networks.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

De acordo com a Organização Mundial da Saúde, as doenças cardíacas foram a principal causa de morte no mundo entre os anos de 2000 e 2021. Com o objetivo de compreender melhor as enfermidades responsáveis por esse elevado número de fatalidades, modelos *in silico* têm sido cada vez mais empregados. Esses modelos possibilitam a análise aprofundada dos mecanismos patológicos, apoiando decisões médicas personalizadas. A precisão e a rapidez no diagnóstico gerado por esses modelos são fundamentais para a prevenção de complicações e para o tratamento adequado dos pacientes. Contudo, cada paciente possui características eletrofisiológicas únicas e a utilização de modelos matemáticos computacionais permite a melhor individualização do diagnóstico.

Nesses modelos é importante reconhecer os parâmetros mais influentes e suas incertezas, que podem ser obtidos com o auxílio de métodos estatísticos como a Análise de Sensibilidade e a Quantificação de Incertezas. Ao empregar essas ferramentas, surge o desafio de obter grandes quantidades de dados, onde a utilização de dados reais se torna inviável, custosa financeiramente e temporalmente. As simulações computacionais solucionam parte do problema, mas, ainda assim, podem demorar um longo período para serem executadas, acarretando elevado custo temporal e financeiro, por requererem máquinas com alto poder computacional que processem simulações por um longo período.

Visando solucionar esta necessidade de obter grande quantidade de dados para realizar a Análise de Sensibilidade e Quantificação de Incertezas, os emuladores tornam-se ferramentas úteis. Com isso, o presente estudo compara três diferentes emuladores quanto à sua acurácia e eficácia ao treinar e predizer valores de dois modelos eletrofisiológicos.

Os emuladores utilizados foram: Expansão em Caos Polinomial (PCE), Processos Gaussianos (GP) e Redes Neurais (NN). Os dois modelos eletrofisiológicos utilizados neste estudo foram obtidos de alterações no modelo de ten Tusscher 2004. O primeiro, um modelo isquêmico, simula diferentes graus de isquemia através da alteração de três componentes principais: acidose, hipóxia e hipercalemia. O segundo, um modelo isquêmico com diversidade celular, combina os parâmetros isquêmicos com nove coeficientes multiplicativos adicionais que representam a variabilidade natural entre diferentes células cardíacas.

Há um enfoque maior nos emuladores por Processos Gaussianos, onde são realizadas análises dos parâmetros de alguns kernels e, em seguida, uma análise exploratória dos dados. Após essas etapas, é possível escolher os kernels que serão comparados quanto à sua acurácia e eficácia.

Após avaliar os kernels escolhidos, com diferentes tamanhos de dados de treino, pode-se calcular a nota normalizada e obter a média das métricas apuradas. Essa nota possibilita escolher os melhores kernels dos Processos Gaussianos, que são selecionados e comparados com os demais emuladores: Expansão em Caos Polinomial e Redes Neurais.

A comparação da acurácia dos emuladores em cada modelo é realizada em três configurações distintas dos emuladores e com quatro quantidades variadas de dados de treinamento. Para cada combinação de emulador e quantidade de dados de treino, a avaliação da acurácia é feita com base no erro quadrático médio (MSE), enquanto a eficácia é medida pelo tempo necessário para treinar cada emulador e realizar inferências com novos dados. Por fim, calcula-se a nota normalizada e obtém-se a média das métricas, a fim de identificar o melhor emulador para cada um dos dois modelos eletrofisiológicos propostos.

1.1 Objetivo

O objetivo geral do trabalho é comparar a efetividade e acurácia de emuladores em modelos cardíacos eletrofisiológicos simulados, verificando como os Processos Gaussianos performam frente a outros emuladores.

Os objetivos específicos incluem:

- Analisar a alocação de memória na unidade de processamento gráfico pelos diferentes kernels dos Processos Gaussianos em função do tamanho da amostra de treino;
- Avaliar o tempo de treinamento dos diferentes Processos Gaussianos utilizando diferentes configurações de kernel;
- Investigar a eficiência de inferência dos kernels dos Processos Gaussianos;
- Quantificar o erro quadrático médio (MSE) dos kernels para diferentes tamanhos de amostra de treino;
- Comparar a acurácia (MSE) e a eficiência (tempo de treino e tempo de inferência) entre os emuladores por Processos Gaussianos, Expansão em Caos Polinomial e Redes Neurais.

1.2 Organização do texto

O manuscrito foi elaborado em etapas. No referencial teórico, tem-se as definições, os trabalhos que embasam o tema e os avanços alcançados em trabalhos análogos ao estudado. Na modelagem eletrofisiológica cardíaca, tem-se a biofísica básica sobre potencial de ação e modelagem matemática dos modelos estudados. Na metodologia, há a expressão dos passos percorridos, da geração e interpretação de dados. No item de resultados e discussão, são apresentados os testes com emuladores, avaliando a acurácia e eficácia de dois modelos. A conclusão revisita os principais resultados alinhados aos objetivos do trabalho. Ao final deste item, tem-se as considerações que podem nortear trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Contextualização sobre doenças cardíacas

De acordo com a Organização Mundial da Saúde (World Health Organization, 2023), doenças cardíacas foram a principal causa de morte no mundo entre os anos de 2000 e 2021, sendo responsáveis por 13% deste total. Em 2000, estima-se que foram 2,7 milhões, chegando a 9,1 milhões em 2021. Visando compreender melhor as enfermidades que acarretam nesse grande quantitativo de fatalidades, há a necessidade de utilizar ferramentas, como a simulação cardíaca, para permitir uma melhor investigação.

2.2 Simulação eletrofisiológica cardíaca

Uma simulação busca representar, imitando, o funcionamento de um sistema real por meio de um modelo computacional (Merriam-Webster Dictionary, 2024). Visando simular o sistema cardíaco, simulações computacionais vêm sendo empregadas de forma crescente para compreender mecanismos patológicos, planejar terapias e apoiar decisões médicas personalizadas. Modelos *in silico* combinam eletrofisiologia, mecânica e hemodinâmica para reproduzir virtualmente o comportamento do coração humano sob condições fisiológicas e patológicas, permitindo quantificar índices de função cardíaca de forma não invasiva e testar intervenções terapêuticas antes de sua aplicação em pacientes reais (TRAYANOVA, 2011; NIEDERER; LUMENS; TRAYANOVA, 2019).

Dentre as diversas vertentes da simulação cardíaca, se destaca a eletrofisiológica. Estudos demonstraram como variações nas condutâncias iônicas e na condução tecidual influenciam a suscetibilidade a arritmias e a resposta a bloqueadores de canais, servindo de plataforma para ensaios clínicos virtuais que aceleram a triagem de compostos, melhorando a compreensão deste fenômeno e aumentando a segurança cardiológica de novos tratamentos (BIFULCO; AKOUM; BOYLE, 2021; MARGARA et al., 2021; YANG et al., 2025).

Os modelos eletrofisiológicos cardíacos baseiam-se fundamentalmente no formalismo matemático estabelecido por Hodgkin e Huxley (HODGKIN; HUXLEY, 1952), que descreve o comportamento elétrico da membrana celular através de equações diferenciais ordinárias. Estes modelos representam os canais iônicos como condutâncias dependentes da voltagem e do tempo, permitindo a simulação detalhada do potencial de ação cardíaco. O modelo de ten Tusscher (TUSSCHER et al., 2004), amplamente utilizado na literatura científica, representa um marco na modelagem eletrofisiológica ventricular humana, incorporando descrições detalhadas de 12 correntes transmembrânicas distintas e permitindo a diferenciação entre tipos celulares epicárdicos, endocárdicos e do meio-miocárdio.

2.3 Abordagem inicial dos emuladores

Um emulador pode ser entendido como uma aproximação estatística do simulador. Em vez de fornecer um único valor $f(x)$ para uma dada configuração de entrada x, ele fornece uma estimativa $\hat{f}(x)$ acompanhada de uma distribuição de probabilidade, que descreve a incerteza associada à aproximação (O'HAGAN, 2006).

Paralelamente ao desenvolvimento das simulações, os emuladores têm-se mostrado substitutos eficazes para modelos cardíacos, que requerem um grande custo computacional. Isto o torna muito mais rápido do que um simulador, podendo gerar mais saídas com um menor custo computacional (O'HAGAN, 2006; ALIZADEH; ALLEN; MISTREE, 2020). Esta eficiência computacional é particularmente valiosa quando há a necessidade de uma grande quantidade de dados ou de resultados mais céleres, principalmente quando se tem interesse em utilizar ferramentas estatísticas como a análise de sensibilidade (SA) e/ou a quantificação de incertezas (UQ) (O'HAGAN, 2006; DOMINGUEZ-GOMEZ et al., 2024; PIGOZZO; SANTOS; ROCHA, 2025). Essas ferramentas permitem explorar grandes espaços paramétricos com baixo custo computacional, fornecendo índices de sensibilidade que orientam a priorização de variáveis e aprimoram a robustez de estratégias terapêuticas, e os emuladores são cruciais na sua aplicabilidade (CHANG; STRONG; CLAYTON, 2015; CAMPOS et al., 2020; RUPP et al., 2020).

2.4 Processos Gaussianos: Fundamentos teóricos e aplicações

2.4.1 Definição dos Processos Gaussianos

Um Processo Gaussiano (GP) é formalmente definido como uma coleção de variáveis aleatórias, de modo que qualquer subconjunto finito dessas variáveis segue uma distribuição normal multivariada (RASMUSSEN; WILLIAMS, 2006). Esta definição, aparentemente abstrata, possui implicações práticas profundas para a modelagem de funções desconhecidas (WILLIAMS; RASMUSSEN, 1995).

Matematicamente, um processo gaussiano é completamente especificado por sua função de média $m(x)$ e sua função de covariância (kernel) $k(x, x')$:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{2.1}$$

onde $m(x) = \mathbb{E}[f(x)]$ e $k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$.

A distribuição conjunta de qualquer conjunto finito de pontos $\{x_1, x_2, \ldots, x_n\}$ segue uma distribuição normal multivariada (RASMUSSEN; WILLIAMS, 2006):

$$\mathbf{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \tag{2.2}$$

onde $\boldsymbol{\mu} = [m(x_1), m(x_2), \ldots, m(x_n)]^T$ é o vetor de médias e $\mathbf{K}$ é a matriz de covariância com elementos $K_{ij} = k(x_i, x_j)$.

Esta propriedade fundamental dos Processos Gaussianos (GPs) garante que todas as distribuições marginais e condicionais também sejam gaussianas, o que facilita enormemente os cálculos analíticos e a interpretação dos resultados. A natureza não-paramétrica dos processos gaussianos permite que eles se adaptem automaticamente à complexidade dos dados, evitando a necessidade de especificar a priori uma forma funcional específica (RASMUSSEN; WILLIAMS, 2006).

2.4.2 Aplicação dos Processos Gaussianos

Os GPs têm aplicações diversas na ciência e engenharia, sendo tradicionalmente reconhecidos por sua excelência em duas áreas principais: Quantificação de Incertezas e Regressão Não-paramétrica (WILLIAMS; RASMUSSEN, 1995).

A primeira aplicação, a Quantificação de Incertezas (UQ), se destaca devido à capacidade dos GPs de quantificar de forma natural e rigorosa a incerteza associada às predições. Diferentemente de métodos determinísticos, os GPs fornecem não apenas uma estimativa pontual, mas também uma medida de confiabilidade através da variância preditiva (O'HAGAN, 2006). Esta característica é fundamental em aplicações onde a compreensão da incerteza é crucial para a tomada de decisões, como em análise de risco e otimização sob incerteza.

A segunda área de destaque é a Regressão Não-paramétrica, já que os GPs constituem uma poderosa ferramenta de regressão que não assume uma forma funcional específica a priori. Esta flexibilidade permite a modelagem de relações complexas e não-lineares entre variáveis de entrada e saída, adaptando-se automaticamente à complexidade inerente dos dados (RASMUSSEN; WILLIAMS, 2006). A capacidade de interpolar suavemente entre pontos de dados observados, mantendo propriedades de regularidade desejáveis, torna os GPs particularmente adequados para problemas de aproximação de funções.

2.4.3 Utilidades dos Processos Gaussianos

Os Processos Gaussianos (GPs) são amplamente utilizados para quantificação de incertezas, mas também desempenham um papel crucial em diversas outras áreas, como emulação de modelos complexos e regressão não-paramétrica. Em particular, eles têm se mostrado extremamente eficazes como substitutos rápidos e precisos para simulações computacionalmente dispendiosas, como no caso de modelos eletrofisiológicos cardíacos baseados no modelo de ten Tusscher (2004) (TUSSCHER et al., 2004). Nesse contexto, os GPs são empregados para emular o comportamento de sistemas complexos, reduzindo significativamente o custo computacional associado à execução repetida de simulações de alta complexidade.

Os GPs servem como substitutos estatísticos rápidos e precisos para simulações computacionalmente custosas de modelos eletrofisiológicos cardíacos. Uma vez treinados com um conjunto limitado de simulações, os emuladores podem predizer instantaneamente as saídas do modelo para novas configurações de parâmetros de entrada, eliminando a necessidade de executar simulações adicionais (COVENEY et al., 2021).

A eficiência computacional dos emuladores permite a realização de estudos extensivos de sensibilidade paramétrica, onde milhares ou milhões de avaliações do modelo são necessárias. Isto seria computacionalmente proibitivo utilizando diretamente o simulador original (CHANG; STRONG; CLAYTON, 2015).

Os emuladores gaussianos facilitam a aplicação de métodos avançados de análise estatística, como quantificação de incertezas Bayesianas e otimização global, que requerem múltiplas avaliações da função objetivo (FRAZIER, 2018).

2.5 Emuladores por Processos Gaussianos para regressão

2.5.1 Metodologia de emulação

Os Processos Gaussianos podem ser empregados na construção de emuladores de regressão que aproximam o comportamento de simulações eletrofisiológicas cardíacas. O processo de emulação segue uma metodologia bem estabelecida que pode ser dividida em etapas distintas (O'HAGAN, 2006):

- Fase de Treinamento: Inicialmente, um conjunto de simulações é obtido com amostragem por Hipercubo Latino (LHS). Estes dados simulados constituem o conjunto de treinamento $\mathcal{D} = \{(x_i, f(x_i)) : i = 1, \ldots, n\}$, onde x_i representa os parâmetros de entrada e $f(x_i)$ as correspondentes saídas da simulação;
- Fase de Predição: Uma vez treinado, o emulador gaussiano pode gerar predições instantâneas para novas configurações de parâmetros x^* sem necessidade de executar simulações adicionais. Esta capacidade preditiva é fundamental para estudos que requerem milhares de avaliações do modelo.

Os emuladores gaussianos constituem uma aplicação específica da teoria de GPs para a construção de substitutos estatísticos de simuladores computacionais determinísticos. A abordagem trata o simulador como uma função determinística desconhecida $f : \mathcal{X} \to \mathbb{R}^m$, onde $\mathcal{X} \subseteq \mathbb{R}^d$ representa o espaço de entrada e $\mathbb{R}^m$ o espaço de saída multidimensional.

Para um novo ponto x^*, a distribuição preditiva posterior é (RASMUSSEN; WILLIAMS, 2006):

$$f(x^*)|\mathcal{D} \sim \mathcal{N}(\mu(x^*), \sigma^2(x^*)), \tag{2.3}$$

onde:

$$\mu(x^*) = m(x^*) + k(x^*, \mathbf{X})^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}), \tag{2.4}$$

$$\sigma^2(x^*) = k(x^*, x^*) - k(x^*, \mathbf{X})^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}k(x^*, \mathbf{X}), \tag{2.5}$$

com $\mathbf{X} = [x_1, \ldots, x_n]^T$ representando a matriz dos pontos de entrada utilizados no treinamento do emulador, $\mathbf{y} = [f(x_1), \ldots, f(x_n)]^T$ sendo o vetor das saídas observadas nos respectivos pontos de treinamento, e $\mathbf{m} = [m(x_1), \ldots, m(x_n)]^T$ sendo o vetor das expectativas a priori da função de média avaliada nos pontos de treinamento. O termo $k(x^*, \mathbf{X}) = [k(x^*, x_1), \ldots, k(x^*, x_n)]^T$ é o vetor de covariâncias entre o ponto de predição x^* e todos os pontos de treinamento. O parâmetro σ_n^2 representa a variância do ruído observacional, enquanto $\mathbf{I}$ é a matriz identidade.

2.5.2 Implementação para problemas multidimensionais

Uma característica importante da implementação dos GPs é o tratamento de saídas multidimensionais. Como os modelos eletrofisiológicos cardíacos produzem múltiplas quantidades de interesse (por exemplo, duração do potencial de ação, velocidade máxima de despolarização, potencial de repouso), emprega-se uma estratégia de múltiplos Processos Gaussianos independentes.

Para cada variável de saída $j \in \{1, 2, \ldots, m\}$, um GP independente é treinado:

$$f_j(x) \sim \mathcal{GP}(m_j(x), k_j(x, x')) \tag{2.6}$$

Esta abordagem, embora não capture correlações entre as variáveis de saída, oferece vantagens computacionais significativas e facilita a interpretação dos resultados. Além disso, permite a utilização de kernels específicos otimizados para cada variável de saída, adaptando-se às características particulares de cada quantidade de interesse.

2.5.3 Normalização e pré-processamento

Para garantir estabilidade numérica e convergência adequada dos algoritmos de otimização, os dados são normalizados antes do treinamento. A normalização segue a transformação:

$$y_{norm} = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{2.7}$$

onde y_{min} e y_{max} representam os valores mínimo e máximo observados nos dados de treinamento para cada variável de saída. As predições são posteriormente desnormalizadas para retornar à escala original dos dados.

2.6 Aspectos metodológicos dos Processos Gaussianos

2.6.1 Kernels e funções de covariância

A função kernel constitui o elemento central dos emuladores gaussianos, determinando as propriedades estruturais das funções que podem ser representadas pelo processo gaussiano. Formalmente, uma função kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ deve ser simétrica e positiva semidefinida para garantir que a matriz de covariância resultante seja válida (RASMUSSEN; WILLIAMS, 2006).

Outro fator de customização é habilitar para os kernels o Automatic Relevance Determination (ARD), onde cada dimensão de entrada possui seu próprio parâmetro de escala de comprimento, permitindo que o modelo aprenda automaticamente a relevância relativa de cada variável de entrada (NEAL, 1996).

2.6.2 Estimação de hiperparâmetros

Os hiperparâmetros dos kernels são estimados através da maximização da verossimilhança marginal dos dados de treinamento (RASMUSSEN; WILLIAMS, 2006):

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})^T \mathbf{K}_{\boldsymbol{\theta}}^{-1}(\mathbf{y} - \boldsymbol{\mu}) - \frac{1}{2}\log|\mathbf{K}_{\boldsymbol{\theta}}| - \frac{n}{2}\log(2\pi) \tag{2.8}$$

onde $\boldsymbol{\theta}$ representa o vetor de hiperparâmetros e $\mathbf{K}_{\boldsymbol{\theta}}$ é a matriz de covariância correspondente.

O otimizador utilizado como base pela biblioteca GPyTorch é o Adam (GARDNER et al., 2018), cuja integração com o ecossistema PyTorch permite otimização eficiente utilizando a Unidade de Processamento Gráfica (GPU) e compatibilidade direta com estruturas de redes neurais. Já a biblioteca scikit-learn emprega o método L-BFGS-B para realizar a otimização dos hiperparâmetros do kernel (PEDREGOSA et al., 2011).

2.6.3 Avaliação e validação de performance

A validação de emuladores gaussianos envolve múltiplas dimensões de avaliação. A acurácia das predições é quantificada através do erro quadrático médio (MSE), enquanto a eficiência computacional é avaliada pelos tempos de treinamento e inferência. Adicionalmente, o uso de memória GPU é monitorado para caracterizar os requisitos computacionais.

2.7 Outros métodos de emulação utilizados no trabalho

2.7.1 Expansão em Caos Polinomial

A Expansão em Caos Polinomial (PCE) aproxima a saída de um simulador determinístico por uma soma finita de polinômios ortogonais à distribuição das variáveis de entrada (XIU; KARNIADAKIS, 2002):

$$\hat{f}(x) = \sum_{i=0}^{P} c_i \, \Phi_i(x), \tag{2.9}$$

onde P é o grau máximo da expansão e $\Phi_i(x)$ são polinômios que formam uma base ortogonal, por exemplo, Legendre para variáveis uniformes e Hermite para gaussianas. Os coeficientes c_i são estimados via regressão linear entre as saídas simuladas e as bases Φ_i. Graus superiores capturam mais variância, mas aumentam a complexidade.

Campos et al. (2023) utilizaram PCEs para modelagem de mecânica cardíaca passiva empregando o modelo constitutivo de Holzapfel-Ogden, demonstrando eficácia para análise de sensibilidade global. A principal vantagem das PCEs reside na obtenção analítica de momentos estatísticos diretamente dos coeficientes da expansão, permitindo cálculo eficiente de médias, variâncias e índices de sensibilidade de Sobol.

A biblioteca ChaosPy oferece as ferramentas necessárias para gerar bases ortogonais, ajustar coeficientes e calcular automaticamente esses indicadores (FEINBERG; LANGTANGEN, 2015).

2.7.2 Redes Neurais

As Redes Neurais (NN) representam uma terceira categoria de emuladores que têm ganhado crescente atenção na modelagem cardíaca devido à sua flexibilidade e capacidade de aproximação universal (HAYKIN, 2001). A integração das NNs com modelagem de ordem reduzida tem-se mostrado eficaz para emular simulações complexas, facilitando a quantificação de incerteza e análise de sensibilidade de parâmetros em modelos eletrofisiológicos (PAGANI; MANZONI, 2021; REGAZZONI et al., 2022).

As NNs aproximam funções por meio de composições de transformações lineares e não lineares, sendo universais aproximadores (HORNIK; STINCHCOMBE; WHITE, 1989):

$$\hat{\mathbf{y}} = \sigma\Big(W^{(L)}\,\sigma(W^{(L-1)}\cdots\sigma(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)}\Big), \tag{2.10}$$

onde $W^{(i)}$ e $\mathbf{b}^{(i)}$ são os parâmetros de peso e viés da camada i, e σ é a função de ativação. Arquiteturas de diferentes profundidades e larguras permitem equilibrar capacidade de modelagem e custo computacional, variando o número de camadas e

neurônios por camada. A utilização do otimizador Adam para ajustar os parâmetros e da função de ativação SiLU (Sigmoid Linear Unit) propicia à NN a capacidade de extrair não linearidades suaves.

3 MODELAGEM ELETROFISIOLOGIA CARDÍACA

Este capítulo apresenta os fundamentos teóricos necessários para a compreensão dos modelos eletrofisiológicos cardíacos utilizados neste trabalho. Os conceitos abordados fornecem a base matemática e fisiológica para o desenvolvimento e aplicação de emuladores em simulações cardíacas, com ênfase particular nos modelos baseados no modelo de ten Tusscher modificados para representar condições isquêmicas e diversidade celular.

3.1 Potencial de ação cardíaco

O potencial de ação cardíaco é uma alteração transitória e coordenada do potencial de membrana das células do coração, permitindo a propagação do estímulo através das células do miocárdio, coordenando a contração dos átrios e ventrículos (KLÉBER; RUDY, 2004).

Como mostra a Figura 1, o potencial de ação cardíaco inicia-se com a fase de repouso (A), depois ocorre a fase de despolarização rápida (B), desencadeada pela abertura dos canais de sódio (Na^+) que permite a entrada desses íons, o que eleva o potencial transmembrânico. Logo após, esses canais se inativam, dando início à fase de repolarização inicial (C). Na fase de platô (D), os canais de cálcio (Ca^{2+}) se abrem e permitem a entrada de cálcio extracelular, e nesta fase também há a saída de potássio (K^+), fazendo com que o potencial transmembrânico forme um platô. Por fim, na fase de repolarização (E), há a saída de potássio e o potencial retorna ao seu valor de repouso (BERG, 2022).

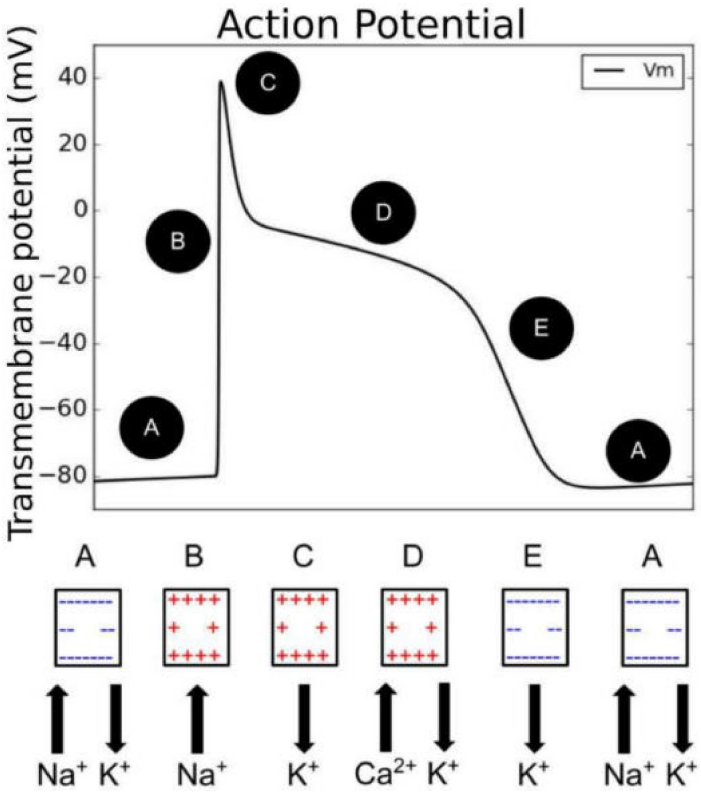

Figura 1 – Representação de um potencial de ação em uma célula cardíaca. A fase (A) indica Repouso, (B) Despolarização, (C) Repolarização inicial, (D) Platô e (E) Repolarização. Na parte inferior da figura são mostradas as correntes iônicas relacionadas à mudança de potencial em cada fase. Imagem obtida de Berg (2022).

3.2 Modelagem eletrofisiológica

A modelagem matemática da eletrofisiologia cardíaca baseia-se fundamentalmente no formalismo pioneiro estabelecido por Hodgkin e Huxley em 1952 (HODGKIN; HUXLEY, 1952). Embora originalmente desenvolvido para o axônio gigante da lula, este modelo estabeleceu a base da compreensão dos mecanismos iônicos responsáveis pela geração e propagação do potencial de ação, através da representação da membrana celular como um circuito elétrico equivalente.

3.2.1 Fundamentos matemáticos

A equação fundamental do modelo de Hodgkin-Huxley expressa o balanço de correntes através da membrana celular:

$$C_m \frac{dV_m}{dt} + I_{ion} = I_{ext}, \tag{3.1}$$

onde C_m representa a capacitância da membrana, V_m o potencial transmembrânico, I_{ion} as correntes iônicas totais e I_{ext} a corrente externa de estímulo aplicada.

As correntes iônicas são modeladas através da lei de Ohm generalizada:

$$I_{ion} = G_i \cdot (V_m - E_i), \tag{3.2}$$

sendo G_i a condutância específica do canal e E_i o potencial de reversão da corrente iônica correspondente.

A genialidade do modelo reside na descrição das condutâncias como funções dinâmicas da voltagem e do tempo, através de variáveis de ativação e inativação:

$$\frac{dx}{dt} = \alpha_x(V)(1 - x) - \beta_x(V)x, \tag{3.3}$$

onde x representa qualquer variável de ativação de canal iônico, e α_x e β_x são funções empíricas da voltagem que determinam as taxas de transição entre os estados fechado e aberto dos canais.

3.2.2 Modelo de ten Tusscher

O modelo de ten Tusscher, desenvolvido em 2004 (TUSSCHER et al., 2004), incorpora 12 correntes transmembrânicas distintas e utiliza 17 variáveis de estado, fornecendo uma descrição detalhada da eletrofisiologia humana.

As correntes modeladas incluem:

- Corrente rápida de sódio (I_{Na})
- Corrente de cálcio tipo L (I_{CaL})
- Correntes retificadoras tardias rápida (I_{Kr}) e lenta (I_{Ks})
- Corrente retificadora de entrada (I_{K1})
- Corrente transitória de saída (I_{to})
- Corrente de potássio de platô (I_{pK})
- Trocador sódio-cálcio (I_{NaCa})
- Bomba sódio-potássio (I_{NaK})
- Bomba de cálcio sarcolemmal (I_{pCa})
- Correntes de fundo de sódio (I_{bNa}) e cálcio (I_{bCa})

Os detalhes completos sobre a modelagem dessas correntes iônicas podem ser encontrados na referência original (TUSSCHER et al., 2004).

A equação diferencial principal mantém a forma clássica:

$$C_m \frac{dV}{dt} = -(I_{Na}+I_{CaL}+I_{Kr}+I_{Ks}+I_{K1}+I_{to}+I_{NaCa}+I_{NaK}+I_{pCa}+I_{pK}+I_{bNa}+I_{bCa}) \quad (3.4)$$

Uma característica do modelo de ten Tusscher é a capacidade de simular diferentes tipos de células ventriculares. As células epicárdicas, endocárdicas e do meio-miocárdio são diferenciadas primariamente através das condutâncias das correntes I_{to} e I_{Ks}.

3.3 Quantidades de interesse e análise eletrofisiológica

Neste estudo, obteve-se, do potencial de ação de uma célula cardíaca, quatro quantidades de interesse (QoI) que caracterizam aspectos fundamentais da eletrofisiologia do coração, que são as seguintes:

- Velocidade máxima de despolarização (dVmax):

$$dVmax = \max_{t\in[t_0,t_1]} \left(\frac{dV}{dt}\right) \quad (3.5)$$

  Reflete diretamente a disponibilidade e cinética dos canais de sódio, servindo como indicador da excitabilidade celular e velocidade de condução. Valores normais situam-se entre 200-400 V/s, sendo significativamente reduzidos em condições isquêmicas.

  A velocidade máxima de despolarização representa o maior valor da derivada do potencial transmembrânico em relação ao tempo dentro de um intervalo definido (t_0 a t_1). Essa QoI é evidenciada na fase 0 do potencial de ação representado na Figura 2.

- Duração do potencial de ação (APD50 e APD90):

$$APD50 = t_{50\%} - t_0 \quad (3.6)$$

$$APD90 = t_{90\%} - t_0 \quad (3.7)$$

  Quantificam diferentes aspectos da repolarização com base na duração do potencial de ação. O valor de APD50 corresponde ao intervalo entre o início do potencial de ação (t_0) e o momento em que o potencial atinge 50% da diferença entre o pico da despolarização e o potencial de repouso. Já o APD90 refere-se ao tempo até que o potencial retorne a 90% da repolarização total. Essas QoIs são evidenciadas na fase 3 do potencial de ação representado na Figura 2.

- Potencial de repouso (Vreps): É definido pelo potencial elétrico através da membrana celular quando a célula está em estado de repouso, ou seja, sem estar sendo estimulada. Esse valor resulta do equilíbrio dinâmico entre as concentrações de íons e é evidenciado pela linha pontilhada preta presente no potencial de ação representado na Figura 2.

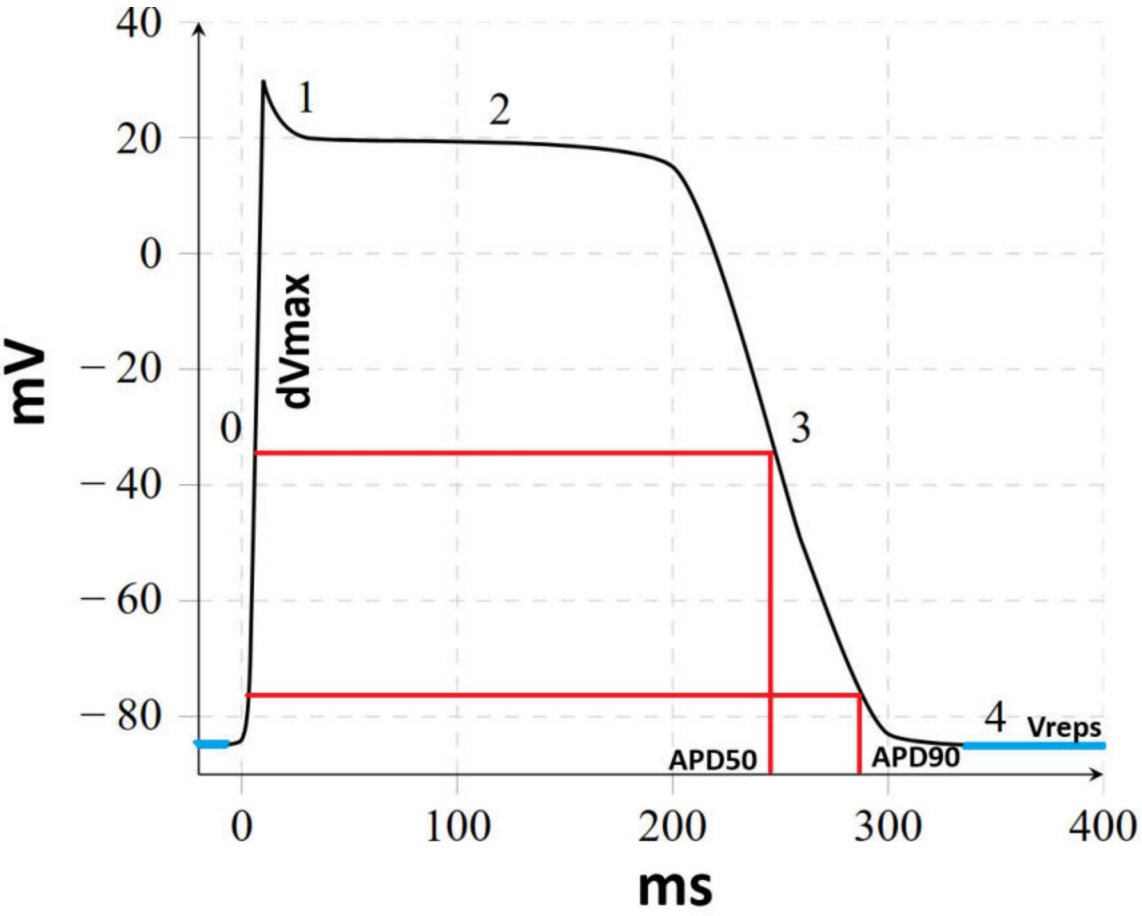

Figura 2 – Representação do potencial de ação de uma célula cardíaca e suas fases. A fase (0) é a despolarização, (1) repolarização inicial, (2) platô, (3) repolarização final e (4) repouso. Os locais onde se obtém as quantidades de interesse estão evidenciados, sendo eles dVmax, APD50, APD90 e Vreps.

3.4 Modelo A - isquêmico

O Modelo A implementa uma representação computacional de condições isquêmicas através da modificação sistemática de parâmetros eletrofisiológicos. Esta abordagem, fundamentada no trabalho de Oliveira et al. (2018), posteriormente utilizada por Lawson et al. (2020), incorpora três componentes fisiopatológicos principais da isquemia miocárdica, que são integrados ao modelo pela adição de fatores multiplicativos nas condutâncias que são alteradas pelas variáveis de entrada do modelo.

3.4.1 Parametrização matemática

O modelo utiliza três variáveis de entrada, acidose, hipóxia e hipercalemia, que variam uniformemente entre 0 (condições fisiológicas normais) e 1 (estado patológico máximo), como mostra a Tabela 1.

A acidose intracelular, característica da isquemia, corresponde à alteração do pH sanguíneo e afeta múltiplas correntes iônicas através de mecanismos diretos e indiretos. O modelo A implementa essas alterações através das seguintes modificações:

Condutância de sódio modificada:

$$g_{Na} = g_{Na_{base}} \cdot (1 - 0.25 \cdot acidose) \tag{3.8}$$

Condutância de cálcio tipo L modificada:

$$g_{CaL} = g_{CaL_{base}} \cdot (1 - 0.25 \cdot acidose) \tag{3.9}$$

Concentração intracelular de potássio modificada:

$$[K^+]_i = [K^+]_{i_{base}} - 13.3 \cdot acidose \tag{3.10}$$

Estas modificações evidenciam que a redução de g_{Na} resulta na diminuição de dVmax, enquanto a redução de g_{CaL} afeta o platô do potencial de ação.

A hipóxia, que é a falta de oxigênio, faz com que a mitocôndria pare de produzir energia (ATP) de forma eficiente, o que diminui os níveis de ATP dentro da célula e ativa canais de potássio ($I_{K(ATP)}$).

Concentração de ATP modificada:

$$[ATP]_i = [ATP]_{i_{base}} - 3.0 \cdot hipóxia \tag{3.11}$$

A redução do ATP intracelular ativa os canais $I_{K(ATP)}$, resultando em encurtamento significativo da duração do potencial de ação, acarretando no encurtamento do potencial de ação.

A hipercalemia, que é o acúmulo extracelular de potássio, constitui uma das alterações mais precoces e significativas durante a isquemia, resultante da alteração da permeabilidade da membrana e da redução da atividade da bomba sódio-potássio:

Concentração extracelular de potássio modificada:

$$[K^+]_o = [K^+]_{o_{base}} + 4.6 \cdot hipercalemia \tag{3.12}$$

Esta alteração afeta diretamente o potencial de repouso através da relação de Nernst:

$$E_K = \frac{RT}{F} \ln \left(\frac{[K^+]_o}{[K^+]_i} \right), \tag{3.13}$$

resultando em uma despolarização significativa do potencial de repouso e consequente redução da disponibilidade dos canais de sódio para ativação. Com isso, a hipercalemia estabelece uma forte correlação com Vreps e dVmax, demonstrando ser o principal determinante desses parâmetros nas condições isquêmicas modeladas.

3.5 Modelo B - isquêmico + diversidade celular

O Modelo B constitui uma extensão do Modelo A que incorpora a diversidade celular observada em populações de células cardíacas. Esta abordagem reconhece que existe variabilidade significativa na expressão de canais iônicos entre células individuais.

3.5.1 Parametrização matemática

O modelo utiliza doze variáveis de entrada, combinando os três parâmetros isquêmicos do Modelo A (acidose, hipóxia e hipercalemia) com nove coeficientes multiplicativos que representam a diversidade celular, como mostra a Tabela 1.

As condutâncias no Modelo B são calculadas através da aplicação sequencial dos efeitos isquêmicos e dos coeficientes de diversidade celular. É importante notar que os coeficientes de diversidade multiplicam os valores já modificados pelos parâmetros isquêmicos:

Condutância de sódio:

$$g_{Na} = g_{Na_{base}} \cdot (1 - 0.25 \cdot acidose) \cdot g_{NaC} \tag{3.14}$$

Condutância de cálcio tipo L:

$$g_{CaL} = g_{CaL_{base}} \cdot (1 - 0.25 \cdot acidose) \cdot g_{CaLC} \tag{3.15}$$

Estas condutâncias sofrem duplo efeito modificador, primeiro pela acidose e posteriormente pelos coeficientes g_{NaC} e g_{CaLC}. A redução de g_{Na} por acidose e sua modulação adicional pela diversidade celular resulta em impacto direto sobre dVmax. Similarmente, a modificação de g_{CaL} afeta a amplitude e duração do platô do potencial de ação, influenciando principalmente no potencial de ação.

Condutâncias de Potássio:

$$g_{K1} = g_{K1_{base}} \cdot g_{K1C} \tag{3.16}$$

$$g_{Kr} = g_{Kr_{base}} \cdot g_{KrC} \tag{3.17}$$

$$g_{Ks} = g_{Ks_{base}} \cdot g_{KsC} \tag{3.18}$$

$$g_{to} = g_{to_{base}} \cdot g_{toC} \tag{3.19}$$

Os coeficientes de condutância de potássio apresentam correlações inferiores às das variáveis do modelo isquêmico e dos parâmetros g_{NaC} e g_{CaLC}, indicando que a variabilidade individual nestas condutâncias tem baixo impacto. Este padrão sugere que os mecanismos isquêmicos dominam sobre a diversidade celular natural na determinação das características eletrofisiológicas.

Condutâncias de Fundo e Bombas:

$$g_{bCa} = g_{bCa_{base}} \cdot g_{bCaC} \tag{3.20}$$

$$g_{pK} = g_{pK_{base}} \cdot g_{pKC} \tag{3.21}$$

$$g_{pCa} = g_{pCa_{base}} \cdot g_{pCaC} \tag{3.22}$$

Similarmente às condutâncias de potássio, as condutâncias de fundo e bombas apresentam correlações fracas, sendo g_{pCaC} a única que apresenta correlação ligeiramente mais expressiva, devido ao seu papel secundário na regulação do cálcio intracelular durante a repolarização, interferindo assim no potencial de ação.

4 METODOLOGIA

4.1 Modelos eletrofisiológicos

Este trabalho utilizou dados simulados de dois modelos eletrofisiológicos, Modelo A e Modelo B, para treinar e validar os emuladores. Os modelos fundamentam-se na utilização de duas parametrizações distintas do modelo eletrofisiológico de ten Tusscher 2004 (TT) (TUSSCHER et al., 2004), cada uma projetada para capturar aspectos específicos da variabilidade eletrofisiológica cardíaca.

4.1.1 Modelo A - isquêmico

O primeiro modelo, denominado de Modelo A, representa diferentes graus de isquemia. Esta enfermidade é simulada pela alteração de três componentes principais: acidose, hipóxia e hipercalemia. Essa parametrização baseia-se diretamente nas modificações propostas por Oliveira et al. (2018), posteriormente utilizadas por Lawson et al. (2020), cuja abordagem serve de base para o presente trabalho. O controle de cada componente é observado na Tabela 1 e tem o seu potencial de ação apresentado na Figura 3.

4.1.2 Modelo B - isquêmico + diversidade Celular

O segundo modelo, denominado Modelo B, constitui uma extensão do Modelo A, pois combina os três parâmetros isquêmicos do primeiro modelo com nove coeficientes multiplicativos adicionais relativos à diversidade celular de cada indivíduo, que modulam as condutâncias iônicas fundamentais do modelo de ten Tusscher, como pode ser observado na Tabela 1 e tem o seu potencial de ação apresentado na Figura 3. Esta estratégia permite representar a diversidade morfológica dos potenciais de ação observada em populações celulares, capturando a variabilidade natural existente entre diferentes células cardíacas ou diferentes regiões do miocárdio.

Tabela 1 – Comparação entre os modelos isquêmico e de diversidade celular por meio das variáveis de entrada e seus efeitos nos parâmetros do modelo de ten Tusscher. O Modelo A contêm três parâmetros (corresponde ao isquêmico) e o Modelo B contêm 12 parâmetros (isquêmico + diversidade celular).

Modelo	**Parâmetro**	**Entrada**	**Valor base**	**Valor da doença**	**Intervalo**
Isquêmico	Acidose	g_{Na}	1,50E+01	1,10E+01	0,0-1,0
		g_{Cal}	1,75E-01	1,31E-01	
		K_i	1,38E+02	1,25E+02	
	Hipóxia	ATP	5,40E+00	4,05E+00	0,0-1,0
	Hipercalemia	K_0	5,60E+00	1,00E+01	0,0-1,0
	Parâmetro	**Entrada**	**Valor base**		**Intervalo**
Diversidade celular	g_{NaC}	g_{Na}	g_{Na}	-	0,5–1,5
	g_{CalC}	g_{Cal}	g_{Cal}	-	0,5–1,5
	g_{K1C}	g_{K1}	5,40E+00	-	0,5–1,5
	g_{KrC}	g_{Kr}	9,60E+01	-	0,5–1,5
	g_{KsC}	g_{Ks}	2,45E+02	-	0,5–1,5
	g_{toC}	g_{to}	2,94E+02	-	0,5–1,5
	g_{bCaC}	g_{bCa}	5,92E-01	-	0,5–1,5
	g_{pCaC}	g_{pCa}	g_{pCa}	-	0,5–1,5
	g_{pKC}	g_{pK}	1,46E-02	-	0,5–1,5

Fonte: Elaboração própria.

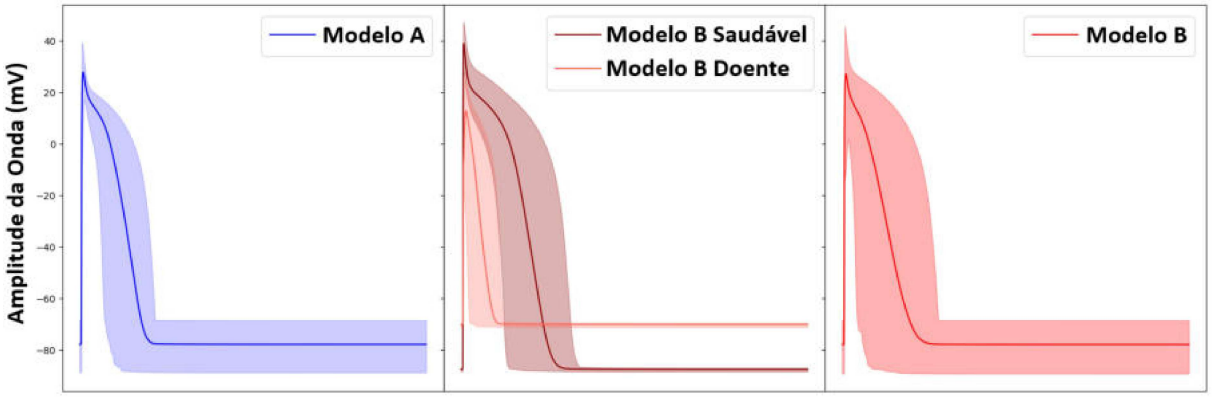

Figura 3 – Variação da forma de onda do potencial de ação para os Modelos A e B. Imagem à esquerda contem o Modelo A com todos os parâmetros variando de 0 a 1 (sem doença até o máximo de doença). Imagem central expõe o Modelo B com coeficientes de condução variando de 0.5 a 1.5, parâmetros isquêmicos fixados em 0 ou 1. Imagem à direita exibe o Modelo B com parâmetros isquêmicos variando de 0 a 1, coeficientes de condução variando de 0 a 1.5.

4.2 Simulação eletrofisiológica

A primeira etapa da simulação consiste na amostragem dos parâmetros de entrada dos modelos, de acordo com os intervalos apresentados na Tabela 1. Para cumprir esse requisito, utiliza-se o método de Hipercubo Latino (LHS), uma técnica de amostragem estratificada que garante cobertura uniforme e eficiente do espaço paramétrico multi-dimensional, apresentando vantagens significativas sobre a amostragem aleatória pura, principalmente em termos de convergência estatística mais rápida e melhor distribuição espacial das amostras, características essenciais para a construção de emuladores precisos.

O processo de simulação eletrofisiológica segue, utilizando cada combinação de parâmetros gerada pelo LHS para executar uma simulação completa do modelo de ten Tusscher. Cada simulação resolve numericamente um sistema composto por diversas equações diferenciais ordinárias que regem a evolução temporal do potencial transmembrânico, gerando, assim, cada ciclo do potencial de ação. A simulação é conduzida por 20 ciclos de potencial de ação, pois, dessa forma, atinge-se o estado estacionário, minimizando os efeitos transitórios que poderiam contaminar as quantidades de interesse. Os ciclos do potencial de ação são iniciados excitando a célula até $-52mV$ por um milissegundo, a cada 1 segundo. Após esses ciclos, são extraídas, do potencial de ação, quatro quantidades de interesse fundamentais, que foram mostradas na Figura 2: a velocidade máxima de subida do potencial de ação (dVmax), a duração do potencial de ação quando se atinge 50% da repolarização (APD50), a duração do potencial de ação quando se atinge 90% da repolarização (APD90) e o potencial de repouso (Vreps).

4.3 Criação das bases de dados

Cada dado representa um conjunto formado pelos valores de entrada e pelos valores de saída de uma simulação. Os valores de entrada correspondem às variáveis do modelo, definindo as condições sob as quais a simulação será executada. Já os valores de saída são obtidos ao calcular as quantidades de interesse (QoI) analisadas. Assim, cada simulação produz um par entrada-saída: as entradas especificam os valores que cada variável do modelo deve assumir, e as saídas resultam do cálculo das QoIs após a simulação sob essas condições.

Os scripts das simulações foram executados várias vezes para gerar conjuntos de dados com tamanhos de 100, 500, 1000 e 5000, com o objetivo de treinar os diferentes emuladores e investigar como o aumento do tamanho da amostra impacta sua acurácia e eficiência. Por fim, foi gerada uma última simulação com 100 mil dados, que serviu como base para a inferência posterior dos dados após o treinamento dos emuladores.

Após a geração das bases de dados de diferentes tamanhos, as suas colunas foram divididas, criando assim os dados de entrada e de saída para a construção dos emuladores. Os dados de entrada são compostos pelas variáveis que determinam as características isquêmicas e/ou a diversidade celular. O Modelo A possui três colunas, enquanto o Modelo B possui 12 colunas. Os dados de saída correspondem às quantidades de interesse, que são quatro no total: dVmax, APD50, APD90 e Vreps.

4.4 Análise dos efeitos dos parâmetros dos kernels

Antes de utilizar os dados gerados pelas simulações eletrofisiológicas, foi realizada uma análise dos kernels dos Processos Gaussianos, com o intuito de compreender como os parâmetros influenciam o poder de ajuste aos dados. Dessa forma, podem-se selecionar alguns kernels para avaliar sua performance nos dados simulados e, posteriormente, escolher os melhores para realizar o comparativo com as NNs e os PCEs.

A análise foi feita utilizando a função:

$$f(x) = x \cdot sen(x), \tag{4.1}$$

Esta equação foi escolhida por ser contínua, possuir fácil visualização em gráficos, variações locais e ser oscilatória. Além disso, a função possui características importantes para observar como os diferentes kernels e seus parâmetros capturam suas oscilações periódicas, devido ao $sen(x)$, e seu crescimento linear de amplitude, causado pelo fator x.

Foram gerados mil valores para $x = [0, 25]$ para serem utilizados na inferência dos Processos Gaussianos, e então foram amostrados dez valores, dentro do intervalo de cinco a vinte, igualmente espaçados para serem os dados de treino. Esse recorte foi feito entre

o intervalo de inferência e de treino para observar como alguns modelos podem gerar overfitting ou underfitting. O overfitting ocorre quando o modelo se ajusta excessivamente aos dados de treino, capturando até mesmo o ruído e as flutuações aleatórias, o que prejudica sua capacidade de inferir novos dados. Em contraste, o underfitting acontece quando o modelo não consegue capturar a complexidade dos dados, tanto nos dados de treino quanto nos de teste.

Para o treinamento dos Processos Gaussianos, nessa análise, não houve otimização dos parâmetros de cada kernel, logo eles foram fixados em valores pré-definidos para evidenciar a atuação isolada de cada parâmetro na capacidade de ajuste. Nos kernels que possuíam dois parâmetros, adotou-se uma estratégia de análise individual, onde inicialmente, um parâmetro era fixado enquanto o outro era variado através de diferentes valores pré-definidos. Em seguida, o processo era invertido. A avaliação do desempenho foi realizada com o cálculo do coeficiente de determinação R^2, que quantifica o grau de correspondência entre os valores inferidos pelo GP e os dados reais, comparando a inferência com os mil pontos gerados inicialmente no intervalo $[0, 25]$. Complementarmente, utilizou-se auxílio visual com gráficos para uma análise qualitativa mais detalhada do ajuste obtido pelos kernels em cada configuração de parâmetros.

4.4.1 Radial basis function

O kernel Radial Basis Function (RBF) é um dos mais utilizados devido à sua simplicidade e eficácia em modelar funções suaves. Este kernel, apresentado pela equação (4.2), produz funções infinitamente diferenciáveis e é controlado por um único parâmetro principal, ℓ, que é o parâmetro de largura (lengthscale), responsável por definir a escala de correlação entre os pontos de entrada.

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2\ell^2}\right) \tag{4.2}$$

O kernel RBF assume que pontos próximos no espaço de entrada têm saídas correlacionadas, e essa correlação decai exponencialmente com a distância (d) entre os pontos x_i e x_j, sendo controlada pelo parâmetro ℓ, onde valores pequenos resultam em correlações que decaem rapidamente (funções mais rugosas), enquanto valores grandes mantêm correlações por distâncias maiores (funções mais suaves).

Para compreender o impacto do parâmetro lengthscale no desempenho do kernel, foram testados os seguintes valores: 0.1, 0.5, 1.0, 2.0 e 5.0.

4.4.2 Rational quadratic

O kernel Rational Quadratic (RQ) é uma generalização do kernel RBF que pode ser interpretado como uma mistura infinita de kernels RBF com diferentes lengthscales.

Este kernel, apresentado pela equação (4.3), é controlado por dois parâmetros principais: ℓ (lengthscale), que define a escala de correlação espacial entre os pontos, e α, que controla a mistura relativa de diferentes escalas de comprimento.

$$k(x_i, x_j) = \left(1 + \frac{d(x_i, x_j)^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{4.3}$$

Uma propriedade fundamental do kernel RQ é sua relação com o kernel RBF: quando $\alpha \to \infty$, o kernel RQ converge para o kernel RBF. Para valores finitos de α, o kernel apresenta caudas mais pesadas que o RBF, proporcionando maior robustez a outliers e capturando correlações de longo alcance. O parâmetro α controla essa transição, onde valores pequenos geram funções com alta variabilidade e valores grandes produzem comportamento semelhante ao RBF.

Como o kernel dispõe de dois parâmetros, inicialmente foi analisado o impacto do parâmetro α definindo o seu valor em 0.1, 0.5, 1.0, 2.0 e 10.0, mantendo o parâmetro $\ell = 1.0$. Logo depois, fixou-se $\alpha = 1.0$ e o parâmetro ℓ assumiu os valores 0.1, 0.5, 1.0, 2.0 e 5.0.

4.4.3 Matérn

O kernel Matérn é uma família de kernels que oferece maior flexibilidade em relação ao kernel RBF, permitindo controlar explicitamente a suavidade das funções geradas através do parâmetro ν. Este kernel, apresentado pela equação (4.4), é controlado por dois parâmetros principais: ℓ (lengthscale), que define a escala de correlação espacial entre os pontos, e ν, que controla o grau de suavidade e diferenciabilidade das funções resultantes.

$$k(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} \cdot d(x_i, x_j)}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} \cdot d(x_i, x_j)}{\ell}\right) \tag{4.4}$$

onde $\Gamma(\nu)$ é a função gama e K_ν é a função de Bessel modificada de segunda espécie. Uma propriedade importante do kernel Matérn é sua relação com o kernel RBF: quando $\nu \to \infty$, o kernel Matérn converge para o kernel RBF, tornando-se infinitamente diferenciável. Para valores finitos de ν, as funções geradas são diferenciáveis até $\lfloor \nu - 1/2 \rfloor$ vezes, oferecendo um controle preciso sobre a rugosidade do modelo.

Os valores mais comuns são $\nu = 1/2$ (kernel exponencial), $\nu = 3/2$ (Matérn 3/2) e $\nu = 5/2$ (Matérn 5/2), pois possuem formas fechadas simples e cobrem uma ampla gama de comportamentos de suavidade.

Para $\nu = 3/2$, o kernel Matérn assume a forma simplificada apresentada na equação (4.5), gerando funções que são diferenciáveis uma vez, mas com derivadas não contínuas, proporcionando um grau intermediário de rugosidade.

$$k(x_i, x_j) = \left(1 + \frac{\sqrt{3} \cdot d(x_i, x_j)}{\ell}\right) \exp\left(-\frac{\sqrt{3} \cdot d(x_i, x_j)}{\ell}\right) \tag{4.5}$$

Para $\nu = 5/2$, o kernel Matérn assume a forma apresentada na equação (4.6), gerando funções duas vezes diferenciáveis, proporcionando maior suavidade intrínseca em comparação ao Matérn 3/2.

$$k(x_i, x_j) = \left(1 + \frac{\sqrt{5} \cdot d(x_i, x_j)}{\ell} + \frac{5 \cdot d(x_i, x_j)^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5} \cdot d(x_i, x_j)}{\ell}\right) \tag{4.6}$$

Para compreender o impacto do parâmetro lengthscale nos Matérn 3/2 e 5/2, foram testados diferentes valores fixos: 0.1, 0.5, 1.0, 2.0 e 5.0. Logo depois, fixou-se $\ell = 1.0$ e o parâmetro ν assumiu os valores 0.5, 1.5 e 2.5.

4.4.4 Periódico

O kernel periódico é uma função de covariância específica para processos gaussianos que permite modelar funções que apresentam comportamento repetitivo ou cíclico. Este kernel é controlado por dois parâmetros principais: o período p, que define o intervalo de repetição da periodicidade, e o lengthscale ℓ, que controla a suavidade das oscilações dentro de cada período.

$$k(x_i, x_j) = \exp\left(-\frac{2sen^2\left(\frac{\pi|x_i - x_j|}{p}\right)}{\ell^2}\right) \tag{4.7}$$

Uma característica fundamental do kernel periódico é sua capacidade de capturar repetições exatas na função modelada. Diferentemente de outros kernels que assumem apenas correlações locais, o kernel periódico estabelece correlações entre pontos separados por múltiplos do período, permitindo que o modelo reconheça padrões que se repetem em intervalos regulares.

Para compreender o impacto do parâmetro lengthscale no kernel periódico, manteve-se o período $p = 6.28$, e variou ℓ em 0.1, 1.0 e 5.0. Logo depois, fixou-se $\ell = 1.0$ e o parâmetro p assumiu os valores 2.0, 3.14, 6.28 e 12.56.

4.4.5 Polynomial

O kernel Polynomial representa uma função de covariância não estacionária que modela correlações por meio de produtos internos ponderados entre pontos de entrada. Este kernel, definido pela equação (4.8), é caracterizado por dois parâmetros fundamentais: c (offset), que controla o termo de viés adicionado ao produto interno, e d (power), que determina o grau polinomial da função de covariância.

$$k(x_i, x_j) = \left(\mathbf{x_i}^T \mathbf{x_j} + c\right)^d \tag{4.8}$$

Uma característica distintiva do kernel Polynomial é sua capacidade de capturar correlações de longo alcance e tendências globais nos dados, especialmente adequada para funções que exibem comportamentos polinomiais intrínsecos. Para $d = 1$, o kernel reduz-se ao kernel linear, enquanto valores maiores de d permitem a modelagem de interações não lineares mais complexas. O parâmetro c desempenha um papel crucial na estabilidade numérica, influenciando diretamente o condicionamento da matriz de covariância resultante.

No entanto, o kernel Polynomial apresentou limitações relacionadas à estabilidade numérica quando utilizado com hiperparâmetros fixos não otimizados, com isso, não foi possível realizar a análise deste kernel. Problemas semelhantes são discutidos por Foster et al. (2009), que demonstram como determinadas escolhas de hiperparâmetros e formulações mal-condicionadas da matriz de covariância podem levar a falhas em decomposições de Cholesky em modelos de Processos Gaussianos, afetando diretamente a precisão das predições.

4.4.6 Linear

O kernel Linear representa a função de covariância mais simples dentre os kernels não estacionários, baseando-se exclusivamente no produto interno entre os pontos de entrada. Este kernel, definido pela equação (4.9), é caracterizado por um único parâmetro v (variance), que controla a magnitude global das correlações.

$$k(x_i, x_j) = v \cdot \mathbf{x_i}^T \mathbf{x_j} \tag{4.9}$$

O kernel Linear representa essencialmente uma regressão linear Bayesiana, sendo o caso mais simples, onde as funções amostradas do GP são estritamente lineares. Esta simplicidade conceitual, entretanto, vem acompanhada de limitações práticas significativas relacionadas à deficiência de posto da matriz de covariância resultante. Diferentemente de kernels estacionários como RBF ou Matérn, que garantem matrizes de covariância positivas definidas, o kernel Linear pode gerar matrizes singulares ou mal condicionadas quando os dados de entrada apresentam dependências lineares ou estão em subespaços de dimensão reduzida.

Uma característica distintiva do kernel Linear é sua capacidade de modelar tendências globais e correlações de longo alcance de forma extremamente eficiente computacionalmente. Por representar apenas o produto interno ponderado, este kernel não decai com a distância entre pontos, permitindo que correlações se mantenham mesmo entre pontos muito distantes no espaço de entrada. Esta propriedade é particularmente útil

para modelar fenômenos que apresentam comportamentos lineares globais, mas pode ser inadequada para funções com variações locais complexas.

No entanto, o kernel Linear, assim como o kernel Polynomial, apresentou limitações relacionadas à estabilidade numérica quando utilizado com hiperparâmetros fixos não otimizados, com isso, não foi possível realizar a análise deste kernel. Como já pontuado, problemas semelhantes são discutidos por Foster et al. (2009), que demonstram que a formulação mal-condicionada da matriz de covariância pode levar a falhas em decomposições de Cholesky e, assim, afetando diretamente a precisão das predições.

4.4.7 Composição de soma e multiplicação

Os kernels podem ser utilizados isoladamente ou em composições de soma e multiplicação para criar funções de covariância mais expressivas e adequadas a problemas específicos. Esta propriedade permite que diferentes características dos dados sejam capturadas simultaneamente através da combinação de kernels com propriedades complementares. Embora as composições ofereçam maior flexibilidade e poder de ajuste, elas também introduzem complexidades adicionais relacionadas à interpretabilidade, otimização de hiperparâmetros e risco de overfitting e underfitting.

A composição de soma possui a capacidade de capturar diferentes aspectos ou escalas temporais/espaciais de uma função simultaneamente. Quando dois kernels são somados, o kernel resultante herda e combina as propriedades de ambos os componentes, permitindo que o modelo capture padrões de forma mais eficiente do que os kernels individuais conseguiriam modelar isoladamente. Entretanto, a composição de soma também apresenta limitações significativas que podem comprometer o desempenho dos modelos. Uma das principais desvantagens é o aumento da complexidade computacional, pois cada componente adicional requer cálculos e otimizações independentes. Além disso, existe o risco de overfitting, uma vez que a composição pode se tornar excessivamente flexível e capturar ruído específico dos dados de treino em vez de padrões generalizáveis da função modelada, comprometendo assim o ajuste e a capacidade preditiva do emulador.

A composição de multiplicação oferece uma abordagem diferente da soma para combinar kernels, pois resulta em correlações significativas apenas quando ambos os componentes simultaneamente indicam alta similaridade entre os dados de entrada. Esta propriedade torna a composição multiplicativa adequada para modelar fenômenos onde múltiplas condições devem ser satisfeitas simultaneamente para que o emulador se ajuste aos dados. A composição multiplicativa, assim como na linear, frequentemente resulta em funções mais suaves que os componentes individuais, o que pode contribuir para gerar predições mais estáveis. Entretanto, a interpretabilidade torna-se consideravelmente mais complexa em comparação com composições aditivas, podendo apresentar comportamento mais restritivo que os componentes individuais e resultar em underfitting, já que a

otimização de parâmetros é mais complexa.

4.5 Implementação dos Processos Gaussianos

Após o estudo dos kernels e da influência de seus parâmetros no ajuste do modelo, foi utilizada a implementação do GPyTorch para criar os Processos Gaussianos (GPs) para avaliar e selecionar os melhores kernels a serem comparados com as Redes Neurais (NNs) e as Expansões de Caos Polinomial PCEs. O código foi desenvolvido em Python, com arquitetura modular para suportar múltiplos kernels sobre os conjuntos de dados simulados do Modelo A e Modelo B. A seguir, detalham-se as etapas de preparo dos dados, configuração de treinamento, obtenção de métricas e hardware utilizado.

4.5.1 Preparo dos dados

As variáveis de entrada foram normalizadas com StandardScaler, transformando-as em distribuição normal com média zero e desvio padrão unitário. Essa normalização foi aplicada ao conjunto de treino e de teste tanto para os Processos Gaussianos quanto para os demais emuladores analisados. As quatro variáveis de saída de cada modelo também foram normalizadas antes do treinamento e reescaladas após a predição para cálculo das métricas.

Para cada uma das quatro variáveis de saída foi treinado um GP distinto. Em cada replicação experimental, os dados de treino eram amostrados aleatoriamente em incrementos de cinquenta observações, de 50 até 5000, totalizando 100 tamanhos amostrais diferentes. Para garantir comparabilidade entre kernels, utilizou-se sementes determinísticas, dessa forma, em cada uma das cinco repetições, todos os kernels recebiam o mesmo subconjunto de treino, definido por uma semente específica.

4.5.2 Configurações de treinamento

Foram realizadas cinco repetições para cada combinação de tamanho amostral e kernel, tanto para o Modelo A quanto para o Modelo B, totalizando 4000 experimentos por kernel. Os experimentos foram executados de forma sequencial, a fim de evitar a concorrência pelos recursos disponíveis, garantindo que, durante a execução, nenhum outro processo fosse executado no hardware.

O treinamento de cada GP foi realizado com o otimizador Adam (KINGMA, 2014) e learning rate de 0.01, por 100 iterações. A função objetivo foi a log-verossimilhança marginal exata. Além disso, Automatic Relevance Determination (ARD) foi habilitado em todos os kernels, atribuindo um lengthscale para cada dimensão de entrada, sendo três parâmetros para o Modelo A e doze para o Modelo B. A limpeza de cache da GPU ocorria ao final de cada experimento para evitar acúmulo de alocações anteriores.

Durante a inferência, das 100 mil observações, foram feitos em lotes fixos para evitar estouro de memória, garantindo eficiência sem comprometer a precisão das predições.

4.5.3 Obtenção das métricas e armazenamento dos resultados

Para cada combinação de kernel, tamanho amostral e variável de saída, foram coletadas as seguintes métricas:

- Erro quadrático médio (MSE) entre valores reais e preditos;
- tempo de treinamento, em segundos;
- taxa de inferência, sendo o número de pontos preditos por segundo;
- pico de uso de memória GPU, em MB.

O consumo de memória GPU foi monitorado a uma taxa de 100 Hz utilizando estatísticas nativas do PyTorch, e apenas o valor máximo foi registrado.

Ao final dos experimentos, todos os resultados foram salvos em arquivos CSV para análise estatística posterior.

4.5.4 Hardware utilizado

Os experimentos foram conduzidos em um computador pessoal com as seguintes configurações:

- CPU Intel Core i5-12600K (Alder Lake): 6 núcleos de performance até 4.9 GHz, 4 núcleos de eficiência até 3.6 GHz, 16 threads, 20 MB de cache L3;
- GPU Gigabyte GeForce RTX 3060 12 GB GDDR6 (Ampere, clock base 1837 MHz, barramento 192 bits, 360 GB/s de banda);
- 16 GB RAM DDR4-3100 MT/s em dual-channel;
- SSD Corsair MP600 Gen4 1 TB NVMe PCIe 4.0 $\times$4 (leitura até 4950 MB/s, escrita até 4250 MB/s);
- Sistema operacional Windows 11 com CUDA 11.5.

Este ambiente de hardware garantiu velocidade de cálculo e acesso rápido aos dados, permitindo comparações consistentes de desempenho entre os diferentes kernels.

4.6 Comparativo dos emuladores

O comparativo com os emuladores seguiu passos semelhantes aos utilizados na análise dos kernels, porém com critérios de otimização distintos. No caso dos emuladores, a otimização foi realizada de forma mais criteriosa e refinada, o que resultou em um tempo de treino superior ao observado no comparativo entre os kernels. Foram avaliadas a acurácia e a eficácia dos Processos Gaussianos (GPs) utilizando as bibliotecas scikit-learn e GPyTorch, das Redes Neurais (NNs) implementadas com a biblioteca torch, e das Expansões em Caos Polinomial (PCEs) com a biblioteca chaospy. A acurácia foi mensurada por meio do erro quadrático médio (MSE), enquanto a eficácia considerou o tempo de treino e o tempo de inferência, sendo que cada emulador foi analisado em três níveis de complexidade: baixa, média e alta.

As análises foram realizadas separadamente para cada uma das quatro quantidades de interesse, para os dois modelos e em quatro conjuntos de dados de treino diferentes, contendo 100, 500, 1000 e 5000 dados. Cada conjunto de treino foi criado a partir de uma amostragem por LHS e os dados foram simulados, para garantir que todo o espaço paramétrico estaria representado, retirando a incerteza de uma amostragem aleatória, para prevalecer nos resultados a diferença de cada emulador, reduzindo efeitos aleatórios.

Depois, cada um dos quatro diferentes emuladores, com três complexidades cada, foi treinado, totalizando 12 treinos para cada um dos tamanhos amostrais. O tempo necessário para realizar essa etapa foi medido e unido com a métrica de MSE e o tempo necessário para realizar a inferência de 100 mil dados. Além dessas três métricas, também foi mensurada a quantidade de memória alocada na GPU no momento da inferência, porém somente dois emuladores utilizam GPU, GPs utilizando a biblioteca GPyTorch e as NNs, sendo assim, não é possível utilizá-la no comparativo com os demais emuladores.

Os resultados das métricas das quantidades de interesse (QoI) serão apresentados na seção 5.5. Nos casos em que os valores individuais apresentaram pouca variação entre si, foi utilizada a média das observações para a construção dos gráficos. Assim, as métricas de memória alocada na GPU, tempo de inferência e tempo de treino foram representadas por meio de gráficos com as médias obtidas pelas QoIs. Já para a métrica MSE, os valores foram normalizados, dividindo-se cada valor pelo maior MSE observado em sua respectiva QoI, e os resultados foram apresentados em gráficos específicos para cada QoI, para permitir uma análise comparativa entre as variáveis.

Posteriormente, foi realizada uma análise comparativa do erro absoluto entre os valores preditos e os valores reais para o melhor e o pior desempenho de cada tipo de emulador avaliado (NN, GP com GPyTorch, GP com scikit-learn e PCE), totalizando oito casos analisados. O objetivo dessa etapa foi identificar as regiões do domínio em que ocorreram os maiores erros, evidenciando os pontos em que cada emulador apresentou maior dificuldade em realizar as predições.

4.7 Score de Pareto para comparar acurácia e eficácia

Após a análise dos diferentes kernels e do comparativo com os emuladores, tornou-se necessário estabelecer uma métrica única que possibilitasse a comparação direta entre eles e, assim, permitisse identificar o melhor kernel e o melhor emulador para cada uma das quantidades de interesse.

Para isso, foi desenvolvido um método estatístico baseado em um score de Pareto, calculado por meio de um processo de normalização e ponderação em duas etapas.

Na primeira etapa, foram invertidas as métricas cujo desempenho é melhor quanto menor o valor obtido. A transformação utilizada foi:

$$x' = \frac{1}{x + 1 \times 10^{-8}} \tag{4.10}$$

O termo residual (1×10^{-8}), presente na equação (4.10) foi adicionado para evitar divisões por zero.

Para a comparação entre kernels, foram invertidas as métricas MSE, tempo de treino e uso de memória GPU. A quarta métrica, pontos inferidos por segundo, não precisou ser invertida, pois já possui interpretação direta, em que valores maiores indicam melhor desempenho.

Para a comparação entre emuladores, todas as métricas utilizadas no cálculo do score foram invertidas, sendo elas: MSE, tempo de treino e tempo necessário para realizar a inferência de 100 mil dados.

Na segunda etapa, todos os valores resultantes foram normalizados para o intervalo entre 0 e 1 por meio da transformação MinMaxScaler:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{4.11}$$

Nessa normalização, o maior valor assume nota 1 e o menor valor assume nota 0. Por fim, o score final de cada kernel e emulador foi calculado como a média simples entre as notas normalizadas obtidas em cada métrica.

Com o objetivo de verificar se as notas atribuídas aos kernels e emuladores apresentaram diferenças estatisticamente significativas, foram utilizados testes não paramétricos. A escolha dessa abordagem foi necessária, pois a ausência de normalidade nos dados e a falta de homogeneidade das variâncias impediram a aplicação de testes paramétricos, como a ANOVA seguida do teste de Tukey (MONTGOMERY, 2017).

Dessa forma, foi aplicado o teste de Kruskal-Wallis (KRUSKAL; WALLIS, 1952), que é um teste não paramétrico utilizado para comparar três ou mais grupos independentes (GIBBONS; CHAKRABORTI, 2014). Este teste verifica se há diferenças significativas

na nota dos grupos, indicando se pelo menos um grupo é estatisticamente diferente dos outros. Quando o teste de Kruskal-Wallis indicou diferenças significativas, foi seguido do teste post-hoc de Dunn com correção de Bonferroni (DUNN, 1964) para identificar quais grupos apresentaram diferenças significativas entre si. Foi utilizado o valor $\alpha = 0.05$. Desse modo, kernels e emuladores com diferenças superiores ao limiar crítico calculado são considerados estatisticamente diferentes e classificados em grupos distintos, representados por letras.

5 RESULTADOS E DISCUSSÃO

5.1 Análise exploratória dos dados

A análise exploratória de dados tem como objetivo analisar características dos dados. Neste caso, vamos investigar aqueles que serão emulados, facilitando assim a escolha dos kernels e a compreensão das correlações entre as variáveis de entrada e saída.

5.1.1 Modelo A

Inicialmente, os dados das três variáveis de entrada do Modelo A seguem uma distribuição uniforme entre zero e um, o que indica que o espaço amostral está igualmente representado. Esta característica de uniformidade é fundamental para garantir uma exploração adequada do espaço paramétrico do modelo eletrofisiológico, permitindo que todas as combinações possíveis de valores sejam igualmente consideradas durante o processo de amostragem. Ressalta-se também que não há qualquer tipo de correlação entre essas variáveis.

Prosseguindo na análise dos dados obtidos pelo modelo eletrofisiológico, a observação da Figura 4 revela aspectos fundamentais das correlações entre as variáveis de saída do Modelo A. A matriz de correlação apresenta padrões que refletem as interações das quantidades de interesse extraídas do potencial de ação cardíaco. A correlação mais notória ocorre entre APD90 e APD50, evidenciando a forte dependência temporal entre diferentes momentos da repolarização. Esta relação perfeita indica que ambas as medidas de duração do potencial de ação variam de forma completamente sincronizada, o que é esperado, já que representam marcos temporais consecutivos da repolarização.

Em contraste com a correlação fortemente positiva, observa-se uma forte correlação negativa entre dVmax e Vreps, revelando uma relação inversa quase perfeita entre a taxa máxima de despolarização e o potencial de repouso. As correlações moderadas entre dVmax e as variáveis APD90 e APD50 indicam acoplamento parcial entre os processos de despolarização e repolarização.

Observando a Figura 5, nota-se que a variável dVmax exibe uma distribuição assimétrica positiva, com um leve platô nos valores centrais do gráfico e uma cauda estendida em direção a valores mais elevados. Esta distribuição reflete a natureza não-linear da despolarização cardíaca. As variáveis APD50 e APD90 apresentam distribuições aproximadamente normais, logo, são simétricas e unimodais. Por fim, a variável Vreps apresenta uma distribuição com formato ligeiramente assimétrico negativo.

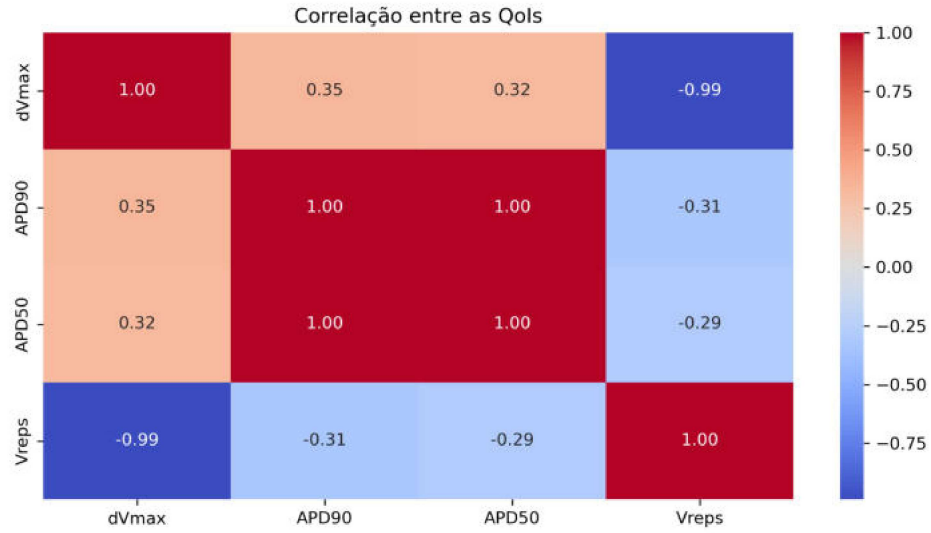

Figura 4 – Gráfico de correlação entre as quatro quantidades de interesse (QoI) do Modelo A.

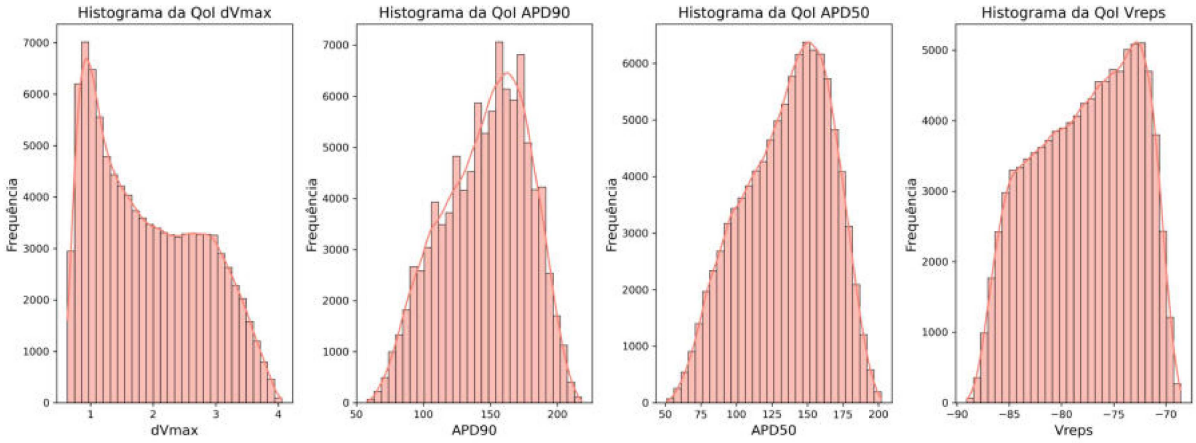

Figura 5 – Histograma das quatro quantidades de interesse (QoI) de saída do Modelo A.

Analisando as correlações entre as variáveis de entrada e saída apresentadas na Figura 6, observa-se uma estrutura complexa de dependências e a análise conjunta dos gráficos de dispersão na Figura 7 complementa essa informação, fornecendo insights visuais sobre as relações entre entradas e saídas. A variável Hipóxia demonstra correlações fracas com as variáveis de saída dVmax ($r = 0.08$) e Vreps ($r = -0.09$), mas possui uma forte correlação negativa com as quantidades de interesse da repolarização APD50 ($r = -0.88$) e APD90 ($r = -0.87$).

Em contraste, a Hipercalemia apresenta correlações não tão significativas com APD50 ($r = -0.33$) e APD90 ($r = -0.36$), e altamente significativas com Vreps ($r = 0.98$)

e dVmax ($r = -0.95$). A variável Acidose apresenta correlações similares com todas as variáveis de saída, dVmax ($r = -0.26$), APD50 ($r = -0.29$), APD90 ($r = -0.28$) e Vreps ($r = 0.16$), sugerindo efeitos moderados, mas consistentes sobre o comportamento eletrofisiológico.

As correlações apresentadas nos parágrafos anteriores, e também nas Figuras 6 e 7, reforçam as considerações feitas na subseção 3.4.1 da modelagem eletrofisiológica cardíaca deste trabalho. Observa-se que a hipercalemia está fortemente associada ao potencial de repouso (Vreps) e à redução da velocidade máxima de despolarização dVmax, em concordância com o mecanismo biológico, onde o aumento da concentração extracelular de potássio leva à inativação dos canais de sódio, afetando as duas QoIs citadas. Por sua vez, a hipóxia possui forte correlação com a redução da duração do potencial de ação APD50 e APD90, devido à ativação de canais de potássio sensíveis ao ATP, que aceleram a repolarização. A acidose, por fim, mostra correlações mais discretas, mas relativamente consistentes com todas as variáveis de saída, sugerindo um efeito difuso e moderado sobre o comportamento eletrofisiológico global.

hipercalemia causa despolarização do potencial de membrana de repouso

Os gráficos de dispersão revelam relações predominantemente lineares para a maioria das combinações entrada-saída, com algumas evidências de não-linearidades sutis, particularmente nas interações das variáveis Hipóxia e Hipercalemia com as variáveis de saída, onde se observam leves curvaturas. Essas características sugerem que kernels não-lineares serão essenciais para capturar adequadamente essas interações, como RBF (Radial Basis Function), Matérn com parâmetros de suavidade apropriados ($\nu = 3/2$ ou $5/2$) ou polinomiais de grau moderado.

Aliada a uma boa escolha de kernel, a implementação de técnicas de *Automatic Relevance Determination* (ARD) será valiosa para identificar quais parâmetros de entrada influenciam preferencialmente cada variável de saída, considerando suas interdependências. O ARD é uma técnica que atribui parâmetros de escala de comprimento (lengthscale) individuais para cada dimensão de entrada no kernel, permitindo que o modelo aprenda automaticamente a relevância relativa de cada variável de entrada. Ao invés de usar um único lengthscale global, o ARD utiliza um vetor de lengthscales $\ell = [\ell_1, \ldots, \ell_d]$, onde cada ℓ_i corresponde à dimensão i do espaço de entrada. Valores menores de ℓ_i indicam que a variável i é mais relevante para a predição, enquanto valores maiores sugerem menor relevância. Como é evidenciado pelas Figuras 6 e 7, há diversas relações entre as variáveis que são fortes e outras que são fracas, ponderar isso com o ARD auxiliará o ajuste do kernel.

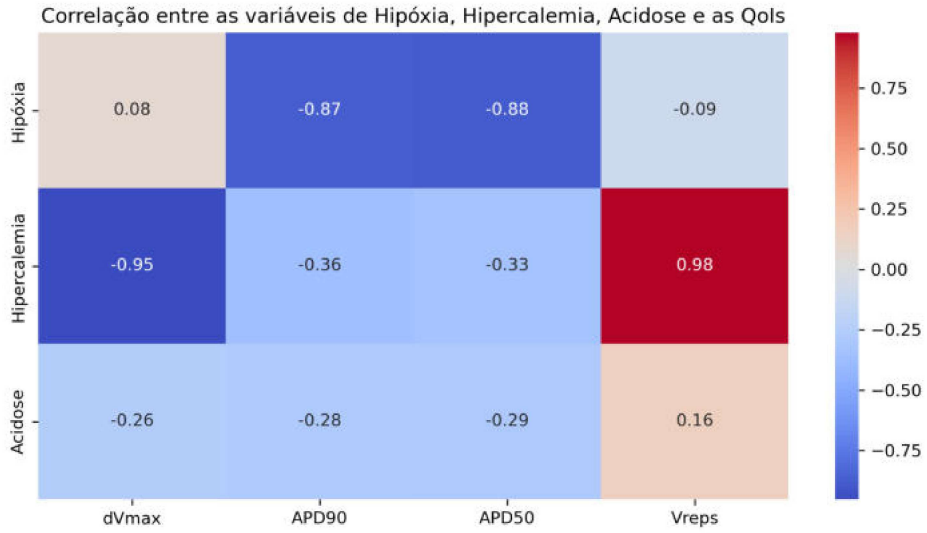

Figura 6 – Gráfico de correlação entre as variáveis de entrada e as quantidades de interesse (QoIs) do Modelo A.

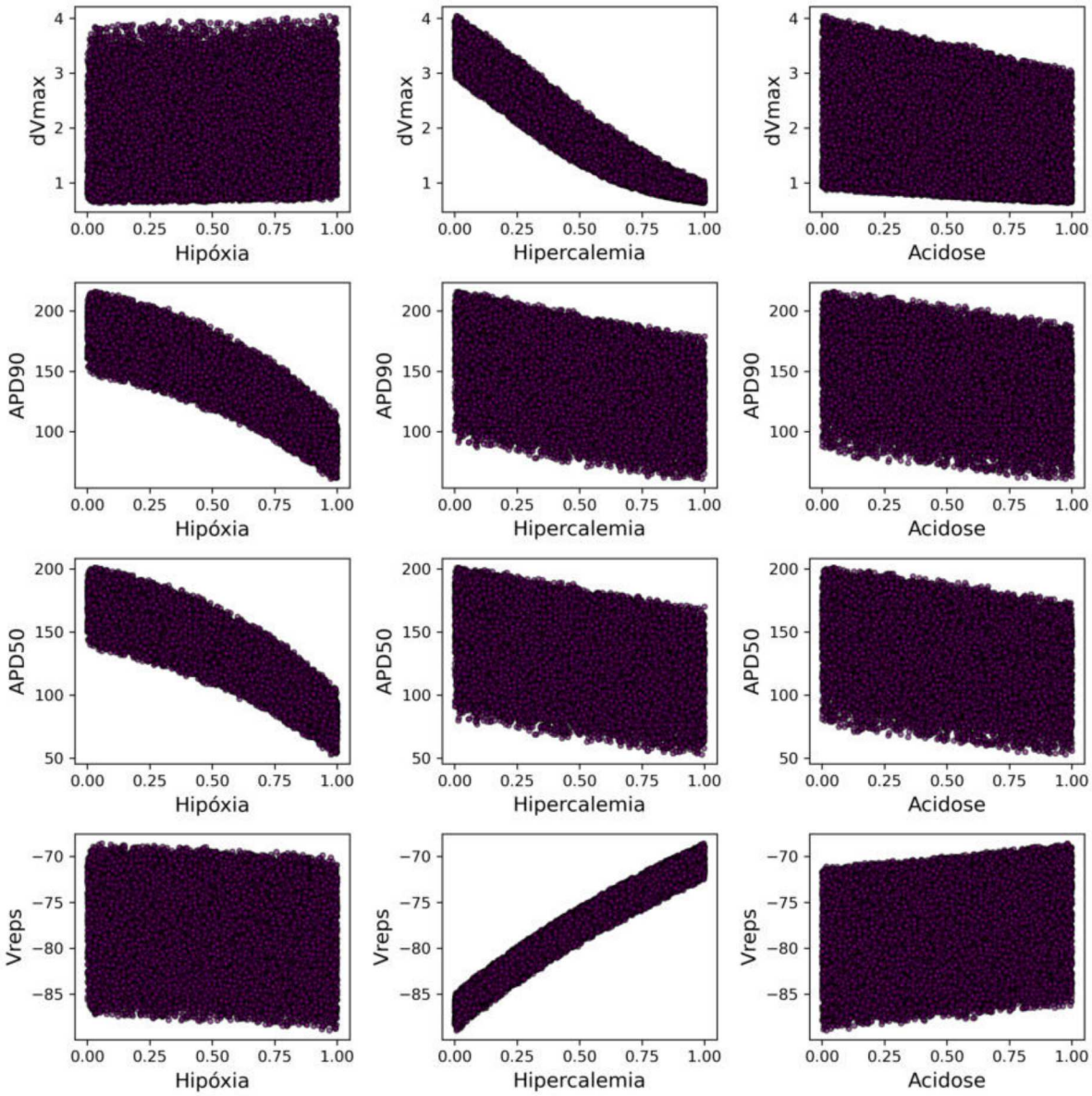

Figura 7 – Gráficos de dispersão com a correlação entre as variáveis de entrada e saída do Modelo A.

5.1.2 Modelo B

Inicialmente, semelhante ao Modelo A, os dados de entrada das variáveis do modelo isquêmico seguem uma distribuição uniforme entre 0 e 1, e as variáveis do modelo de diversidade celular seguem uma distribuição uniforme entre 0.5 e 1.5 indicando que o espaço amostral está igualmente representado, o que confere aos dados uma exploração adequada do espaço paramétrico do modelo eletrofisiológico, permitindo que todas as combinações possíveis de valores sejam igualmente consideradas durante o processo de amostragem. Ressalta-se também que não há qualquer tipo de correlação entre essas variáveis.

Prosseguindo na análise dos dados, a observação da Figura 8 se assemelha à Figura 4, que representa os dados do Modelo A. Novamente, é evidenciada a correlação entre APD90 e APD50, evidenciando a forte dependência temporal entre diferentes momentos da repolarização. Nota-se também a forte correlação negativa entre dVmax e Vreps e as correlações moderadas entre dVmax e as variáveis APD90 e APD50.

Observando a Figura 9, nota-se uma pequena diferença se comparado à saída do Modelo A, mostrado na Figura 5. A variável dVmax exibe uma distribuição assimétrica positiva, mas no Modelo B não apresenta um leve platô nos valores centrais do gráfico, apenas a cauda estendida em direção a valores mais elevados se mantém semelhante. As variáveis APD90 e APD50 possuem distribuições aproximadamente normais, sendo mais simétricas do que as distribuições apresentadas por estas variáveis no Modelo A. Por fim, a variável Vreps apresenta uma distribuição concentrada em uma faixa estreita, com formato ligeiramente assimétrico negativo, igual ao encontrado na análise do primeiro modelo.

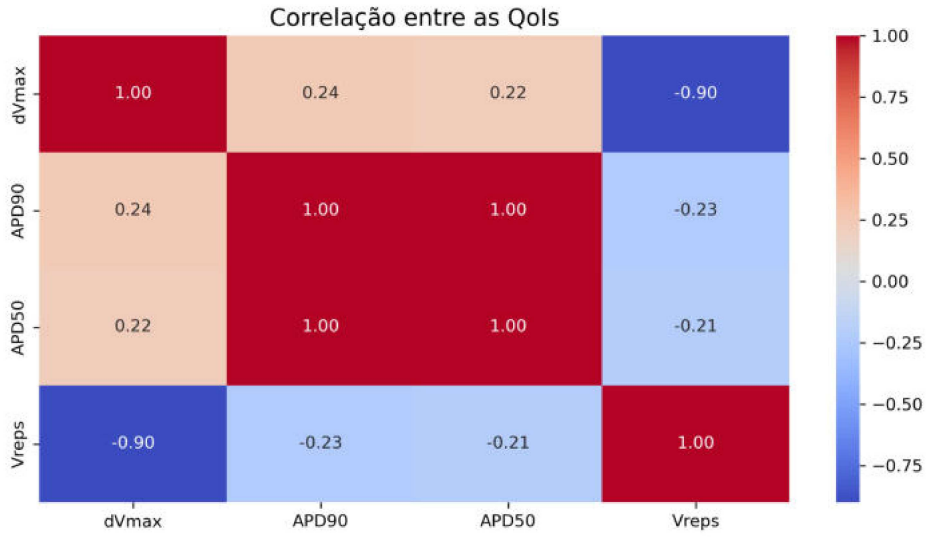

Figura 8 – Gráfico de correlação entre as quatro quantidades de interesse (QoI) do Modelo B.

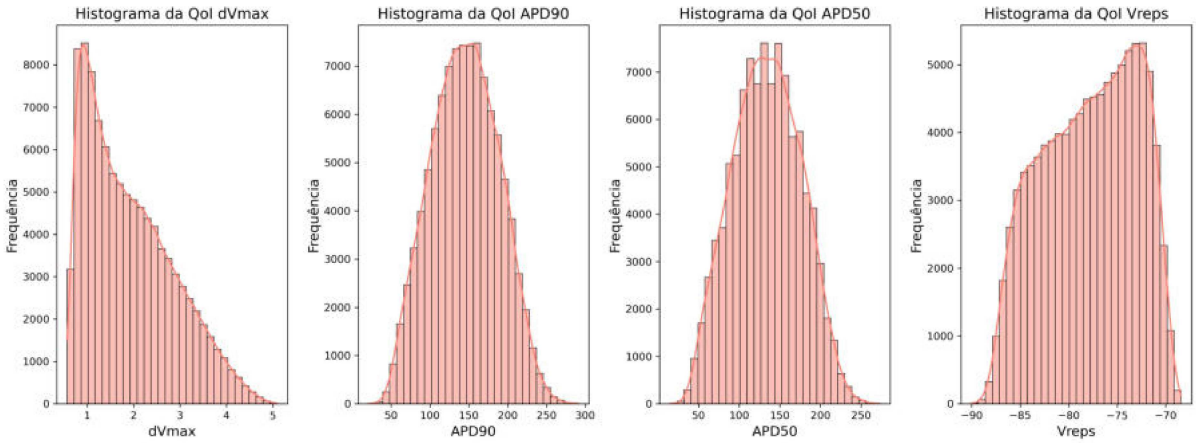

Figura 9 – Histograma das quatro quantidades de interesse (QoI) do Modelo B.

Analisando as correlações entre as variáveis de entrada e saída apresentadas na Figura 10, observa-se uma relação um pouco mais complexa do que a apresentada pelo Modelo A na Figura 6, e a análise conjunta dos gráficos de dispersão nas Figuras 11 e 12 complementa essa visão.

A variável Hipóxia demonstra correlações fracas com as variáveis de saída dVmax ($r = 0.08$) e Vreps ($r = -0.11$), mas possui uma correlação negativa com as quantidades de interesse da repolarização APD50 ($r = -0.63$) e APD90 ($r = -0.63$). Algo similar ocorre com a variável g_{CaLC}, demonstrando correlação significativa apenas com APD50 ($r = 0.63$) e APD90 ($r = 0.65$).

Em contraste, a Hipercalemia apresenta correlações pouco significativas com APD50 ($r = -0.24$) e APD90 ($r = -0.27$), e altamente significativas com Vreps ($r = 0.98$) e dVmax ($r = -0.87$). Já a variável g_{NaC} exibe uma correlação proeminente apenas com dVmax ($r = 0.36$). Finalizando as variáveis que possuem correlações mais relevantes, tem-se a Acidose, que apresenta grau de correlações semelhantes com todas as variáveis de saída, dVmax ($r = -0.24$), APD50 ($r = -0.24$), APD90 ($r = -0.27$) e Vreps ($r = 0.17$). As demais variáveis de entrada não apresentaram correlações significativas com as variáveis de saída, sendo assim, não há necessidade de evidenciá-las.

As correlações expostas pelos parágrafos acima e pelas Figuras 10, 11 e 12 corroboram as análises feitas na subseção 3.5.1, onde foi demonstrado que os parâmetros isquêmicos mantêm sua dominância sobre a diversidade celular. Entre os parâmetros de diversidade celular, g_{CaLC} apresenta as correlações mais expressivas com APD50 e APD90, refletindo seu papel na repolarização tardia, enquanto g_{NaC} mostra correlação moderada apenas com dVmax, confirmando que a variabilidade individual tem impacto limitado comparado aos mecanismos isquêmicos.

Os gráficos de dispersão com as correlações mais relevantes, Figuras 11 e 12, revelam

relações predominantemente lineares para a maioria das combinações entrada-saída que apresentam um certo grau de correlação, com algumas evidências de não-linearidades sutis, particularmente nas interações das variáveis Hipóxia, Hipercalemia e g_{CaLC} com as variáveis de saída, onde se observam leves curvaturas. Essas características sugerem que kernels não-lineares serão essenciais para capturar adequadamente essas interações, como RBF (Radial Basis Function), Matérn com parâmetros de suavidade apropriados ($\nu = 3/2$ ou $5/2$) ou polinomiais de grau moderado.

A mesma análise do Modelo A vale para o Modelo B no quesito de implementar ARD nos kernels, pois ela será valiosa para identificar quais parâmetros de entrada influenciam preferencialmente cada variável de saída, considerando suas interdependências. Como é evidenciado pelas Figuras 10, 11 e 12, há diversas relações entre as variáveis que são fortes e outras que são fracas, ponderar isso auxiliará muito o kernel no ajuste.

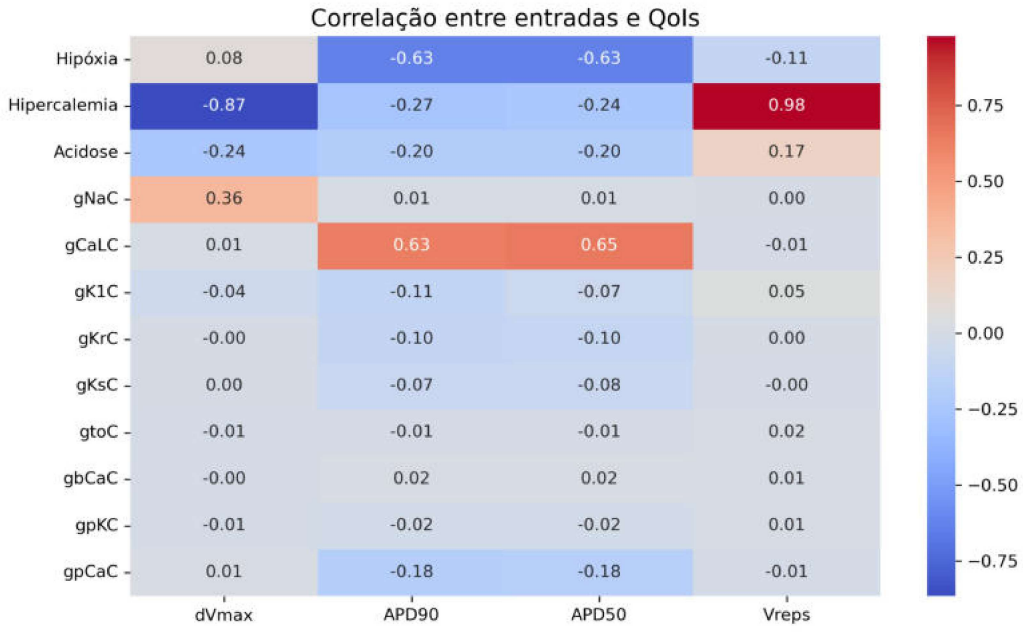

Figura 10 – Gráfico de correlação entre as variáveis de entrada e as quantidades de interesse (QoI) do Modelo B.

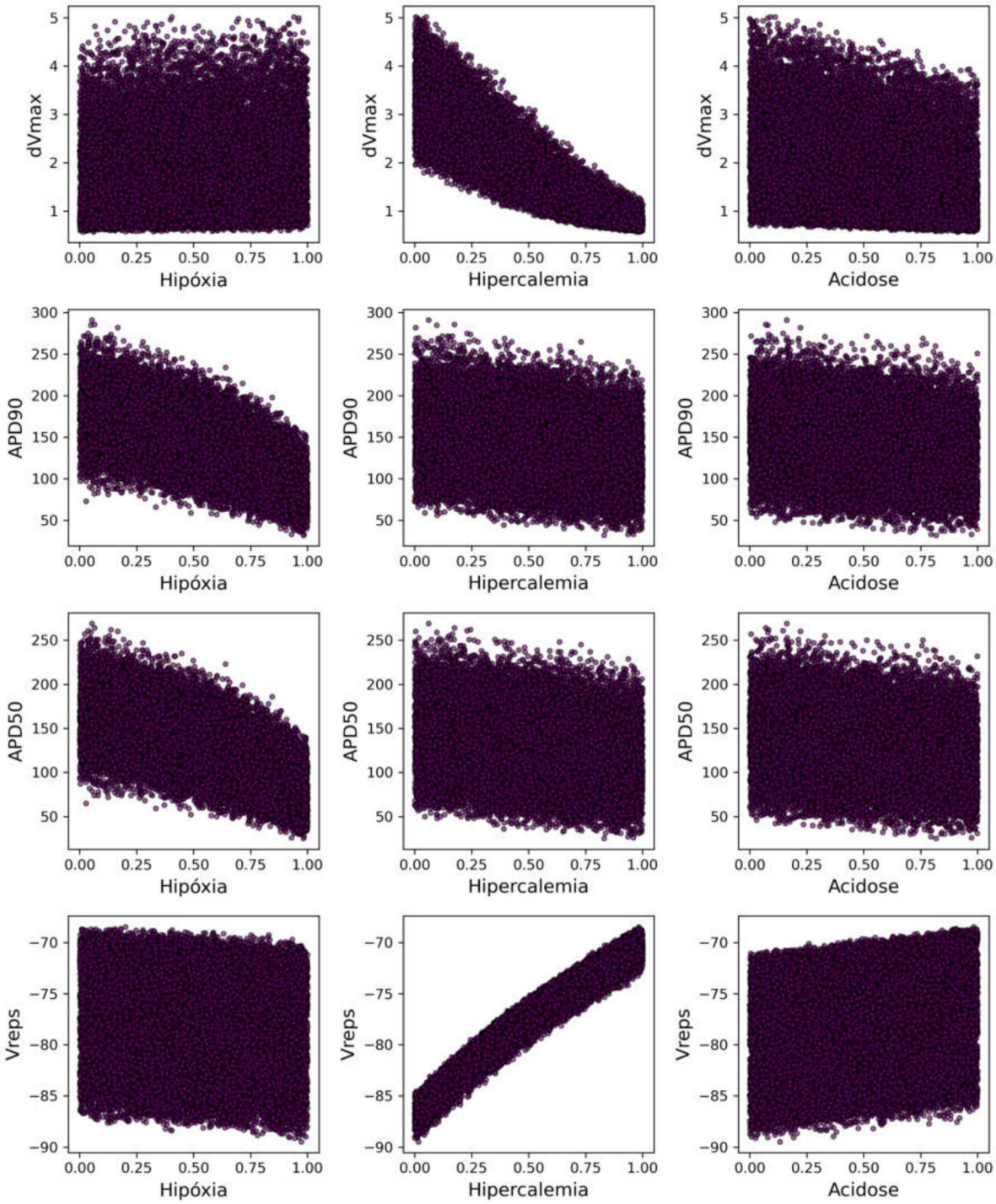

Figura 11 – Primeira parte dos gráficos de dispersão contendo algumas correlações entre as variáveis de entrada e saída do Modelo B.

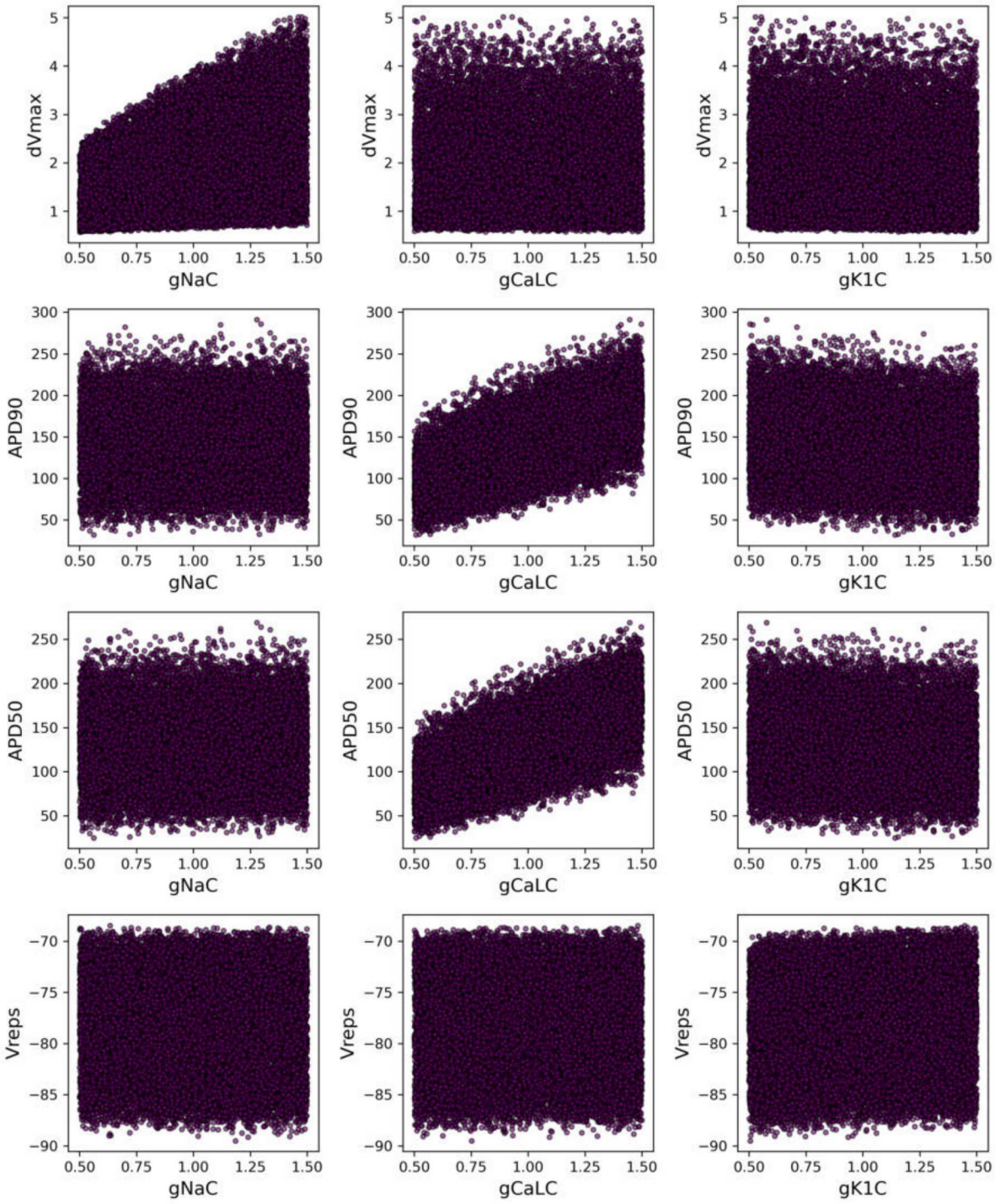

Figura 12 – Segunda parte dos gráficos de dispersão contendo algumas correlações entre as variáveis de entrada e saída do Modelo B.

5.2 Análise dos efeitos dos parâmetros dos kernels

Ao realizar as análises dos parâmetros dos kernels utilizando a função $f(x) = x \cdot sen(x)$, os resultados obtidos permitiram compreender como cada parâmetro influencia o comportamento e a capacidade de ajuste dos diferentes kernels. Os testes realizados

evidenciaram padrões consistentes de overfitting e underfitting conforme os parâmetros foram variados. Essa análise, juntamente com a análise exploratória de dados, forneceu perspectivas valiosas para selecionar adequadamente os kernels que foram avaliados nos Modelos A e B.

5.2.1 Radial basis function

Os resultados da análise do kernel Radial Basis Function (RBF) foram evidenciados na Figura 13. O título expõe o valor do R^2 obtido pela predição daquele GP e também os valor fixo do parâmetro alvo dessa análise.

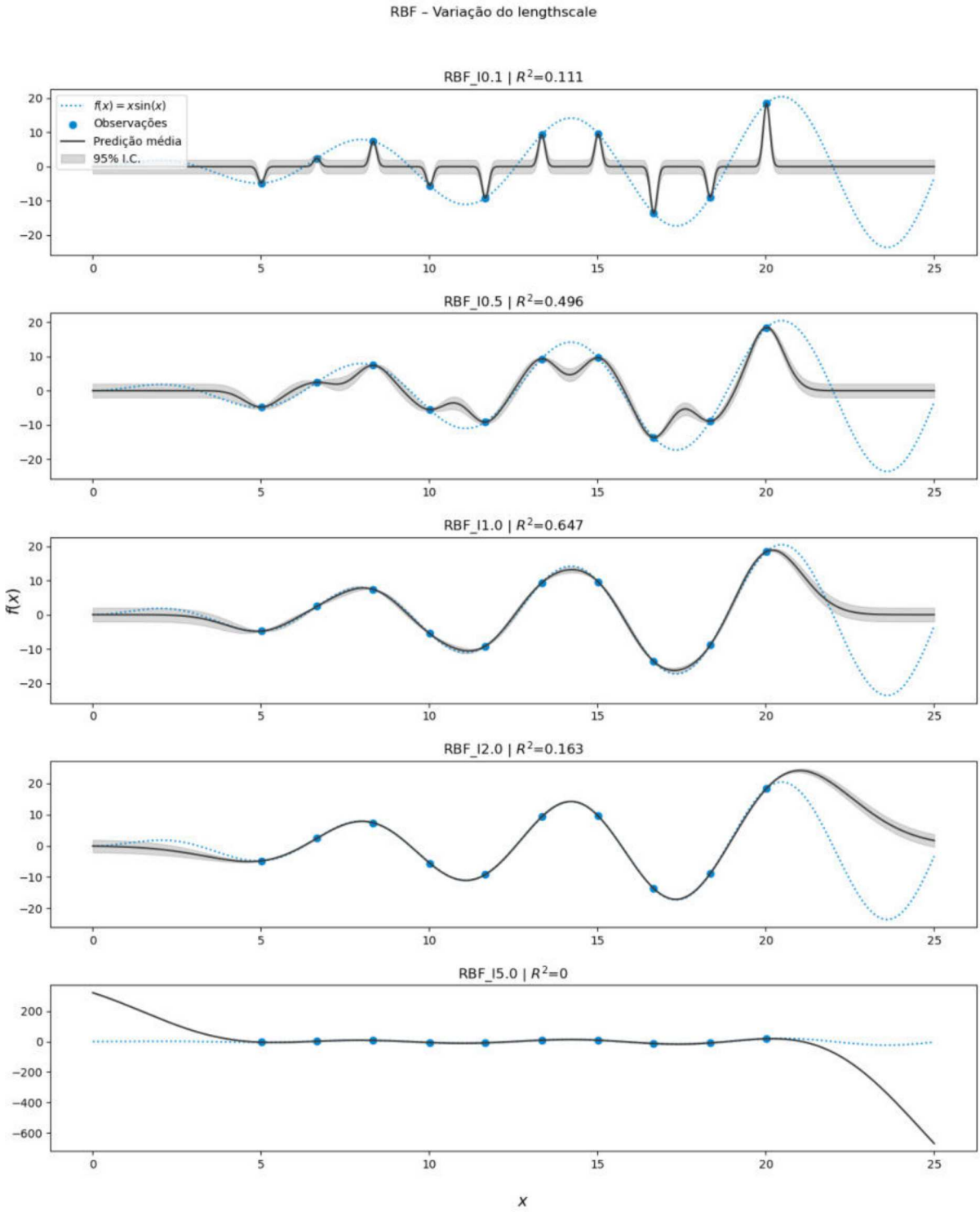

Figura 13 – Comparação entre cinco valores do parâmetro lengthscale (A) $\ell = 0.1$, (B) $\ell = 0.5$, (C) $\ell = 1.0$, (D) $\ell = 2.0$, (E) $\ell = 5.0$, para o kernel Radial Basis Function (RBF).

Observando a Figura 13, nota-se claramente o impacto do parâmetro lengthscale no ajuste do modelo. Quando $\ell = 0.1$, o kernel não compreende como os dados se correlacionam, onde $R^2 = 0.111$ evidencia isso. O kernel não consegue capturar adequadamente o

comportamento global da função e, com isso, gera inferências totalmente desconexas com os valores reais, um exemplo claro de overfitting (quando o modelo se ajusta excessivamente aos dados de treinamento, perdendo capacidade de generalização).

Para $\ell = 0.5$, observa-se uma melhoria significativa no ajuste, com $R^2 = 0.496$. O modelo começa a compreender as variações da função, embora ainda apresente algumas oscilações desnecessárias, demonstrando resquícios de um overfitting.

O desempenho alcançado com $\ell = 1.0$, resultando em $R^2 = 0.647$, demonstra que o kernel compreendeu bem o comportamento da função, tanto que foi o melhor ajuste mostrado pela Figura 13. Isso demonstra que a correlação entre os valores está se otimizando, fazendo com que o kernel compreenda melhor características locais e globais da função estudada.

Quando $\ell = 2.0$, o modelo apresenta $R^2 = 0.163$, uma queda abrupta do ajuste, demonstrando que valores altos de lengthscales geram correlações entre muitos pontos distantes, desnecessariamente. Isso confunde o kernel e faz com que seja gerado um underfitting (quando o modelo não possui flexibilidade suficiente para capturar a variabilidade dos dados). Além disso, valores elevados de lengthscale podem resultar em problemas de predição nas bordas do domínio de entrada, onde há escassez de pontos de treinamento.

Finalmente, para $\ell = 5.0$, o $R^2 = 0.0$ demonstra que o modelo produz predições excessivamente suaves, evidenciando, novamente, um grau de correlação entre os pontos além do que é necessário, intensificando o underfitting.

Com o final da análise do kernel RBF, concluímos que, para a função estudada, valores intermediários de lengthscales (próximos a 1.0) oferecem a melhor relação entre capacidade de ajuste e generalização. Dessa forma, valores muito pequenos de ℓ levam a modelos onde cada amostra de treino influencia uma região estreita que oscila fortemente entre os pontos de treino, causando um overfitting. Em contrapartida, valores muito grandes resultam em modelos cuja covariância decai lentamente, com isso, pontos distantes continuam altamente correlacionados, não conseguindo capturar variações locais, causando underfitting.

5.2.2 Rational quadratic

Para compreender o impacto do parâmetro α no kernel Rational Quadratic (RQ), foram testados diferentes valores fixos, mantendo o parâmetro $\ell = 1.0$, evidenciados nos títulos dos gráficos presentes na Figura 14. O título também expõe o valor do R^2 obtido pela predição daquele GP.

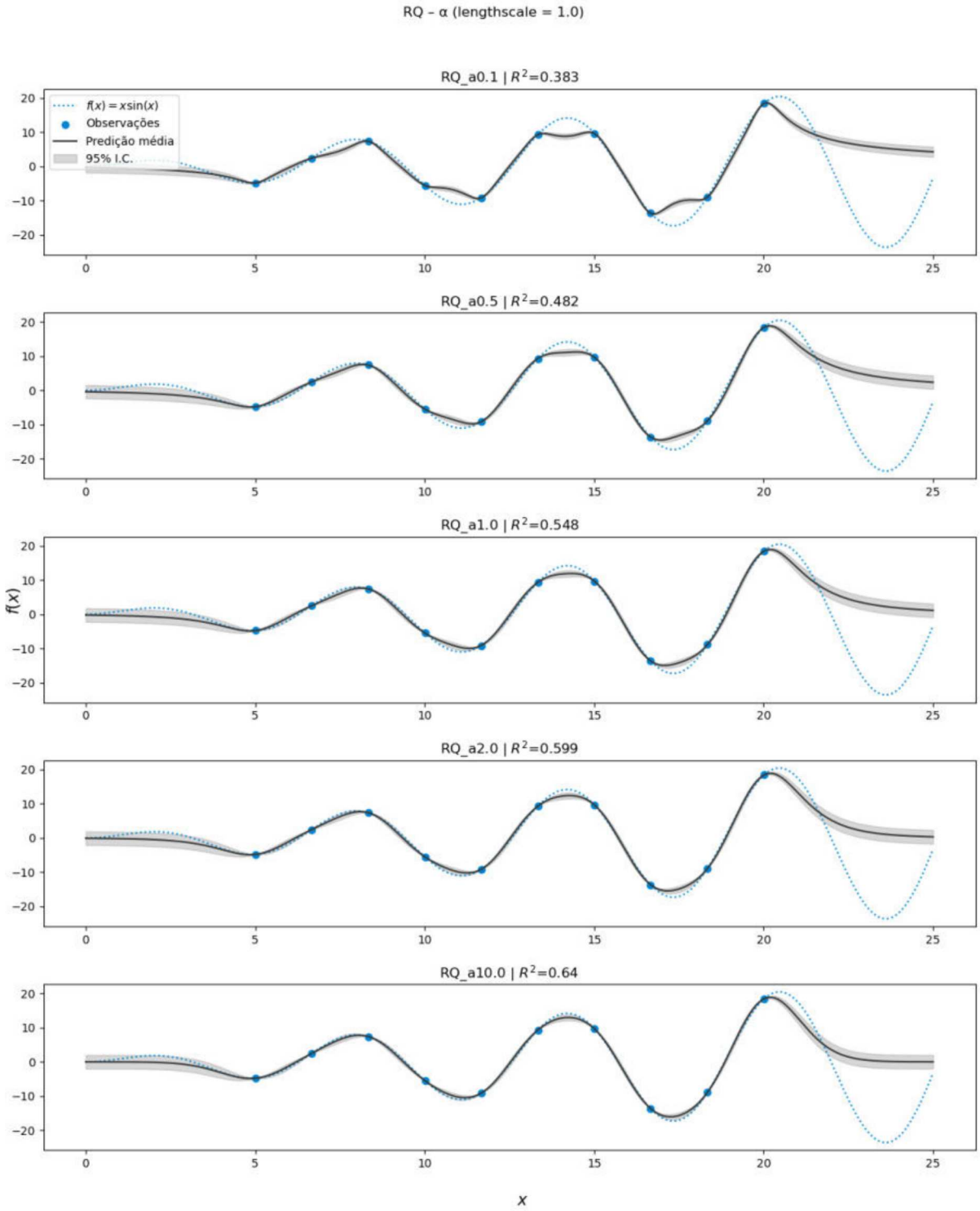

Figura 14 – Comparação entre cinco valores do parâmetro α (A) $\alpha = 0.1$, (B) $\alpha = 0.5$, (C) $\alpha = 1.0$, (D) $\alpha = 2.0$, (E) $\alpha = 10.0$ para o kernel Rational Quadratic (RQ) com lengthscale fixo ($\ell = 1.0$).

Observando a Figura 14, nota-se como o parâmetro α afeta o comportamento do kernel RQ. Quando $\alpha = 0.1$, o kernel apresenta $R^2 = 0.383$, demonstrando um ajuste moderado. Com este valor baixo de α, o kernel gera funções com caudas pesadas que

capturam correlações de longo alcance, mas também introduzem maior variabilidade nas predições, resultando em oscilações que, embora capturem algumas características da função $f(x) = x \cdot sen(x)$, não proporcionam a suavidade ideal, sendo possível observar um leve overfitting.

Para $\alpha = 0.5$, observa-se uma melhoria no ajuste, com $R^2 = 0.482$. O aumento do parâmetro α reduz a influência das caudas pesadas, deixando o kernel mais estável, melhorando a captura do padrão oscilatório da função. A correlação entre pontos distantes diminui adequadamente, permitindo maior fidelidade às variações locais.

O desempenho com $\alpha = 1.0$ resulta em $R^2 = 0.548$, demonstrando uma melhora no equilíbrio. O kernel começa a capturar as características locais e o comportamento global da função senoidal, favorecendo o GP a ter predições mais suaves do que os valores de α apresentados anteriormente.

Quando $\alpha = 2.0$, o modelo apresenta $R^2 = 0.599$, mostrando que à medida que o valor do parâmetro aumenta, o ajuste fica melhor. Neste ponto, o kernel RQ se aproxima mais do comportamento do RBF, mantendo correlações apropriadas entre pontos próximos, enquanto reduz adequadamente as correlações de longo alcance, melhorando a captura das oscilações senoidais.

Para $\alpha = 10.0$, obtém-se $R^2 = 0.640$, evidenciando a tendência de que um maior valor do parâmetro deixa a função mais suave e estável, para uma função senoidal. Neste estado, o kernel RQ se aproxima do comportamento RBF.

Em seguida, analisou-se a influência do parâmetro ℓ no ajuste do kernel RQ, fixando $\alpha = 1.0$, conforme apresentado na Figura 15.

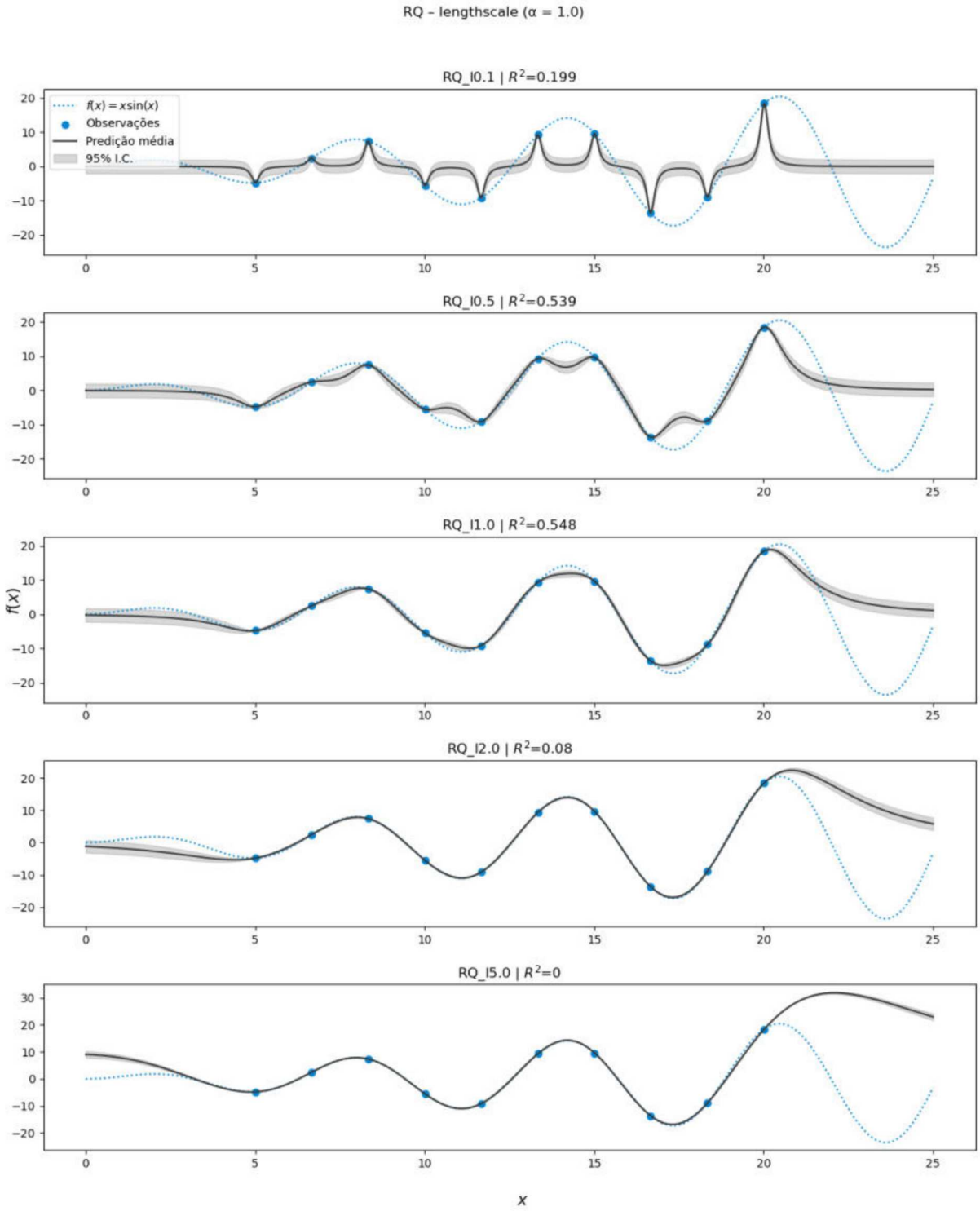

Figura 15 – Comparação entre cinco valores do parâmetro lengthscale (A) $\ell = 0.1$, (B) $\ell = 0.5$, (C) $\ell = 1.0$, (D) $\ell = 2.0$, (E) $\ell = 5.0$ para o kernel Rational Quadratic (RQ) com α fixo ($\alpha = 1.0$).

Analisando a Figura 15, observa-se um padrão similar ao encontrado no kernel RBF ao variar o lengthscale. Para $\ell = 0.1$, o kernel apresenta $R^2 = 0.199$, demonstrando severo overfitting. O lengthscale extremamente pequeno faz com que cada ponto de

treino influencie apenas uma região muito restrita, resultando em predições com oscilações extremamente abruptas que não correspondem ao comportamento suave da função $f(x) = x \cdot sen(x)$.

Com $\ell = 0.5$, o modelo alcança $R^2 = 0.539$, demonstrando melhoria substancial. O kernel começa a estabelecer correlações mais apropriadas entre pontos próximos, reduzindo as oscilações extremas, embora ainda apresente alguns sinais de overfitting com variações desnecessárias entre os pontos de treino.

Para $\ell = 1.0$, o kernel apresenta $R^2 = 0.548$, mantendo-se como o melhor desempenho dentre os testes evidenciados pela Figura 15. Este valor de lengthscale permite ao kernel capturar adequadamente as características da função senoidal, estabelecendo um equilíbrio satisfatório entre flexibilidade local e estabilidade global. Nota-se que o kernel RBF, com este mesmo valor de ℓ, obteve um ajuste melhor. Isso se deve ao fato de que o parâmetro α está fixado em 1.0, deixando o modelo menos suave.

Quando $\ell = 2.0$, o modelo apresenta uma queda abrupta de desempenho com $R^2 = 0.08$, demonstrando uma péssima capacidade de generalização. O kernel apresenta um grande underfitting, proporcionando predições muito mais suaves do que a função de estudo necessita.

Com $\ell = 5.0$, o modelo apresenta $R^2 = 0.0$, seguindo a tendência que o aumento do lengthscale gera. Com este valor, o underfitting se torna mais evidente, pois o kernel não consegue capturar variações importantes da função.

Com o final da análise do kernel Rational Quadratic, conclui-se que este kernel oferece uma alternativa ao RBF através do controle explícito da mistura de escalas via parâmetro α. Para a função estudada, valores elevados de α demonstraram melhor desempenho. O comportamento do lengthscale segue padrões esperados, onde valores muito pequenos (0.1) causam overfitting e valores muito grandes (5.0) geram underfitting severo. A principal vantagem do kernel RQ reside na sua capacidade de interpolar entre comportamentos de diferentes escalas através do parâmetro α, oferecendo maior flexibilidade que o RBF tradicional para capturar tanto correlações locais quanto de longo alcance de forma equilibrada. Para a função $f(x) = x \cdot sen(x)$, esse comportamento não trouxe melhoria nos ajustes, pois o melhor valor de R^2 foi obtido quando $\alpha = 10$, mostrando que quanto mais o kernel RQ se aproximava do RBF, melhor era o ajuste do modelo.

5.2.3 Matérn

Para compreender o impacto do parâmetro lengthscale no kernel Matérn 3/2, foram testados diferentes valores de ℓ, evidenciados no título da Figura 16, mantendo o parâmetro $\nu = 1.5$.

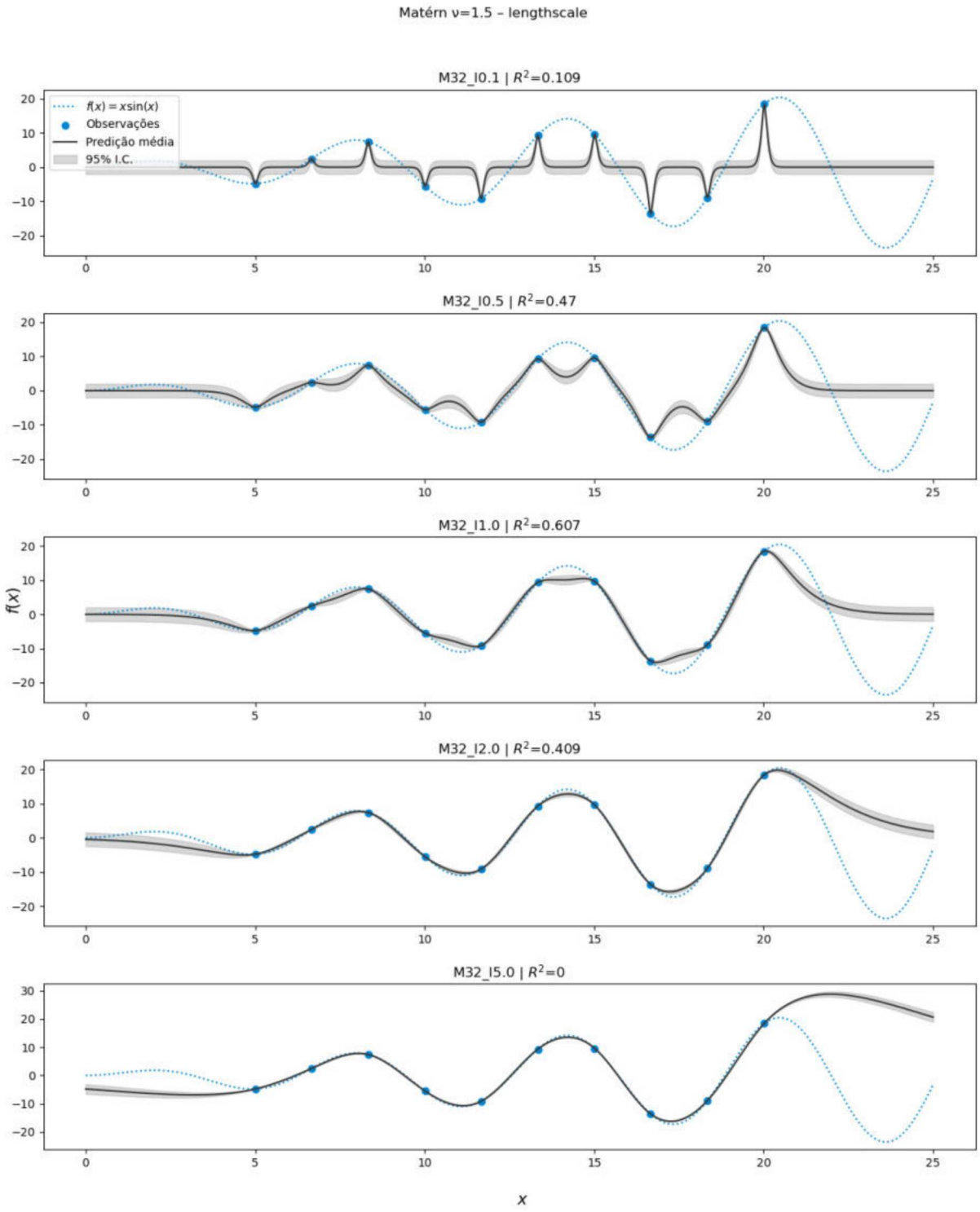

Figura 16 – Comparação entre cinco valores do parâmetro lengthscale (A) $\ell = 0.1$, (B) $\ell = 0.5$, (C) $\ell = 1.0$, (D) $\ell = 2.0$, (E) $\ell = 5.0$, para o kernel Matérn 3/2 ($\nu = 1.5$).

Observando a Figura 16, nota-se claramente como o parâmetro lengthscale afeta drasticamente o comportamento do kernel Matérn 3/2. Quando $\ell = 0.1$, o kernel apresenta $R^2 = 0.109$, demonstrando severo overfitting, onde a função predita apresenta oscilações

extremamente abruptas entre os pontos de treino, com picos e vales que não correspondem ao comportamento real da função $f(x) = x \cdot sen(x)$. O lengthscale muito pequeno faz com que cada ponto de treino tenha influência em uma região extremamente restrita ao seu redor, resultando em predições que não capturam a suavidade da função estudada.

Para $\ell = 0.5$, observa-se uma melhoria substancial no ajuste, com $R^2 = 0.470$. Embora o modelo comece a capturar melhor a tendência geral da função, ainda apresenta oscilações desnecessárias e instabilidades entre os pontos de treino, caracterizando um overfitting moderado. A correlação entre pontos adjacentes começa a se apresentar, mas ainda é insuficiente para produzir predições suaves e confiáveis.

O desempenho com $\ell = 1.0$ resulta em $R^2 = 0.607$, demonstrando um equilíbrio melhor. O kernel consegue capturar adequadamente algumas características locais e um pouco do comportamento global da função senoidal, proporcionando predições mais estáveis e realistas. As oscilações extremas são eliminadas, e a função predita acompanha de forma mais natural o padrão esperado. Por esses motivos, este foi o melhor kernel dentre os observados na Figura 16.

Quando $\ell = 2.0$, o modelo apresenta $R^2 = 0.409$, representando uma piora significativa no ajuste. Contrariamente ao esperado, o aumento do lengthscale não melhora o desempenho, mas sim começa a suprimir variações importantes da função, indicando o início de underfitting.

Para $\ell = 5.0$, obtém-se $R^2 = 0.0$, indicando falha total do modelo, devido a um underfitting severo. O lengthscale excessivamente grande faz com que pontos muito distantes mantenham uma correlação elevada, resultando em predições extremamente suaves que não conseguem capturar nenhuma das oscilações senoidais do início e do final do intervalo de estudo da função original.

A análise do impacto do lengthscale no Matérn 5/2 foi realizada fixando $\nu = 2.5$ e variando ℓ, conforme apresentado na Figura 17.

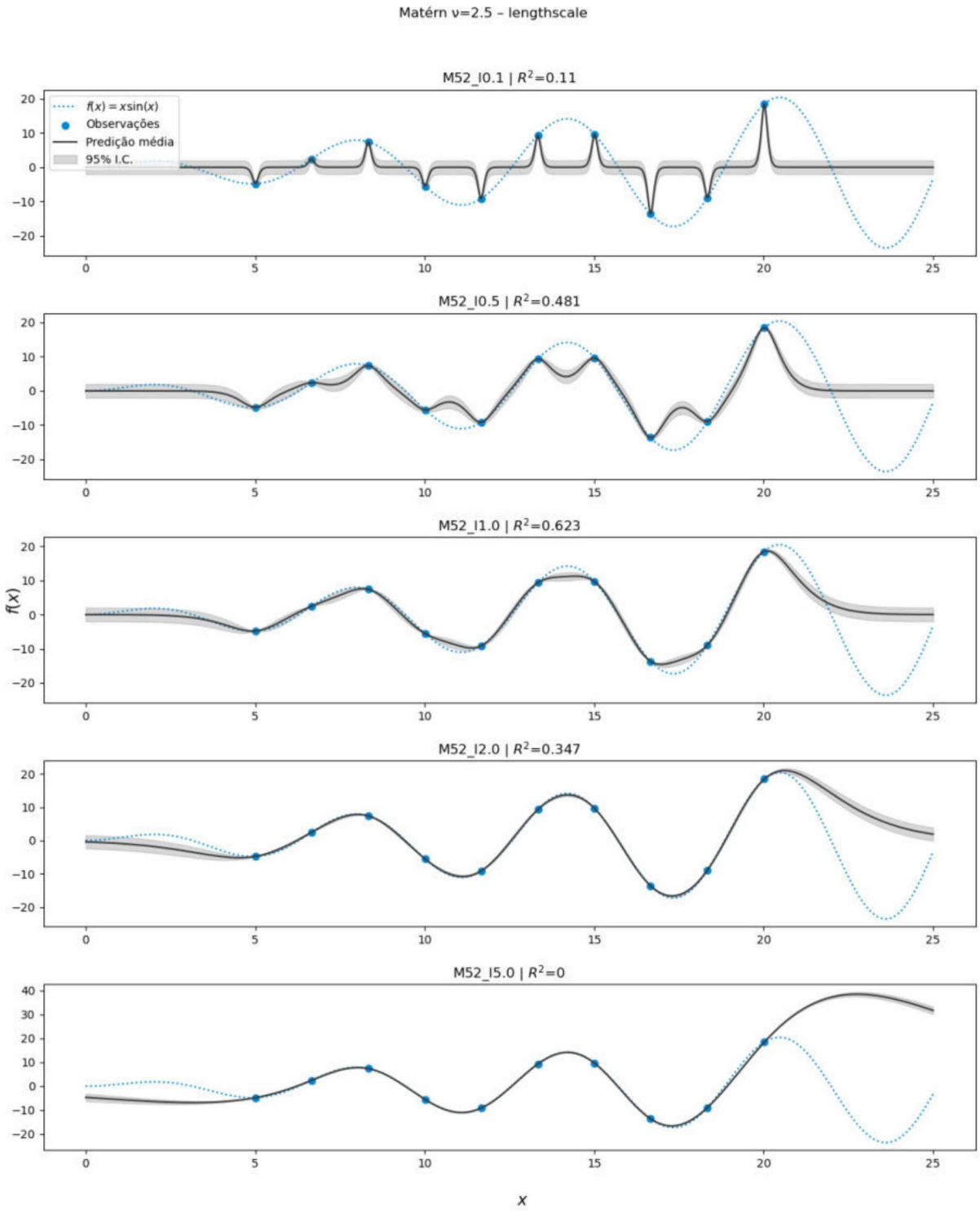

Figura 17 – Comparação entre cinco valores do parâmetro lengthscale (A) $\ell = 0.1$, (B) $\ell = 0.5$, (C) $\ell = 1.0$, (D) $\ell = 2.0$, (E) $\ell = 5.0$, para o kernel Matérn 5/2 ($\nu = 2.5$).

Analisando a Figura 17, observa-se um padrão similar ao Matérn 3/2, mas com maior robustez devido à suavidade superior. Para $\ell = 0.1$, o kernel ainda apresenta $R^2 = 0.110$, confirmando que lengthscales extremamente pequenos são problemáticos

independentemente do valor de ν. A maior diferenciabilidade e suavidade do Matérn 5/2 não conseguem compensar o efeito de um lengthscale inadequado.

Com $\ell = 0.5$, o modelo alcança $R^2 = 0.481$, demonstrando ligeira melhoria em relação ao Matérn 3/2 no mesmo lengthscale. A maior suavidade intrínseca do Matérn 5/2 proporciona maior estabilidade nas predições. Mas, mesmo com a melhora, ainda é possível notar que o overfitting está presente.

Para $\ell = 1.0$, o kernel apresenta melhor desempenho com $R^2 = 0.623$, mantendo-se estável e capturando adequadamente as características da função senoidal. Este resultado demonstra a maior robustez do Matérn 5/2 em relação ao 3/2.

Quando $\ell = 2.0$, o modelo apresenta $R^2 = 0.347$, mostrando uma piora em relação ao Matérn 3/2. Devido ao kernel ser mais suave, o comportamento do lengthscale é intensificado, comprometendo significativamente a capacidade de capturar variações locais, demonstrando um underfitting.

Com $\ell = 5.0$, o modelo apresenta novamente $R^2 = 0.0$, confirmando, novamente, que lengthscales muito grandes levam à falha completa do modelo por underfitting extremo, independentemente do valor de ν.

Para compreender o impacto do parâmetro ν na suavidade do kernel Matérn, foi realizada uma análise fixando o lengthscale em $\ell = 1.0$ e variando ν entre os valores mais comuns: 0.5, 1.5 e 2.5, conforme apresentado na Figura 18.

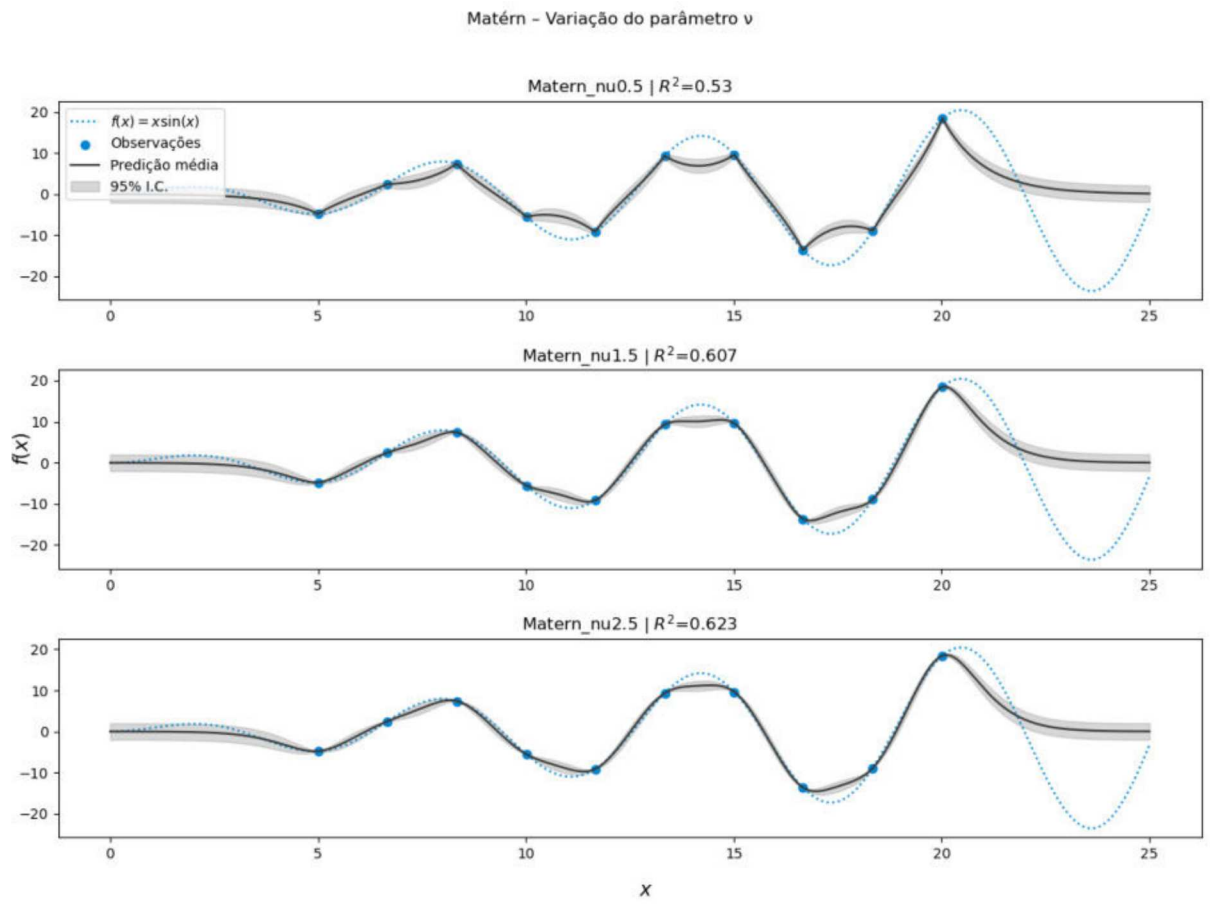

Figura 18 – Comparação entre três valores do parâmetro ν, (A) $\nu = 0.5$, (B) $\nu = 1.5$, (C) $\nu = 2.5$, para o kernel Matérn com lengthscale fixo ($\ell = 1.0$).

Observando a Figura 18, nota-se claramente o impacto do parâmetro ν na capacidade de ajuste do kernel. Para $\nu = 0.5$ (kernel exponencial), o modelo apresenta $R^2 = 0.530$, o pior desempenho dentre os três valores testados. Este kernel gera funções não diferenciáveis que, embora capazes de capturar variações abruptas, são inadequadas para modelar funções com oscilações suaves, como $f(x) = x \cdot sen(x)$.

Com $\nu = 1.5$ (Matérn 3/2), o kernel alcança $R^2 = 0.607$, demonstrando melhoria clara. As funções geradas são diferenciáveis uma vez, proporcionando suavidade adicional que permite capturar melhor o comportamento oscilatório da função senoidal.

O melhor desempenho é observado com $\nu = 2.5$ (Matérn 5/2), resultando em $R^2 = 0.623$. Este valor de ν gera funções duas vezes diferenciáveis que capturam de forma mais eficaz as oscilações suaves da função estudada, demonstrando que maior suavidade beneficia significativamente a modelagem dessa função específica.

É importante destacar que, conforme ν aumenta, o kernel Matérn se aproxima progressivamente do comportamento do kernel RBF. Esta convergência permite que o kernel Matérn ofereça um espectro contínuo de comportamentos, desde funções rugosas e

não diferenciáveis ($\nu = 0.5$) até funções muito suaves que se aproximam do RBF ($\nu \to \infty$).

Com o final da análise do kernel Matérn, conclui-se que este kernel oferece controle mais refinado que o RBF através da parametrização explícita da suavidade com a variação do parâmetro ν. Para a função estudada, valores maiores de ν (2.5) demonstraram melhor desempenho, confirmando que a suavidade adicional é benéfica para capturar o comportamento oscilatório de $x \cdot sen(x)$. Contudo, a análise revela um comportamento crítico do parâmetro lengthscale, pois evidencia que há uma faixa muito estreita de valores adequados (próximos a 1.0), onde valores extremos (0.1 ou 5.0) levam à falha completa do modelo com $R^2 = 0.0$. Este comportamento demonstra que o kernel Matérn, embora mais flexível que o RBF, requer cuidadosa seleção de hiperparâmetros para evitar overfitting, com lengthscales muito pequenos, e underfitting, ao se utilizar lengthscales muito grandes.

5.2.4 Periódico

Para compreender o impacto do parâmetro lengthscale no kernel periódico, foram testados diferentes valores fixos, mantendo o período $p = 6.28$, como pode ser observado na Figura 19. Os resultados demonstram como o lengthscale afeta drasticamente o comportamento do kernel.

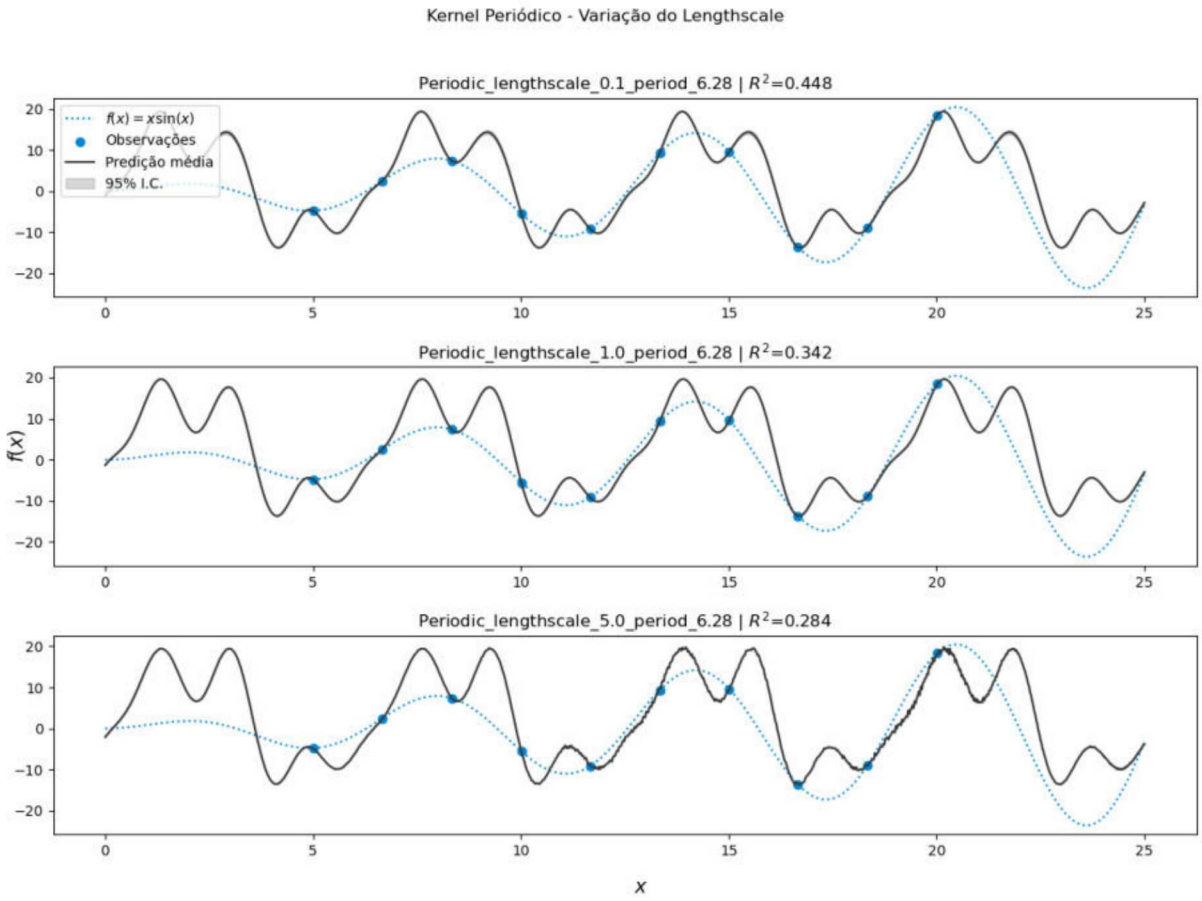

Figura 19 – Comparação entre três valores do parâmetro lengthscale (A) $\ell = 0.1$, (B) $\ell = 1.0$, (C) $\ell = 5.0$, para o kernel Periódico com período fixo ($p = 6.28$).

Quando $\ell = 0.1$, o kernel apresenta o melhor desempenho com $R^2 = 0.448$. Apesar do R^2 razoável obtido, o valor pequeno de lengthscale permite que o kernel capture somente um pouco as variações rápidas e as oscilações locais presentes na função $f(x) = x \cdot sen(x)$.

Para $\ell = 1.0$, observa-se uma degradação no desempenho, com $R^2 = 0.342$. O aumento do lengthscale faz com que o kernel suavize excessivamente as oscilações, perdendo parte da capacidade de capturar as variações rápidas que caracterizam a função estudada. Este comportamento indica que o kernel começa a impor uma suavidade inadequada para a natureza da função.

Quando $\ell = 5.0$, o modelo apresenta $R^2 = 0.284$, representando uma piora significativa no ajuste. O lengthscale excessivamente grande faz com que o kernel perca completamente a sensibilidade às variações locais, resultando em predições excessivamente suaves que não conseguem acompanhar as oscilações características da função $f(x) = x \cdot sen(x)$.

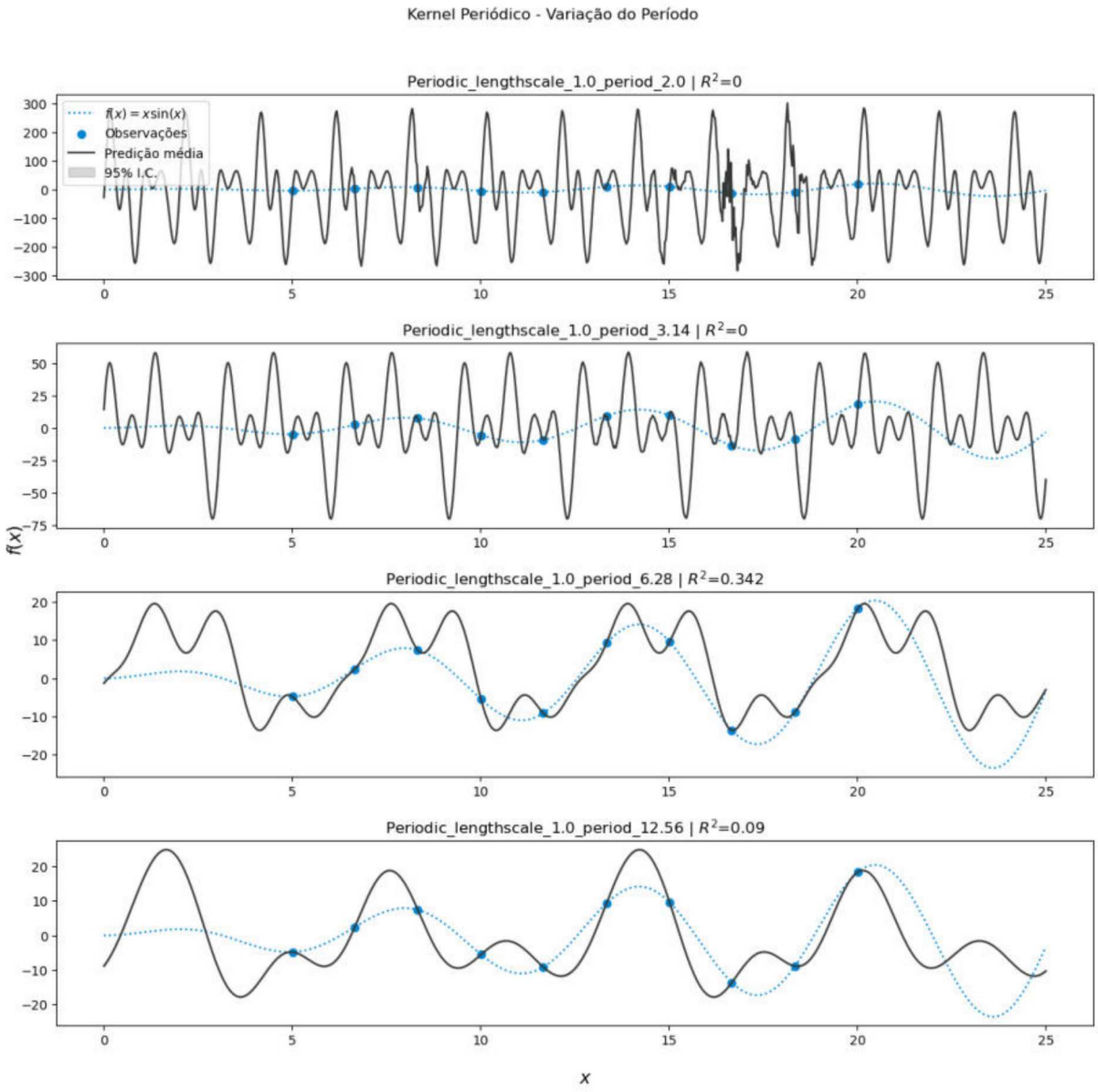

Figura 20 – Comparação entre quatro valores do parâmetro período (A) $p = 2.0$, (B) $p = 3.14$, (C) $p = 6.28$, (D) $p = 12.56$, para o kernel Periódico com lengthscale fixo ($\ell = 1.0$).

A análise do impacto do parâmetro período foi realizada fixando o lengthscale em $\ell = 1.0$ e variando o período, como é mostrado na Figura 20. Os resultados revelam a criticidade extrema deste parâmetro para o sucesso do modelo.

Para $p = 2.0$, o kernel apresenta falha completa com $R^2 = 0.0$. Este período é inadequado pois não corresponde à periodicidade natural da componente senoidal da função. O valor $p = 2.0$ é significativamente menor que $2\pi \approx 6.28$, fazendo com que o kernel tente impor uma periodicidade incorreta que não existe na função real.

Similarmente, para $p = 3.14 \approx \pi$, o modelo também apresenta $R^2 = 0.0$. Embora

este valor esteja relacionado à função senoidal (metade do período real), ainda representa uma periodicidade incorreta. O kernel tenta forçar repetições em intervalos de π quando a função $f(x) = x \cdot sen(x)$ possui zeros em múltiplos de π, mas não apresenta repetição exata devido ao fator multiplicativo x.

O melhor desempenho é observado com $p = 6.28 \approx 2\pi$, resultando em $R^2 = 0.342$. Este valor corresponde exatamente ao período da função $sen(x)$, permitindo que o kernel capture adequadamente a periodicidade do componente senoidal, porém a função $f(x) = x \cdot sen(x)$ não é estritamente periódica, devido ao fator x, com isso, o ajuste do kernel fica prejudicado.

Quando $p = 12.56 \approx 4\pi$, o modelo apresenta $R^2 = 0.09$, indicando underfitting severo. O período excessivamente grande faz com que o kernel perca a capacidade de capturar as oscilações na frequência correta, resultando em predições que não conseguem acompanhar as variações da função original.

O kernel periódico apresenta propriedades únicas que o distinguem de outros kernels. Sua formulação matemática garante que pontos separados por múltiplos exatos do período tenham correlação máxima, enquanto pontos separados por meio período apresentam correlação mínima. Esta característica é fundamental para modelar funções verdadeiramente periódicas. Contudo, a aplicação do kernel periódico à função $f(x) = x \cdot sen(x)$ é limitada, pois a função não é estritamente periódica devido ao fator multiplicativo x que aumenta linearmente a amplitude.

5.3 Escolha de kernels para análises

A seleção dos kernels para análise de acurácia e eficiência dos dados simulados fundamentou-se na diversidade de padrões presentes nos fenômenos cardíacos, na estrutura estatística da análise exploratória dos dados e na análise dos efeitos dos parâmetros dos kernels.

Em primeiro lugar, optou-se por kernels de suavidade local, como RBF e Matérn com $\nu = 3/2$ e $\nu = 5/2$, devido à capacidade que eles possuem de modelar relações onde a proximidade entre pontos implica respostas semelhantes. O RBF serve como referência clássica para funções de alta suavidade, enquanto os Matérn oferecem diferentes graus de diferenciabilidade, sendo utilizados para capturar transições suaves e mudanças abruptas.

Para modelar fenômenos em diferentes escalas simultaneamente, característica essencial para dados um pouco mais complexos, foi escolhido o kernel RQ e a sua combinação aditiva, RQ+RBF, que potencializa a capacidade de modelar padrões locais e globais.

As escolhas para modelos que exibem tendências globais ou crescimentos sistemáticos foram os kernels Poly, RBF*Linear, Matern32+Linear e Matern52*Linear. A composição do RBF com um kernel linear ou dos kernels Matérn com um componente

linear, permite que sejam capturados, simultaneamente, efeitos lineares de longo alcance e variações locais mais sutis.

Por fim, as composições do kernel Poly, sendo Poly*Matern32, Poly+Matern32, Poly*RBF e Poly+RBF, foram selecionadas por aliar a capacidade de modelar comportamentos de crescimento global com as transições locais. A distinção entre soma e multiplicação proporciona flexibilidade para modelar situações em que as tendências e variações das variáveis estão interligadas.

Esta diversidade de kernels garante cobertura desde comportamentos lineares até variações mais drásticas em múltiplas escalas, permitindo encontrar kernels que se ajustem à complexidade dos dados simulados. Após realizar a análise de acurácia e eficiência dos treze kernels apresentados, os três de melhor desempenho serão selecionados para a comparação entre emuladores, com base no score de desempenho (detalhes na seção 4.7). Esses três kernels representarão níveis de baixa, média e alta complexidade, de forma a manter a mesma subdivisão adotada nos demais emuladores.

5.4 Análise de acurácia e eficiência dos kernels utilizando GPyTorch

Os resultados a seguir são referentes às análises de ambos os modelos, Modelo A e Modelo B, em diversos testes de acurácia e eficiência utilizando o emulador por Processos Gaussianos.

São apresentados gráficos contendo a alocação de memória, tempo de treino, pontos inferidos por segundo e erro quadrático médio (MSE) conforme o tamanho amostral aumenta. Para as três primeiras métricas, são exibidos os valores médios das quatro QoIs, uma vez que os gráficos isolados de cada quantidade de interesse (QoI) não apresentaram alterações significativas. Para o MSE, devido às pequenas diferenças observadas entre as QoIs, são apresentados tanto os gráficos individuais quanto o valor médio. O mesmo critério foi aplicado ao score de Pareto, que sofreu pequenas alterações em função das variações entre as QoIs.

5.4.1 Análise da alocação de memória na GPU

Observando as Figuras 21 e 22, nota-se que, conforme o tamanho da amostra dos dados de treino aumenta, cresce o pico de memória alocado pelos batches, havendo um pequeno degrau próximo ao valor de 800 amostras, onde há um decaimento pontual, mas após este a tendência de aumento se mantém. É possível observar que há uma linha pontilhada verde com triângulos, representando o kernel Matérn32+Linear, que se destaca por um uso mais elevado de memória e logo abaixo desta linha uma série de outras em sobreposição, representando os demais kernels, que obtiveram métricas similares, mas ressalta-se que todos demonstraram a mesma tendência de crescimento e o mesmo degrau.

É interessante o fato do degrau ter aparecido para todos os kernels no mesmo momento, ou seja, em um número de amostras específico. Isso pode ser explicado pelo modo como o alocador de cache do PyTorch funciona, pois, ao atingir um limiar, a estratégia de alocação que inicialmente é incremental passa a ser de reutilização otimizada de blocos de memória já existentes, combinado com otimizações específicas do GPyTorch (PYTORCH TEAM, 2024).

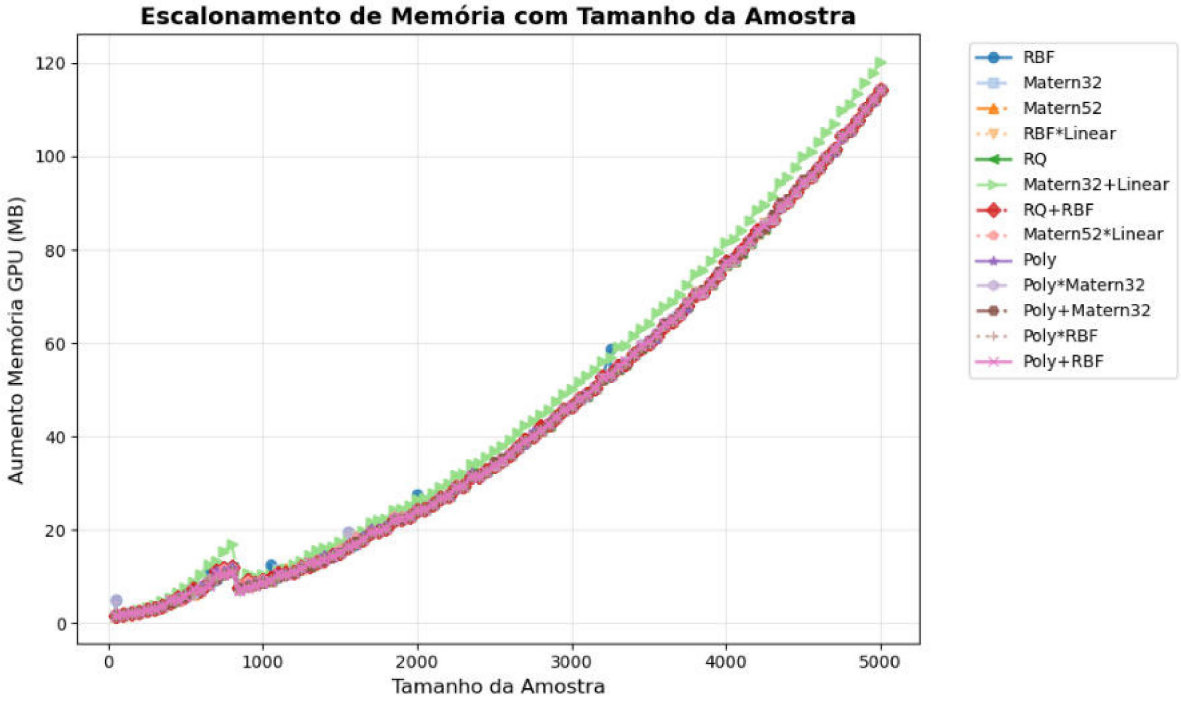

Figura 21 – Gráfico de dispersão mostrando o escalonamento da alocação de memória na GPU (em MB) para cada um dos batches feitos na inferência, para diferentes tamanhos de amostra de treino do Modelo A.

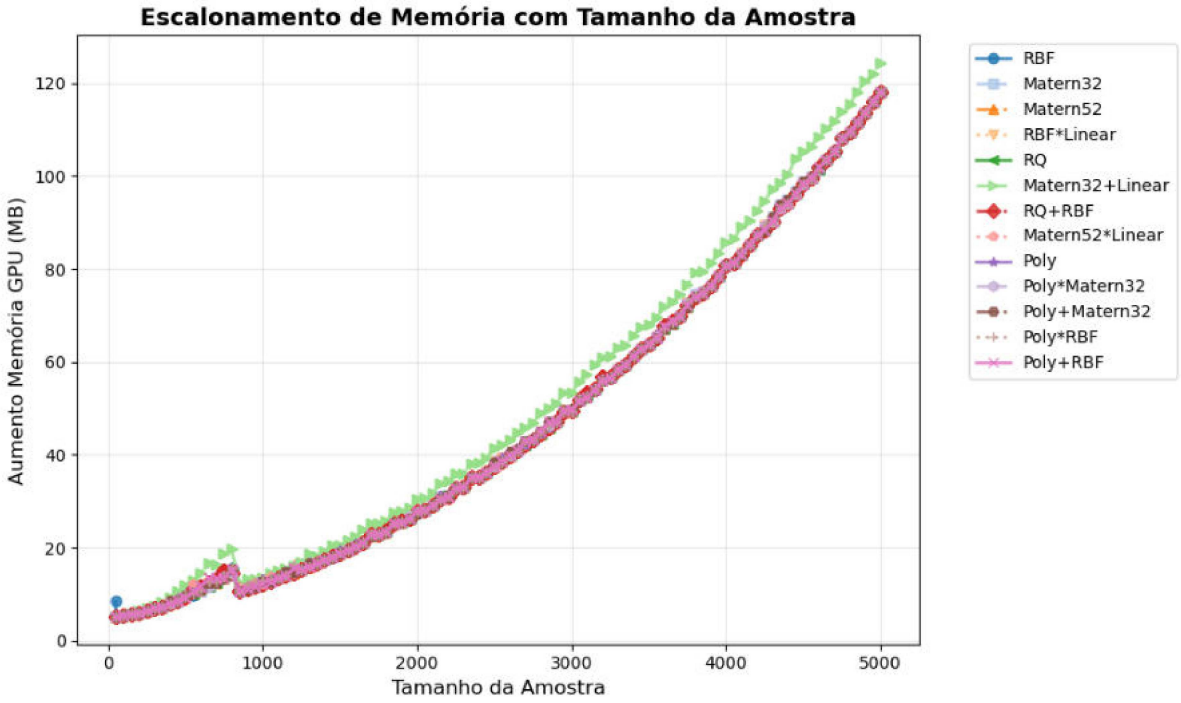

Figura 22 – Gráfico de dispersão mostrando o escalonamento da alocação de memória na GPU (em MB) para cada um dos batches feitos na inferência, para diferentes tamanhos de amostra de treino do Modelo B.

5.4.2 Análise do tempo de treinamento

Analisando as Figuras 23 e 24, que apresentam a relação do tempo de treino de um kernel com o tamanho dos dados de treino, destacamos negativamente novamente o kernel Matérn32+Linear que obteve os maiores tempos de treino. Os demais kernels obtiveram resultados semelhantes.

Aprofundando a interpretação dos gráficos, observa-se uma correlação positiva entre o tempo de treino e o tamanho da amostra, em todos os kernels, e notamos novamente alguns picos que se expressam de forma consistente em tamanhos específicos de amostra. Este fenômeno pode ser explicado, mais uma vez, pelo mecanismo do alocador de cache do PyTorch, onde transições nas estratégias de gestão de memória GPU provocam custos computacionais adicionais que geram tempos de treino mais proeminentes (PYTORCH TEAM, 2024). Novamente, é possível notar que o padrão sincronizado entre todos os kernels reforça a hipótese de que as otimizações internas são ativadas em limiares específicos de tamanho de amostra.

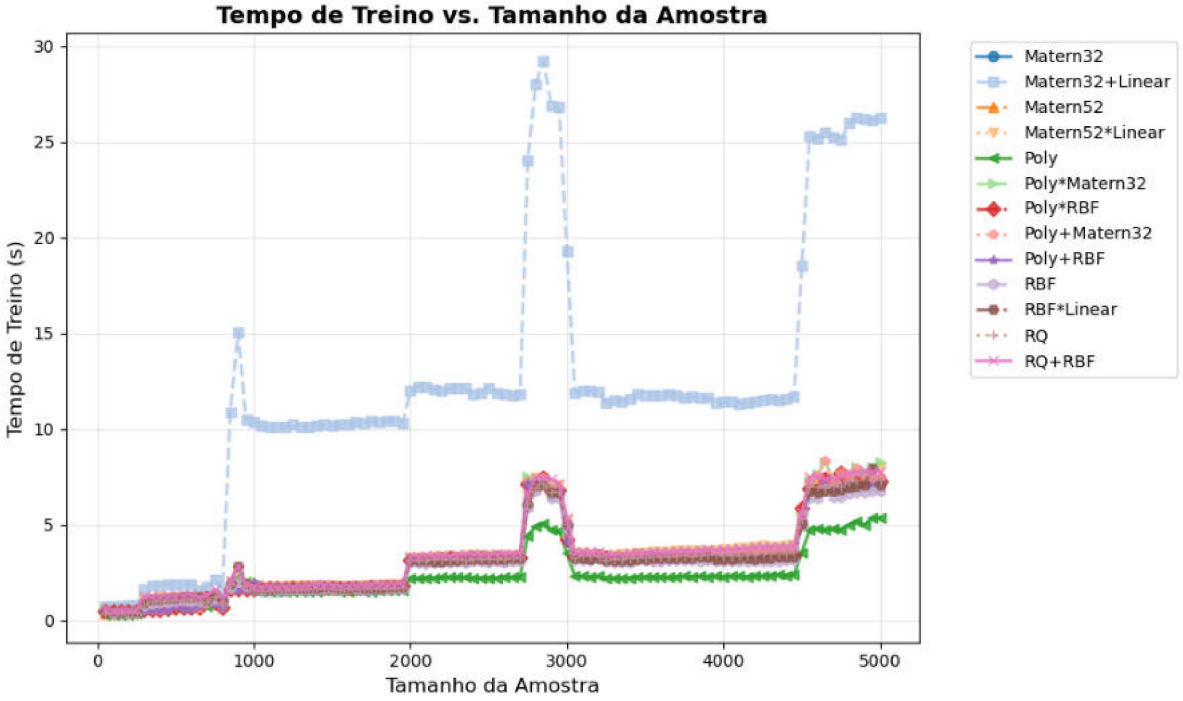

Figura 23 – Gráfico de dispersão mostrando o escalonamento do tempo de treino dos kernels para diferentes tamanhos de amostra de treino do Modelo A.

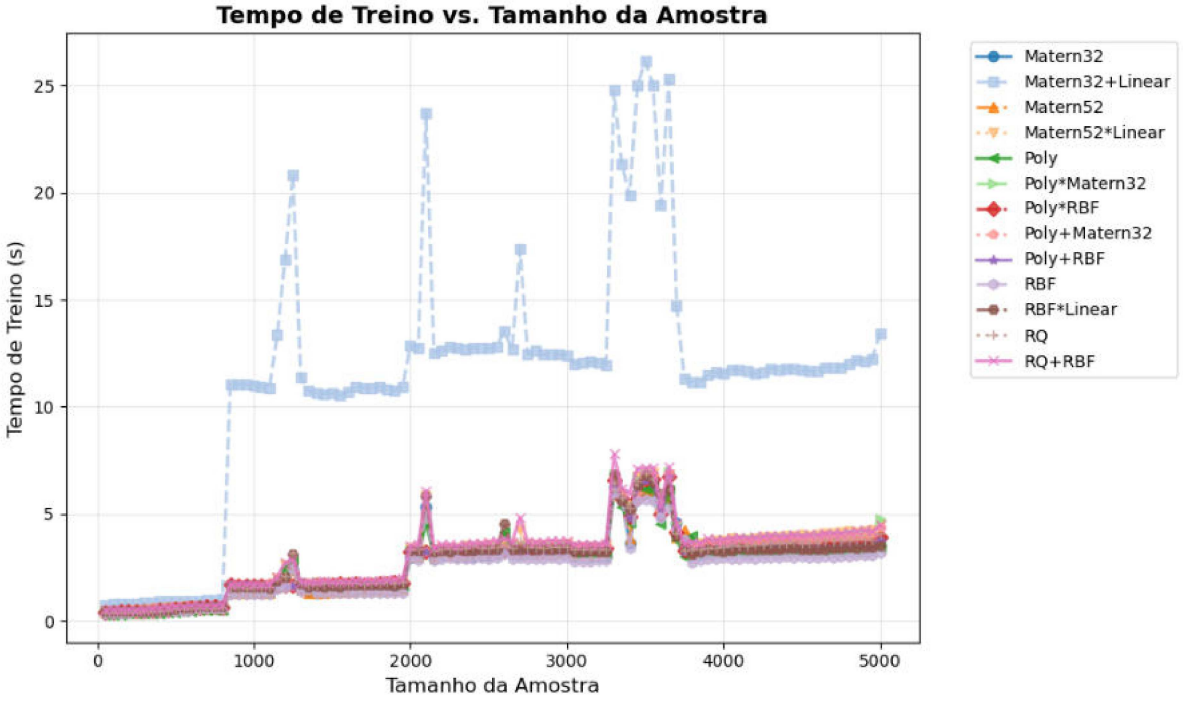

Figura 24 – Gráfico de dispersão mostrando o escalonamento do tempo de treino dos kernels para diferentes tamanhos de amostra de treino do Modelo B.

5.4.3 Análise da eficiência de inferência

Investigando as Figuras 25 e 26, que avaliam a velocidade de inferência dos kernels conforme o tamanho da amostra aumenta, destacamos novamente o kernel Matérn32+Linear com o pior desempenho, mas os demais kernels não estão agrupados, como ocorreu nos gráficos analisados até o momento. É possível observar dois grupos principais. O grupo que se destaca é composto pelos kernels RBF, RQ, Polynomial, Matérn32 e Matérn52, nota-se que esses são os únicos kernels que não são compostos, ou seja, não é uma combinação linear ou soma de dois kernels.

O comportamento observado com quedas abruptas nos mesmos pontos onde ocorrem os picos de tempo de treino, Figuras 23 e 24, é uma consequência direta da relação inversamente proporcional entre estas métricas. Como já abordado, quando o sistema realiza as transições do alocador de memória e reorganiza as estruturas do kernel, o throughput sofre uma redução, assim como acontece com o tempo de treino (PYTORCH TEAM, 2024).

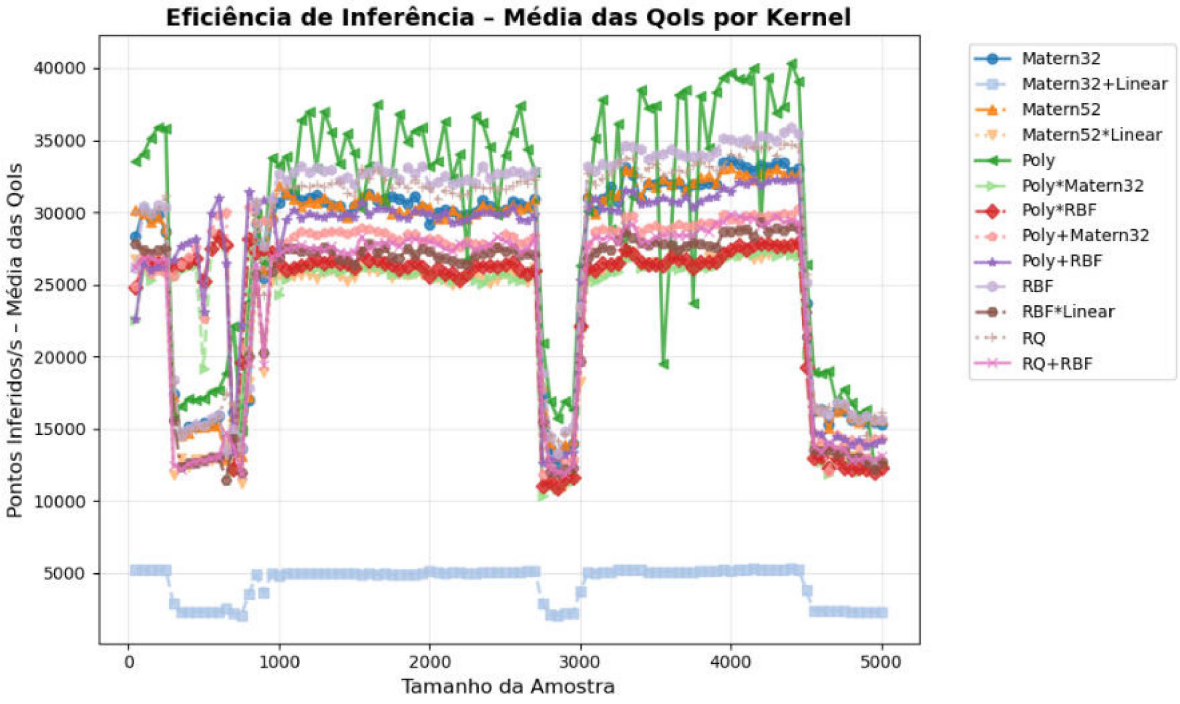

Figura 25 – Gráfico de dispersão mostrando a variação dos pontos inferidos por segundo em cada kernels em diferentes tamanhos de amostra de treino do Modelo A.

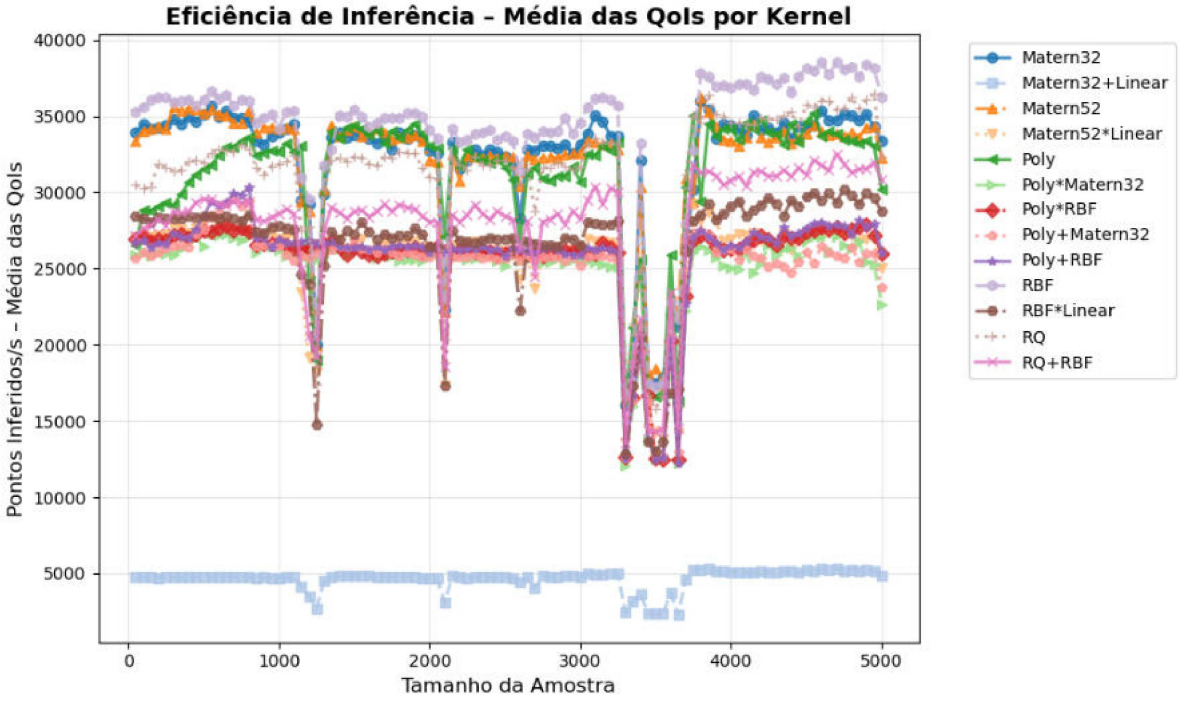

Figura 26 – Gráfico de dispersão mostrando a variação dos pontos inferidos por segundo em cada kernels em diferentes tamanhos de amostra de treino do Modelo B.

5.4.4 Análise do erro quadrático médio (MSE)

Inicialmente, a análise da métrica de erro quadrático médio (MSE) foi conduzida de forma global, utilizando a média dos valores das quatro quantidades de interesse (QoIs) para comparar a evolução do MSE conforme o tamanho da amostra. Posteriormente, examinou-se cada QoI isoladamente para verificar se a ordem de desempenho dos kernels se mantinha em todas as variáveis de saída.

Iniciando a análise global, analisando as Figuras 27 e 29, que exibem como o erro quadrático médio se comporta com o aumento do tamanho da amostra de treino, nota-se que o comportamento dos kernels se difere um pouco, onde alguns possuem um decaimento do MSE mais elevado do que outros.

Observando os dados do Modelo A, Figura 27 e a Tabela 2, verifica-se que os kernels Poly+Matern32 e Poly+RBF se destacam dos demais, atingindo o menor valor de MSE, distando-se dos demais cerca de um grau de magnitude. Outra característica notável desses kernels foi a rápida convergência, pois em torno de 1000 amostras de treino seus erros estabilizaram, atingindo o menor valor de MSE em torno do tamanho amostral 2000. Outro kernel que se destacou foi o Poly, atingindo um platô logo nos primeiros tamanhos amostrais, porém não sendo tão acurado, obtendo apenas o décimo melhor MSE. Os demais kernels, não se destacam em termos de acurácia ou velocidade de convergência, seguindo uma tendência geral de decaimento do erro com o aumento do tamanho da

amostra, mas sem apresentar platôs claros de convergência, mesmo com 5000 amostras. Isso indica que, para esses kernels, valores mais baixos de MSE possivelmente só seriam atingidos com conjuntos de treino ainda maiores.

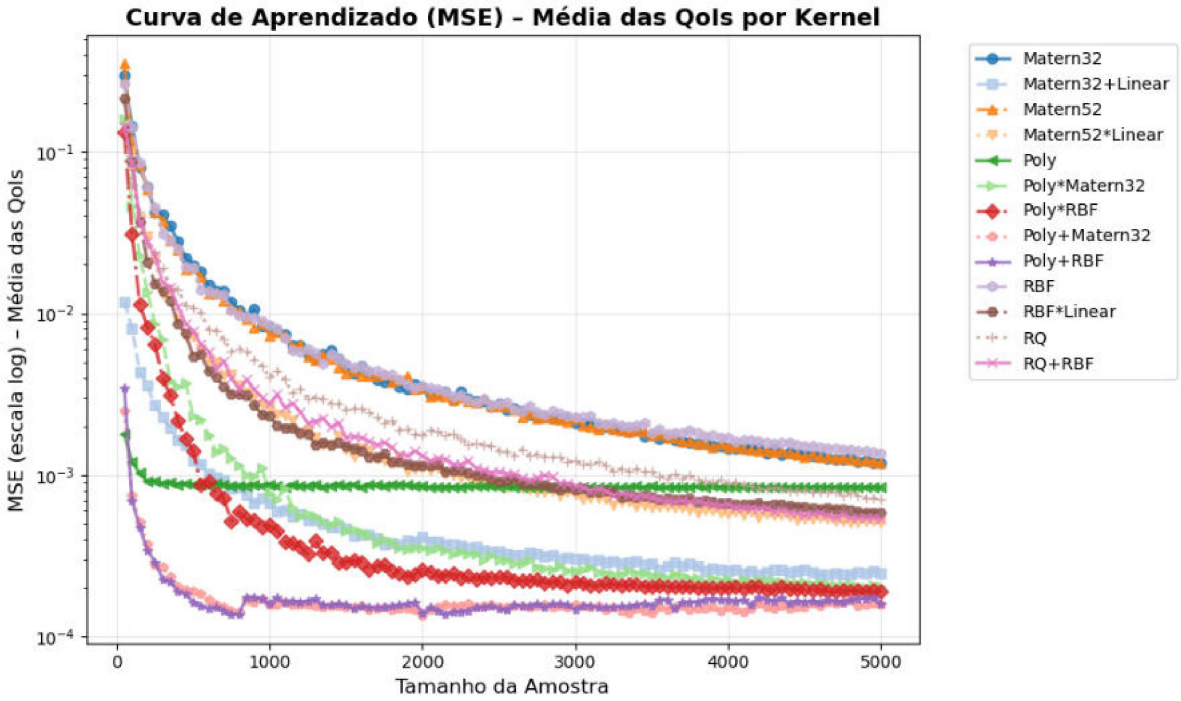

Figura 27 – Gráfico de dispersão mostrando a variação do erro quadrático médio (MSE) para cada kernels em diferentes tamanhos de amostra de treino do Modelo A.

Tabela 2 – Tamanho da amostra de treino para se obter o menor erro quadrático médio (MSE) para cada kernel do Modelo A.

Kernel	**Tamanho da Amostra**	**MSE**
Poly+Matern32	2000	$1{,}347 \times 10^{-4}$
Poly+RBF	750	$1{,}372 \times 10^{-4}$
Poly*RBF	4900	$1{,}884 \times 10^{-4}$
Poly*Matern32	4900	$2{,}008 \times 10^{-4}$
Matern32+Linear	4650	$2{,}411 \times 10^{-4}$
RBF*Linear	4950	$5{,}865 \times 10^{-4}$
Matern52*Linear	5000	$5{,}077 \times 10^{-4}$
RQ+RBF	5000	$5{,}357 \times 10^{-4}$
RQ	5000	$7{,}106 \times 10^{-4}$
Poly	3550	$8{,}343 \times 10^{-4}$
RBF	5000	$1{,}369 \times 10^{-3}$
Matern32	5000	$1{,}178 \times 10^{-3}$
Matern52	5000	$1{,}190 \times 10^{-3}$

Fonte: Elaboração própria.

Na avaliação individual de cada QoI do Modelo A, Figura 28, ao analisar o MSE de cada QoI separadamente, constatou-se que os kernels que lideraram a média global, Poly+Matern32 e Poly+RBF, também ocuparam as duas primeiras posições nas quatro QoIs. Os demais kernels exibiram curvas e colocações consistentes entre si, com exceção do kernel Poly, que, apesar de ocupar o décimo lugar na média, obteve desempenho variável: melhor em Vreps, mediano em APD50 e APD90 e pior em dVmax. Esse comportamento evidencia que, para os demais kernels, a classificação por MSE é robusta e independentemente da QoI considerada, e que o ranking médio reflete corretamente o ranking individual de cada QoI.

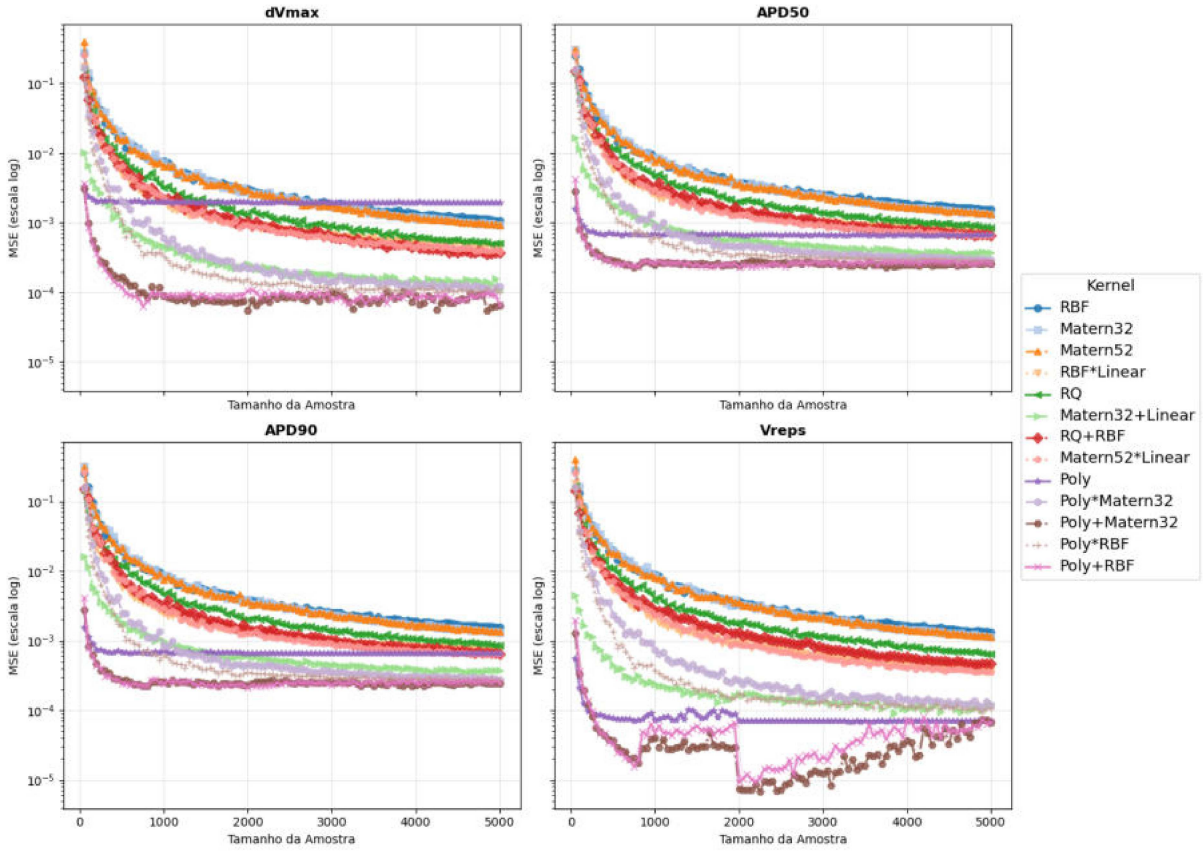

Figura 28 – Gráficos de dispersão apresentando a variação do erro quadrático médio (MSE) para cada quantidade de interesse, avaliando o desempenho de diferentes kernels em diferentes tamanhos de amostra de treino no Modelo A.

Examinando os dados da análise global do Modelo B, Figura 29 e a Tabela 3, verifica-se que os kernels Poly+Matern32, Poly+RBF e Poly se destacam dos demais por terem um forte decaimento inicial do MSE conforme o tamanho amostral aumenta, por isso, esses atingiram o menor valor de MSE, distando dos demais cerca de um grau de magnitude. Outro kernel que se destacou foi o Matern32+Linear, que, apesar de não ter um bom decaimento do MSE, possui um valor de MSE baixo com o menor tamanho amostral e depois mantém-se quase em um platô. Já os kernels RQ e RQ+RBF se aproximam do valor do Matern32+Linear, mas apresentam um decaimento mais acentuado. Os demais kernels apresentaram baixo decaimento e um valor inicial alto, não obtendo um resultado expressivo. Um comportamento interessante de se ressaltar é que, no Modelo B, os valores de MSE decrescem constantemente até o maior tamanho amostral, pois, em 12 dos 13 kernels, o menor valor foi obtido apenas nesse ponto. Esse cenário é contrastante com o observado no Modelo A, onde há registros de alguns kernels atingindo o seu menor MSE com tamanho amostral igual a 750 e 2000, por exemplo.

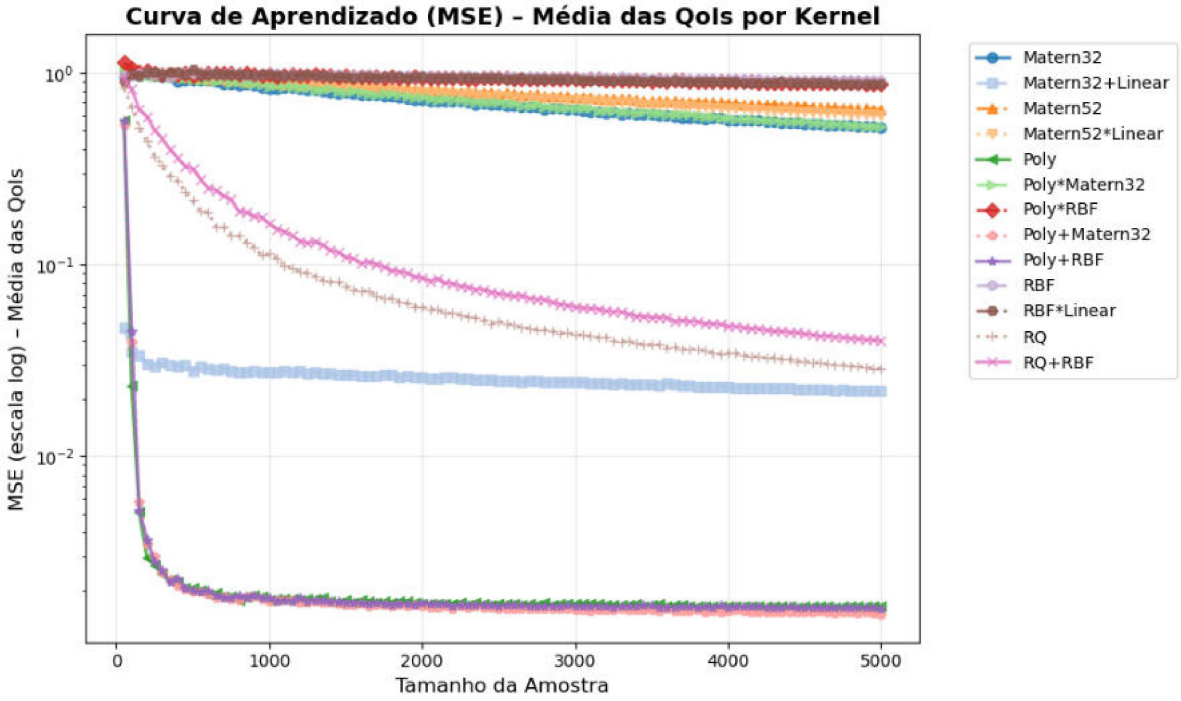

Figura 29 – Gráfico de dispersão mostrando a variação do erro quadrático médio (MSE) para cada kernels em diferentes tamanhos de amostra de treino do Modelo B.

Tabela 3 – Tamanho da amostra de treino para se obter o menor erro quadrático médio (MSE) para cada kernel do Modelo B.

Kernel	**Tamanho da Amostra**	**MSE**
Poly+Matern32	5000	$1{,}492 \times 10^{-3}$
Poly+RBF	5000	$1{,}607 \times 10^{-3}$
Poly	4450	$1{,}635 \times 10^{-3}$
Matern32+Linear	5000	$2{,}187 \times 10^{-2}$
RQ	5000	$2{,}860 \times 10^{-2}$
RQ+RBF	5000	$3{,}999 \times 10^{-2}$
Matern32	5000	$5{,}207 \times 10^{-1}$
Poly*Matern32	5000	$5{,}244 \times 10^{-1}$
Matern52*Linear	5000	$6{,}017 \times 10^{-1}$
Matern52	5000	$6{,}466 \times 10^{-1}$
RBF*Linear	5000	$8{,}749 \times 10^{-1}$
Poly*RBF	5000	$8{,}780 \times 10^{-1}$
RBF	5000	$9{,}078 \times 10^{-1}$

Fonte: Elaboração própria.

Na análise de cada QoI do Modelo B, Figura 30, verificou-se que todos os kernels seguem o mesmo ranking médio em cada variável de saída, exceto o Matern32+Linear, que, assim como o Poly no Modelo A, apresentou desempenho variável: melhor em Vreps, mediano em APD50 e APD90 e pior em dVmax. Esse padrão confirma, novamente, que, para os demais kernels, a ordenação por MSE é robusta e o ranking global resume adequadamente o desempenho individual em cada QoI.

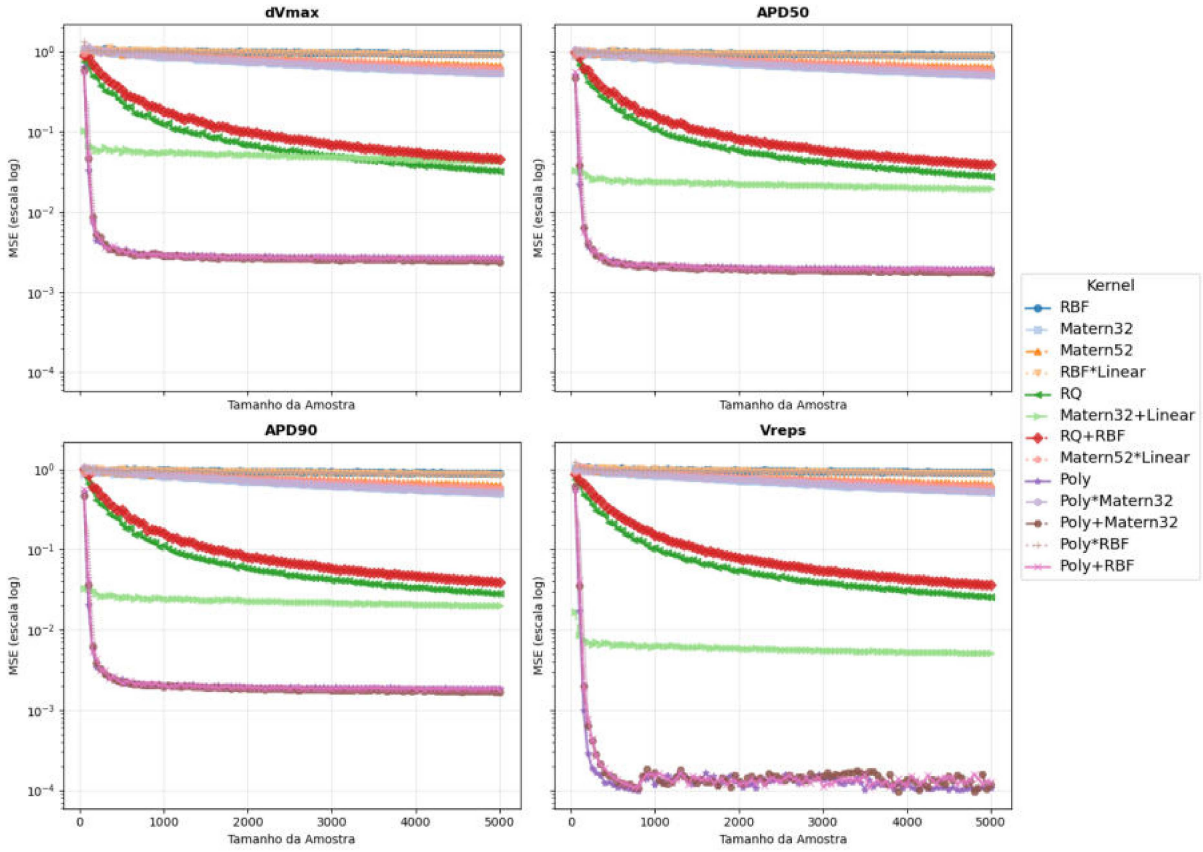

Figura 30 – Gráficos de dispersão apresentando a variação do erro quadrático médio (MSE) para cada quantidade de interesse, avaliando o desempenho de diferentes kernels em diferentes tamanhos de amostra de treino no Modelo B.

5.4.5 Análise do desempenho pelo score de Pareto

Inicialmente, os kernels foram avaliados por meio de um score de Pareto calculado a partir da média dos resultados das quatro quantidades de interesse (QoIs). Essa estratégia justifica-se pelo uso de um único kernel para todas as saídas, de modo que escolher o melhor desempenho médio assegura maior robustez na comparação dos processos gaussianos com outros emuladores. Em seguida, cada QoI foi examinada isoladamente para validar a consistência e a significância das diferenças observadas no ranking geral.

O desempenho de cada kernel foi quantificado segundo quatro métricas: uso de memória da GPU, tempo de treino, pontos inferidos por segundo e MSE, à medida que o tamanho da amostra de treino aumentava. Essas análises revelaram cenários em que determinados kernels se destacaram em aspectos específicos. Com base nesses resultados, atribuiu-se um score relativo a cada kernel, visando selecionar três representantes (baixa, média e alta complexidade) para comparação com Redes Neurais (NN) e Expansão em Caos Polinomial (PCE).

Foram selecionados os três melhores kernels, garantindo que o melhor desempenho

dos GPs estivesse presente no comparativo com os demais emuladores. Considerando as médias dos scores das quatro QoIs, as Figuras 31 e 32 mostram que, no Modelo A, o kernel Poly+RBF apresentou o melhor desempenho, enquanto o Poly+Matern32, em segundo lugar, não apresentou diferença estatisticamente significativa em relação ao primeiro. Já o terceiro colocado, Poly, obteve desempenho significativamente inferior aos dois primeiros. No Modelo B, o melhor kernel foi o Poly, significativamente superior aos demais, seguido por Poly+RBF e Poly+Matern32, que não diferem entre si, mas se destacam em relação aos outros kernels analisados.

Como os três melhores kernels foram os mesmos em ambos os modelos e apresentaram diferenças estatisticamente relevantes em relação aos demais, definiu-se o Poly como kernel de baixa complexidade por ser simples e não composto, o Poly+RBF como de média complexidade por resultar da soma de dois kernels e o Poly+Matern32 como de alta complexidade por também ser composto, mas possuir maior número de parâmetros que o Poly+RBF.

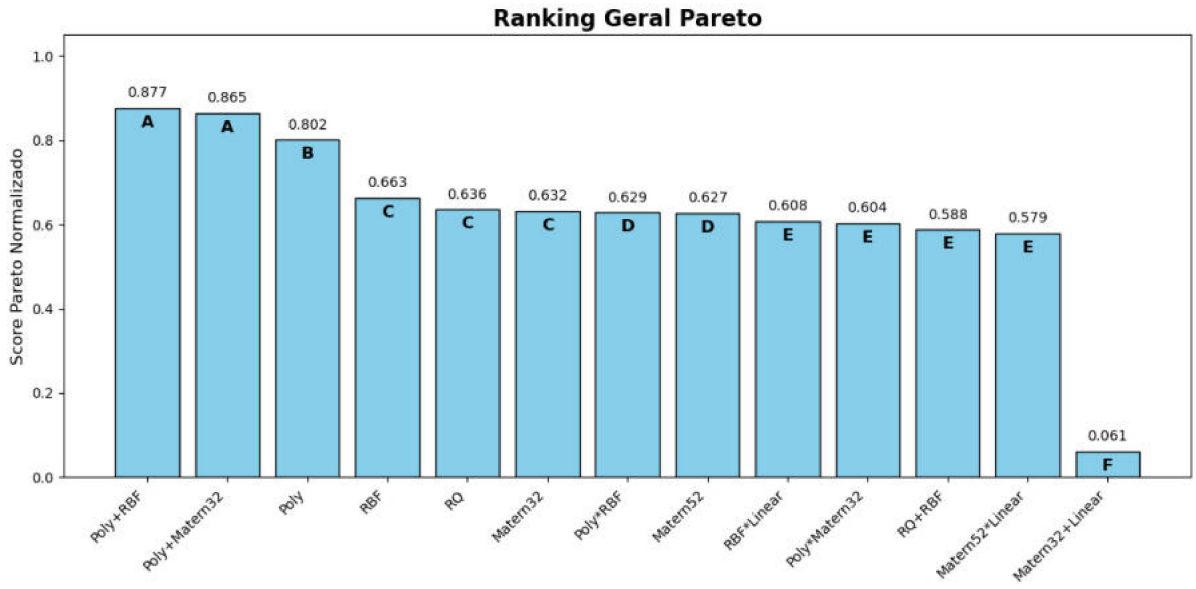

Figura 31 – Gráfico de barras mostrando a nota obtida em cada kernel do Modelo A de acordo com o score de Pareto. As colunas apresentam letras que, se diferentes, indicam que as notas são significativamente diferentes entre si.

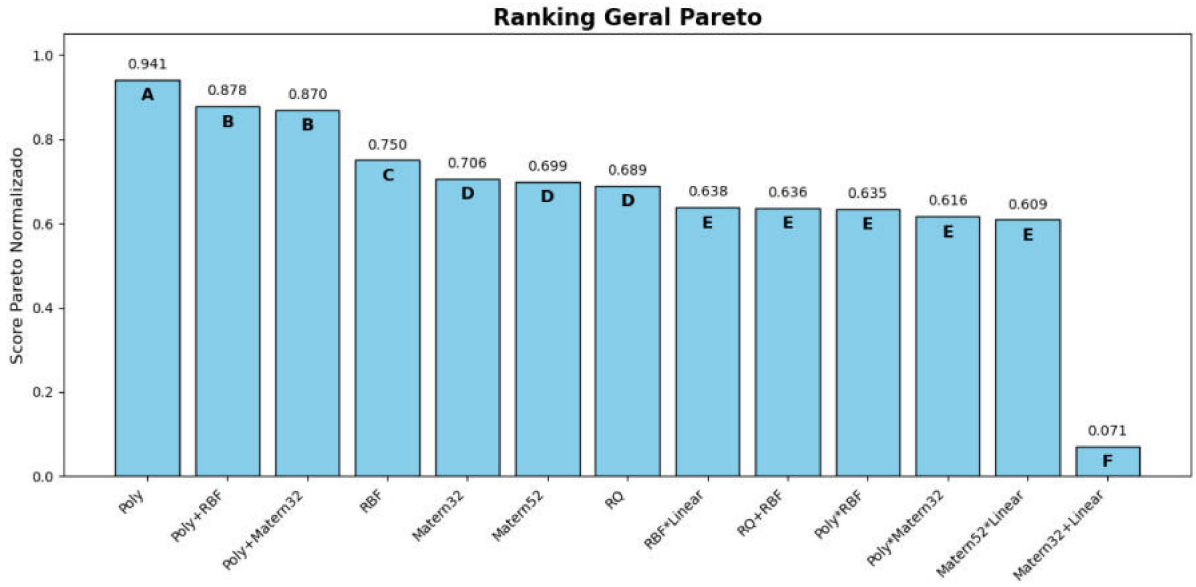

Figura 32 – Gráfico de barras mostrando a nota obtida em cada kernel do Modelo B de acordo com o score de Pareto. As colunas apresentam letras que, se diferentes, indicam que as notas são significativamente diferentes entre si.

As Figuras 33 e 34 confirmam as escolhas feitas a partir da análise global do score de Pareto com as médias das QoIs. A avaliação individual de cada variável evidencia uma consistência marcante, já que os três kernels de melhor desempenho global permanecem entre os três primeiros em todas as quantidades de interesse, mantendo significância estatística em relação aos demais. A análise individualizada por QoI confirma também a consistência observada no pior desempenho, uma vez que o Matern32+Linear permaneceu na última posição em todas as quantidades de interesse nos dois modelos. Já os kernels de desempenho intermediário exibiram maior variabilidade de colocação, alternando suas posições de acordo com a QoI analisada, comportamento esperado diante da proximidade entre seus scores de Pareto.

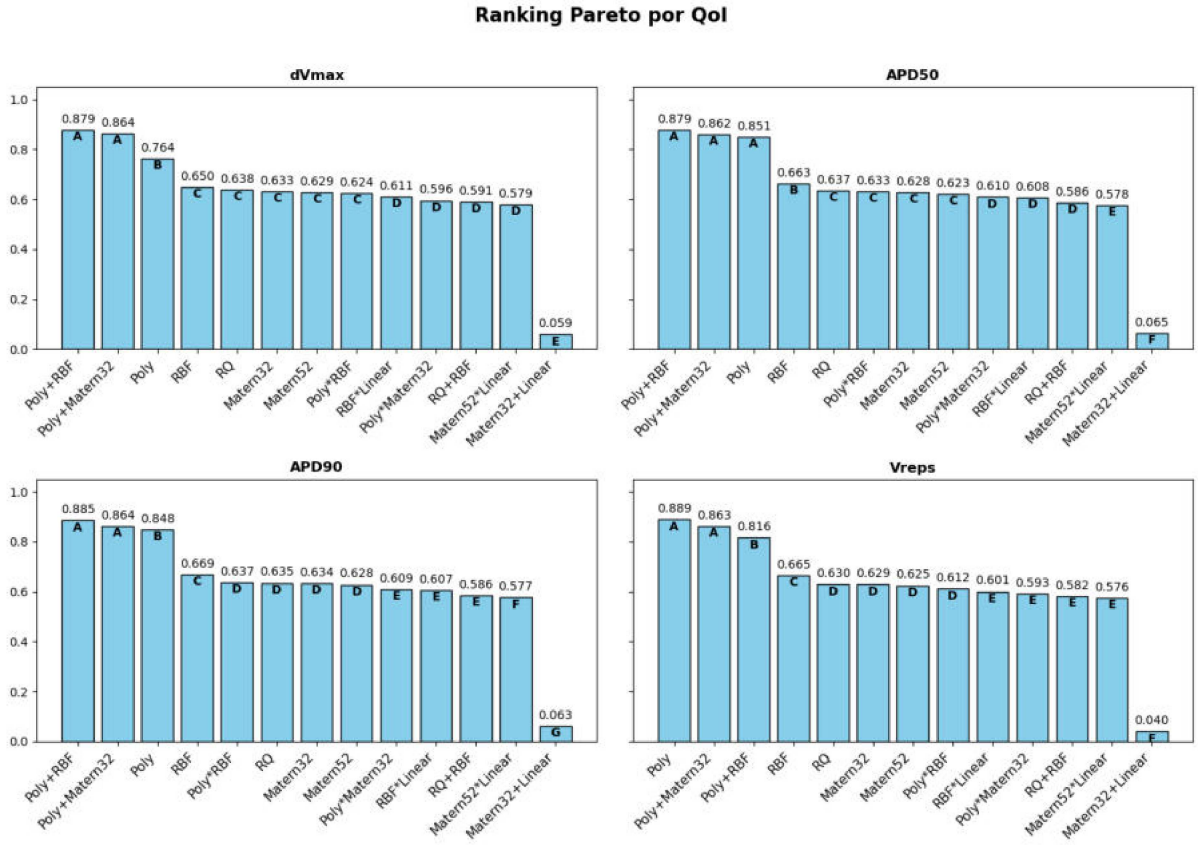

Figura 33 – Gráficos de barras mostrando a nota obtida em cada kernel, em cada uma das quatro quantidades de interesse, do Modelo A de acordo com o score de Pareto. As colunas apresentam letras que, se diferentes, indicam que as notas são significativamente diferentes entre si.

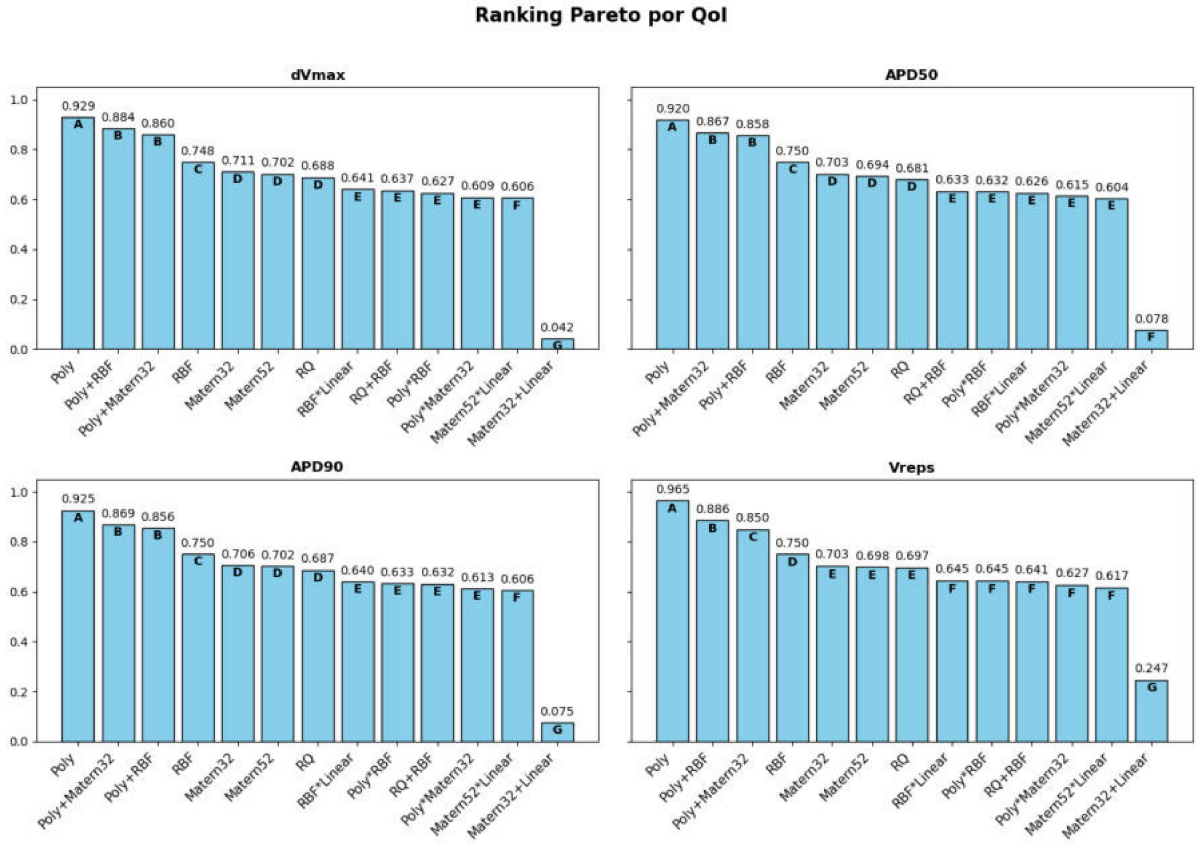

Figura 34 – Gráficos de barras mostrando a nota obtida em cada kernel, em cada uma das quatro quantidades de interesse, do Modelo B de acordo com o score de Pareto. As colunas apresentam letras que, se diferentes, indicam que as notas são significativamente diferentes entre si.

5.5 Comparativo entre os emuladores

Os resultados a seguir são referentes ao comparativo entre os emuladores GP utilizando as bibliotecas GPyTorch e scikit-learn, NN e PCE. Cada abordagem foi subdividida em três níveis de complexidade: baixa, média e alta como mostra a Tabela 4. As análises foram realizadas separadamente para os dois modelos, Modelo A e Modelo B. Para cada modelo, os resultados são apresentados em quatro barras agrupadas, representando os diferentes tamanhos amostrais de treino, sendo eles: 100, 500, 1000 e 5000.

Tabela 4 – Resumo dos emuladores e suas configurações de complexidade.

Emulador	**Complexidade**		
	Pequena	**Média**	**Grande**
Redes Neurais	1 camada e 16 neurônios	2 camadas e 32 neurônios	4 camadas e 64 neurônios
Processos Gaussianos: GPyTorch	Poly	Poly+RBF	Poly+Matern32
Processos Gaussianos: scikit-learn	Poly	Poly+RBF	Poly+Matern32
Expansão de Caos Polinomial	Grau 2	Grau 3	Grau 5

As métricas com pouca variação individual quando se analisa as quatro quantidades de interesse, como memória alocada na GPU, tempo de inferência e tempo de treino, foram representadas pelas médias. Já os gráficos que envolvem a métrica MSE, foram normalizados e apresentados separadamente para cada QoI, assim como o score de Pareto, para permitir uma análise individualizada.

5.5.1 Análise da alocação de memória na GPU

Nas Figuras 35 e 36, observa-se que as redes neurais (NN) não exigiram maior alocação de memória na GPU com o aumento do tamanho amostral ou da complexidade do emulador. Em contraste, os GPs apresentaram aumento na alocação de memória à medida que o tamanho amostral cresce, mas não houve uma tendência ao analisar o efeito da complexidade do modelo, isso evidencia que este emulador é sensível ao aumento amostral, contrastando com as NNs.

No caso dos GPs do Modelo A, Figura 35, verifica-se que, em geral, os emuladores apresentaram valores semelhantes para os tamanhos amostrais de 100 e 1000. Porém, para o tamanho amostral de 500, o GP_P foi o mais custoso, necessitando de mais memória do que os outros emuladores, incluindo os GPs com tamanho amostral igual a 1000. Para o tamanho amostral de 5000, o GP_G foi o menos custoso, seguido pelo GP_P e, por último, o GP_M, que exigiu a maior alocação de memória.

Ao analisar os GPs do Modelo B, Figura 36, nota-se que, de maneira geral, os emuladores apresentaram resultados semelhantes entre os diferentes tamanhos amostrais, com uma tendência de aumento proporcional à medida que o tamanho amostral cresceu. O GP_P se destacou por ser o mais custoso em todos os tamanhos amostrais analisados, já os outros dois possuem comportamento semelhante.

Esse resultado pode estar associado às particularidades do uso de ARD nos GPs, uma vez que cada dimensão de entrada possui seu próprio hiperparâmetro de escala. No caso do kernel Poly, cada dimensão do input recebe um hiperparâmetro de escala ajustado individualmente, mas a ausência de termos estacionários faz com que a matriz de covariância dependa de todos os termos polinomiais de forma isolada, elevando a quantidade de memória necessária. Nos kernels compostos, por outro lado, o componente estacionário (RBF ou Matérn) atua como fator de amortecimento, pois compartilha caches de blocos de covariância entre as dimensões, reduzindo assim a alocação de memória.

Cabe ressaltar que os GP_SKs e os PCEs não utilizam GPU para a realização dos cálculos de treino e predição, motivo pelo qual os valores de memória alocados por esses emuladores permanecem zerados.

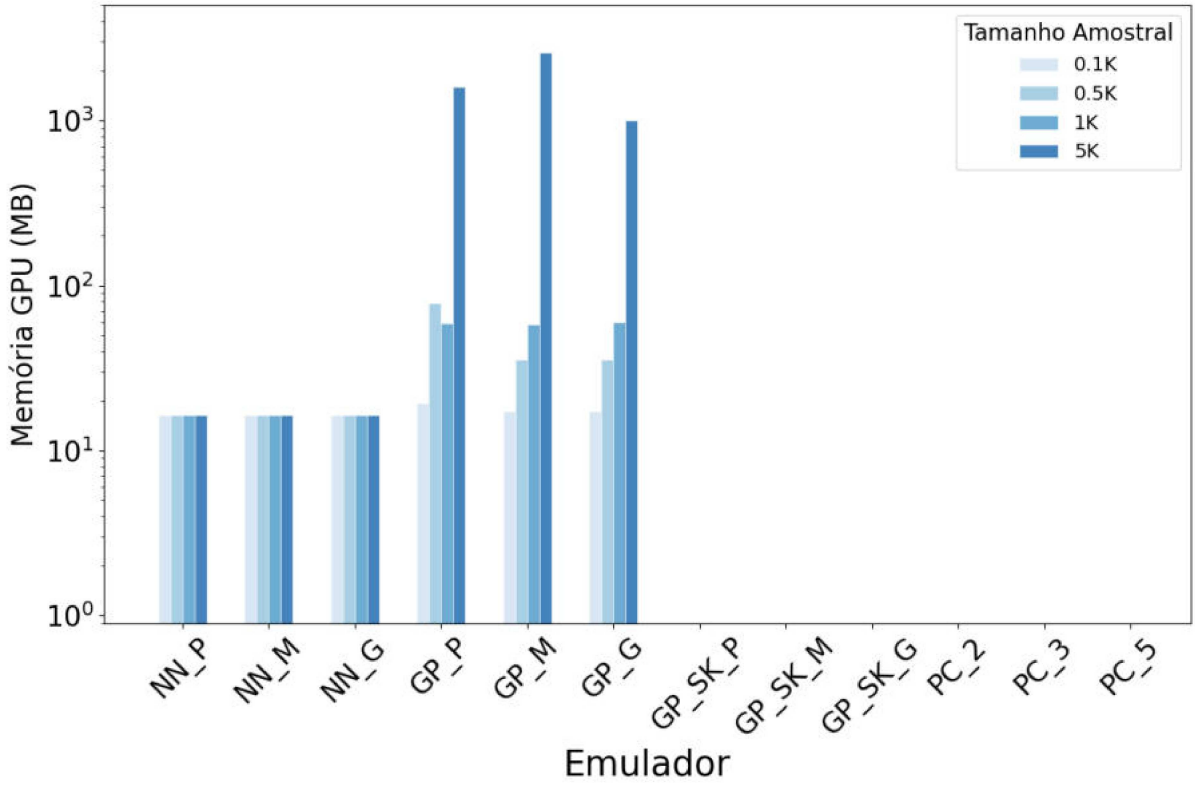

Figura 35 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador do Modelo A, mostrando a quantidade de memória (em MB) alocada na GPU para realizar a inferência dos modelos. Modelo GP_SK e PC não utilizam GPU.

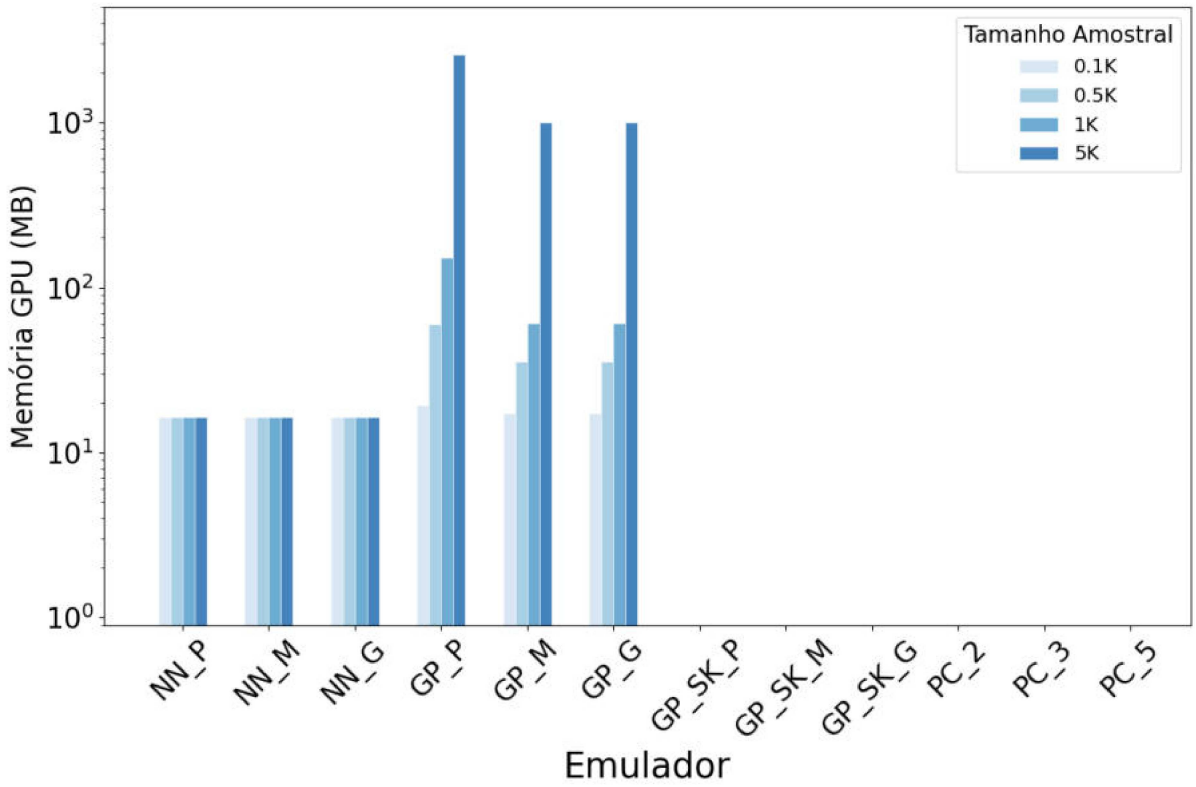

Figura 36 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador do Modelo B, mostrando a quantidade de memória (em MB) alocada na GPU para realizar a inferência dos modelos. Modelo GP_SK e PC não utilizam GPU.

5.5.2 Análise do tempo de treinamento

Nas Figuras 37 e 38 são apresentados os tempos de treino, em segundos, necessários para cada emulador nos diferentes tamanhos amostrais.

Analisando a Figura 37, que apresenta os resultados do Modelo A, observa-se que as PCEs e o GP_SK_P se destacaram em relação aos demais emuladores. Esses modelos apresentaram tempos de treino aproximadamente um grau de magnitude menores que as NNs e cerca de dois graus de magnitude menores que alguns GPs. Entre eles, o destaque principal são as PCEs, que mantiveram tempos de treino praticamente estáveis com o aumento do tamanho amostral. Foi observado apenas um leve declínio após o menor tamanho amostral no PC_2 e um pequeno aumento no maior tamanho amostral para as demais PCEs.

Os emuladores GP_SK_P e GP_SK_G apresentaram tendência geral de aumento do tempo de treino à medida que o tamanho amostral aumentou. Esse padrão não foi observado no GP_SK_M, que exibiu um comportamento irregular, com aumento inicial seguido de queda e posterior novo aumento. Em termos de desempenho positivo, destacam-se os casos de menor tamanho amostral, onde GP_SK_P e GP_SK_G foram mais rápidos

para treinar do que todos os GPs e NNs. Contudo, no caso de maior tamanho amostral, GP_SK_G apresentou o tempo de treino mais elevado entre todos os emuladores avaliados.

O comportamento das NNs mostrou-se oscilante em relação ao aumento do tamanho amostral. A NN_P apresentou o maior tempo de treino para o tamanho 100, a NN_M exibiu tendência de redução no tempo de treino com o aumento amostral e a NN_G apresentou variações, com queda inicial, seguida de aumento e nova queda. Essa ausência de padrão também foi observada nos GPs, em que o GP_M se destacou como o mais oneroso em termos médios de tempo de treino.

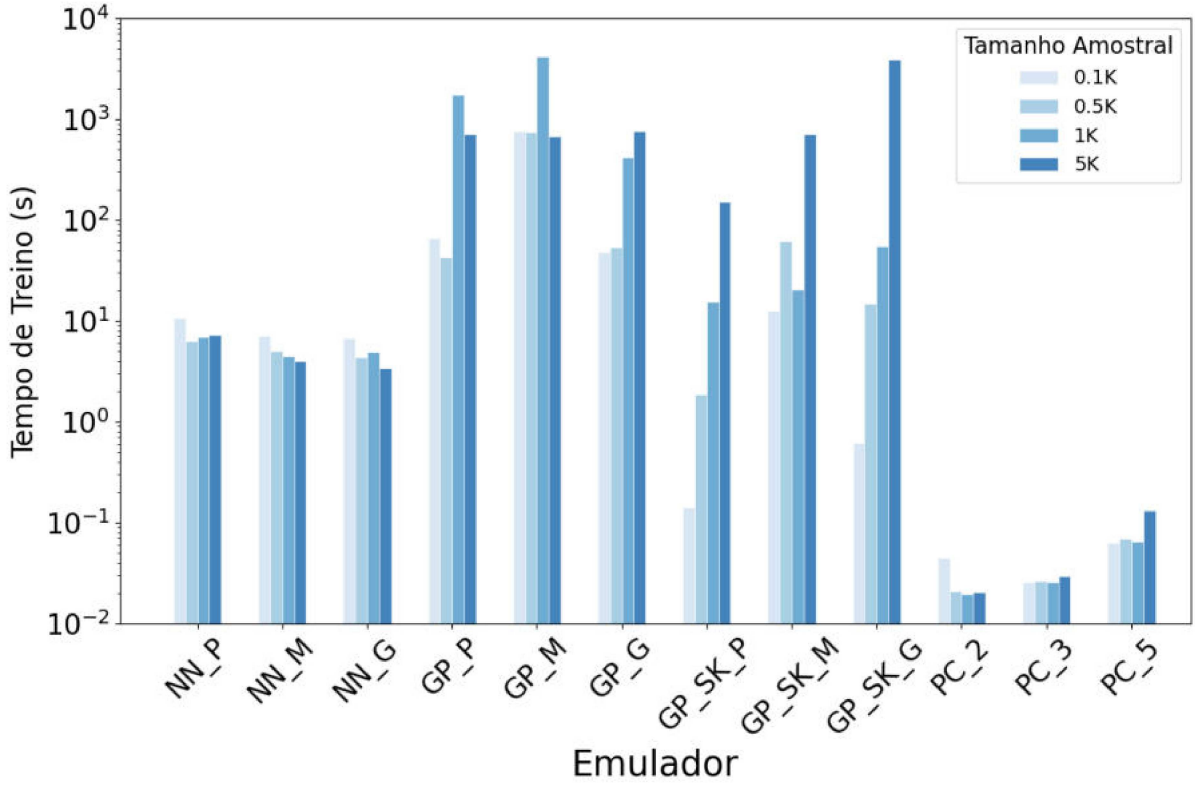

Figura 37 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador do Modelo A, mostrando o tempo, em segundos, necessário para treinar o emulador.

Ao analisar a Figura 38, que apresenta os resultados do Modelo B, observa-se que o PC_2 e o GP_SK_P se destacaram em relação aos outros emuladores. Esses modelos apresentaram tempos de treino aproximadamente um grau de magnitude menor do que as redes neurais (NNs) e cerca de dois graus de magnitude menores do que alguns emuladores GP.

No caso dos PCs, verifica-se um padrão de leve aumento no tempo de treino conforme o tamanho amostral aumenta. Ao comparar as diferentes complexidades, é evidente que, à medida que a complexidade cresce, o tempo de treino também aumenta.

Essa relação é mais pronunciada no Modelo B do que no Modelo A, indicando que o aumento no número de variáveis de treino tem um impacto mais significativo no tempo de treinamento do Modelo B.

Os emuladores GP_SKs apresentaram uma tendência geral de aumento do tempo de treino com o aumento do tamanho amostral. O salto entre os diferentes tamanhos amostrais foi o mais acentuado entre todos os emuladores, mostrando que esse tipo de emulador é o mais sensível ao aumento amostral. Em termos de desempenho positivo, destaca-se o GP_SK_P com o menor tamanho amostral, sendo o emulador mais rápido de se treinar. No entanto, o GP_SK_G, com o maior tamanho amostral, apresentou, novamente, como no Modelo A, o tempo de treino mais elevado entre todos os emuladores avaliados.

As NNs apresentaram um comportamento inconsistente em relação ao aumento do tamanho amostral, mas mantiveram os valores relativamente próximos. A NN_P, para o tamanho amostral de 500, se destacou negativamente, apresentando o maior tempo de treino entre as NNs. Os GPs, diferentemente do Modelo A, mostraram uma tendência mais consistente, com uma queda inicial seguida de aumento no tempo de treino à medida que o tamanho amostral cresceu. Embora essa tendência não tenha sido tão pronunciada quanto a observada para os emuladores GP_SKs, ela demonstra que os GPs, utilizando a biblioteca GPyTorch, escalam menos o tempo de treino conforme o aumento do tamanho amostral.

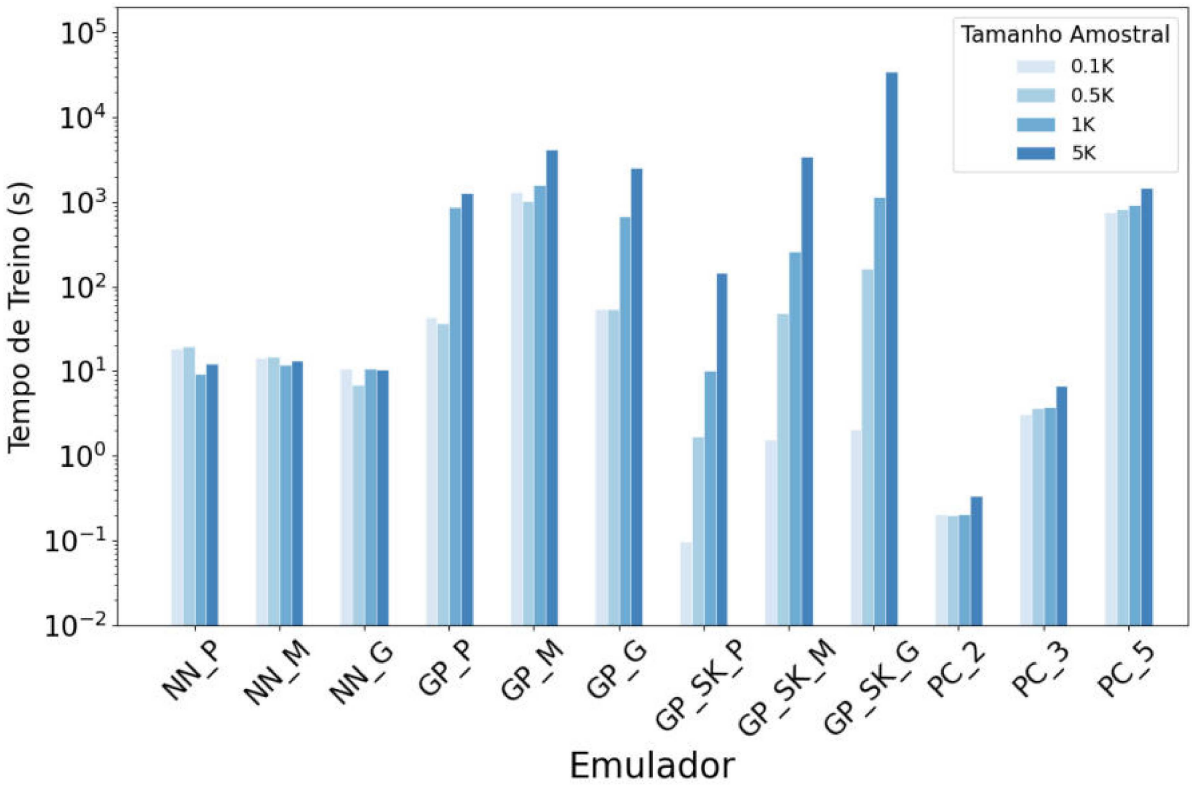

Figura 38 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador do Modelo B, mostrando o tempo, em segundos, necessário para treinar o emulador.

5.5.3 Análise do tempo para realizar inferência

Nas Figuras 39 e 40 são apresentados os tempos, em segundos, necessários para cada emulador realizar a inferência de 100 mil dados, para cada um dos diferentes tamanhos amostrais.

O Modelo A e Modelo B apresentam comportamento muito semelhante, por isso a análise será feita em comum aos dois. Inicialmente observa-se que as NNs se destacam por apresentarem os menores tempos de inferência. Esses tempos são, em geral, um grau de magnitude menor do que os de qualquer outro emulador e chegam a ser três graus de magnitude menores quando comparados, por exemplo, ao PC_5 no Modelo B e aos emuladores GP_SKs, em ambos os modelos. As NNs não apresentaram variações significativas no tempo de inferência em função do aumento do tamanho amostral ou da complexidade do emulador. Esse comportamento também foi observado nos PCs em relação ao tamanho amostral, porém, com o aumento da complexidade, houve um crescimento notável no tempo de inferência. Isso faz com que o PC_2 apresente desempenho significativamente superior ao PC_5, com uma diferença de aproximadamente um grau de magnitude no Modelo B e um pouco menor no Modelo A.

No caso dos GP_SKs, observa-se uma tendência clara de aumento no tempo de inferência conforme cresce o tamanho amostral, evidenciando alta sensibilidade a essa variável. Por outro lado, os GPs apresentaram comportamento distinto, pois nos três menores tamanhos amostrais os tempos de inferência foram bastante similares, indicando que a capacidade da GPU ainda não havia sido plenamente utilizada. Quando o tamanho amostral atinge 5000, esse limiar é ultrapassado e ocorre um leve aumento no tempo de inferência. Diante dessas análises, é evidente que os emuladores GP_SKs são mais sensíveis ao crescimento amostral, enquanto os GPs começam a ser impactados somente quando a carga de dados excede a capacidade de processamento da GPU. Em relação ao aumento da complexidade do emulador, ambos os Processos Gaussianos mantiveram um comportamento relativamente estável.

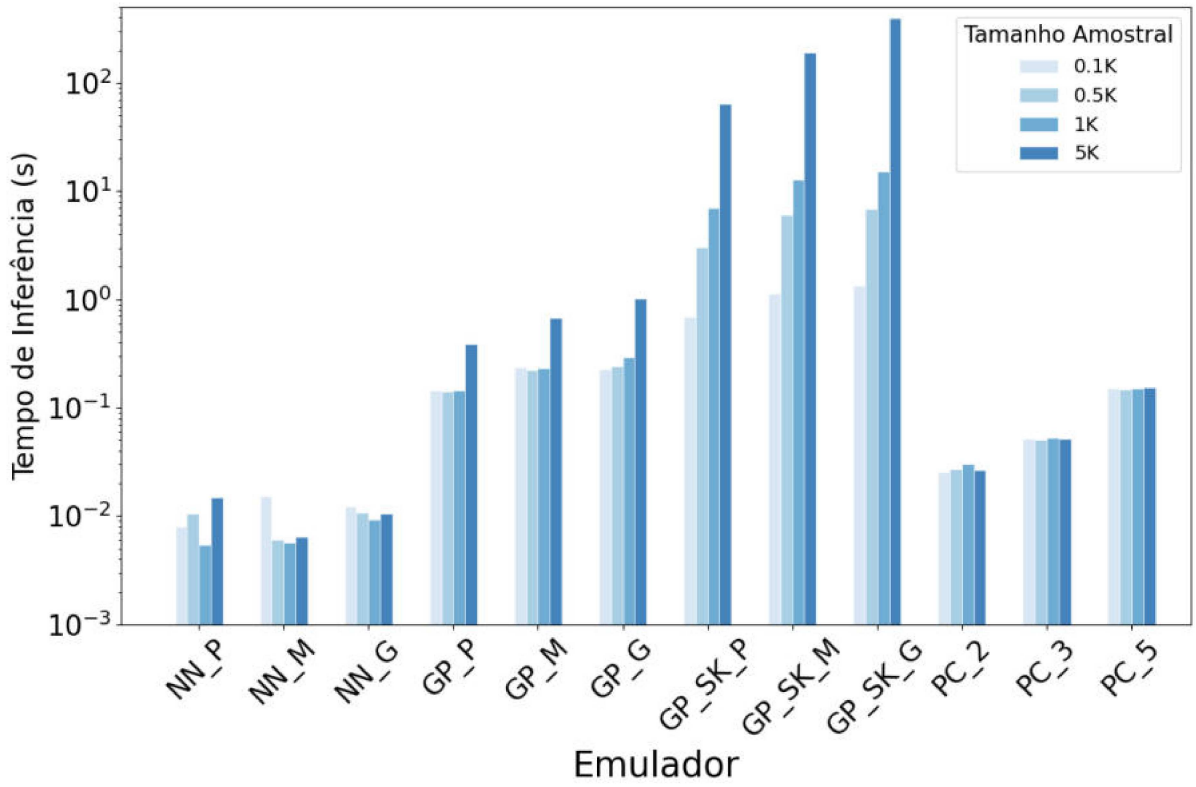

Figura 39 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador do Modelo A, mostrando o tempo, em segundos, necessário para inferir 100 mil amostras.

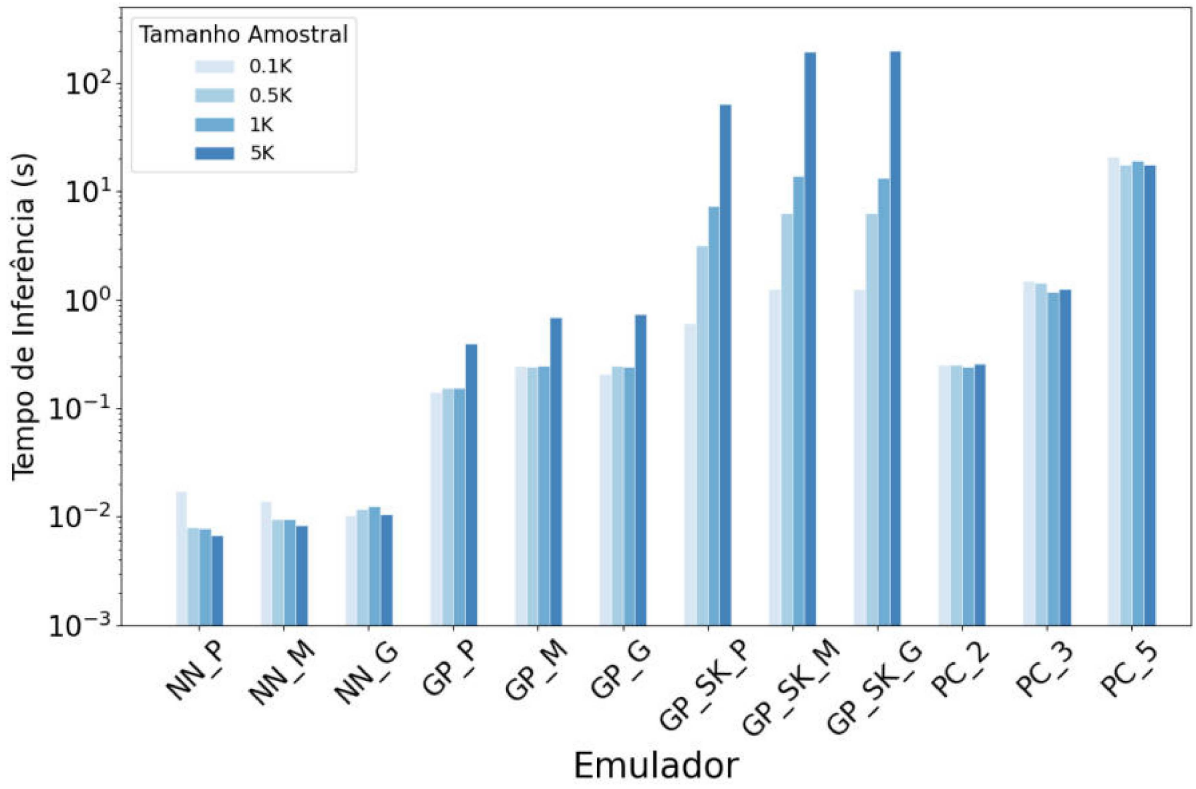

Figura 40 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador do Modelo B, mostrando o tempo, em segundos, necessário para inferir 100 mil amostras.

Após as análises do tempo de treinamento de cada emulador, Figuras 37 e 38, e do tempo de inferência, Figuras 39 e 40, percebe-se um comportamento contrastante dos Processos Gaussianos, pois, apesar de não utilizar paralelização em GPU, os GP_SKs se sobressaíram no tempo de treino frente aos GPs que utilizam a GPU para realizar cálculos. Esse comportamento é explicado pelo overhead de inicialização que a GPU possui, gerando um custo significativo ao carregar o contexto CUDA e transferir tensores de dados para a memória da GPU. Esse custo de configuração não é compensado nos tamanhos amostrais pequenos, como os de 100 e 500 pontos, o que faz com que os tempos de treino dos modelos GPyTorch permaneçam muito próximos e sejam mais lentos que os GP_SKs. Contrastando com o tempo de treino, o tempo de inferência dos Processos Gaussianos que utilizam GPU foi melhor do que os que utilizam apenas CPU, pois, ao inferir 100 mil dados, o custo do overhead é compensado.

5.5.4 Análise do erro quadrático médio (MSE)

Nas Figuras 41 e 42, são apresentados os erros de predição, medidos por MSE e, posteriormente, normalizados, para cada emulador e quantidade de interesse (QoI) em diferentes tamanhos amostrais.

Analisando a Figura 41, que representa os resultados do Modelo A, observando especificamente as QoIs APD50 e APD90, nota-se que o PC_5 apresenta o menor erro para o maior tamanho amostral. Em seguida, destacam-se as NNs e o GP_SK_G, para estas mesmas variáveis, que demonstram maior robustez frente ao aumento da complexidade do emulador, com ganhos discretos de acurácia. Esse padrão é observado especialmente na NN_P e no GP_SK_G, enquanto as demais NNs apresentam oscilações ao aumentar a amostra de treino. Por outro lado, os PCs mostram-se mais beneficiados pelo aumento da complexidade do emulador, embora o PC_2 não apresente melhora com o aumento do tamanho amostral, contrastando com o PC_3 e o PC_5, que demonstraram redução progressiva no erro. Os GPs e os demais GP_SKs não se destacaram, pois apresentam valores de erros mais altos e sem consistência em função da complexidade ou do tamanho amostral.

Para as QoIs dVmax e Vreps, observa-se que o GP_SK_G com o maior tamanho amostral obteve o menor MSE. Destaca-se que os GP_SKs se beneficiaram do aumento da complexidade do modelo, comportamento também observado nos PCEs. Entretanto, essa tendência não ficou evidente nos GPs e NNs, nos quais os GPs de baixa e a alta complexidade apresentam valores médios de erro superiores à complexidade média, enquanto as NNs mantêm média constante de MSE para a variável dVmax. Já para a Vreps, o aumento da complexidade prejudicou a acurácia das NNs.

Importante ressaltar o comportamento inconsistente do GP_M e do GP_SK_M, que apresentam variações no erro à medida que o tamanho amostral aumenta. Esses emuladores utilizam o kernel Poly+RBF, cuja composição por soma tende a gerar uma matriz de covariância mais sensível a ruídos numéricos, resultando em flutuações nas predições. Além disso, durante o ajuste de hiperparâmetros, o método de otimização pode convergir para mínimos locais, aumentando a instabilidade. Em contraste, os emuladores GP_G e GP_SK_G, que utilizam o kernel Poly+Matern32, não apresentam inconsistência significativa, provavelmente devido a uma matriz de covariância mais robusta e a ajustes de hiperparâmetros mais estáveis.

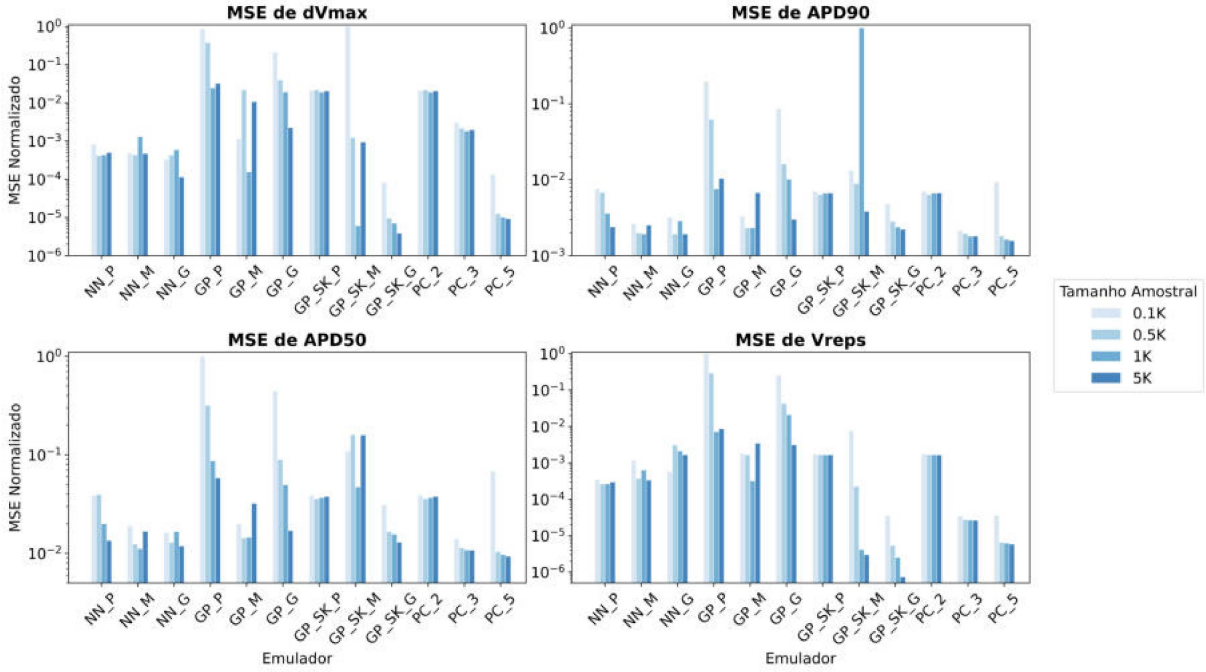

Figura 41 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador e por quantidade de interesse (QoI) do Modelo A, mostrando a métrica de erro quadrático médio (MSE) normalizado obtida pelo emulador ao inferir 100 mil amostras.

Ao analisar os resultados apresentados na Figura 42, que corresponde ao Modelo B, observa-se que o GP_SK_M com o maior tamanho amostral apresenta o menor erro em todas as QoIs, seguido pelo emulador GP_SK_G.

Analisando as variáveis APD50 e APD90, nota-se que as NNs melhoraram a acurácia com o aumento da complexidade do emulador, destacando a NN_G, que apresentou um bom desempenho com o aumento do tamanho amostral. Os demais emuladores não mostraram uma tendência de melhora com o aumento da complexidade, sendo possível notar que alguns valores de média complexidade se sobressaem aos de baixa e alta complexidade.

Na variável dVmax, é evidente a melhora dos emuladores com o aumento do tamanho amostral, sendo que somente NN_P e GP_M não apresentaram uma evolução gradual. Isso evidencia que, para essa quantidade de interesse, o tamanho amostral é influente na acurácia das predições. Tendência semelhante é observada na variável Vreps, embora as NNs tenham apresentado certa dificuldade de ajuste, exibindo oscilações com o aumento de complexidade e do tamanho amostral. Em contraste, os emuladores GP_SKs e PCEs de média e alta complexidade melhoraram consistentemente a acurácia conforme o tamanho amostral aumentou.

Assim como no Modelo A, o kernel GP_M apresentou comportamento inconsistente em todas as QoIs com o aumento do tamanho amostral, indicando que houve problemas no treino do emulador.

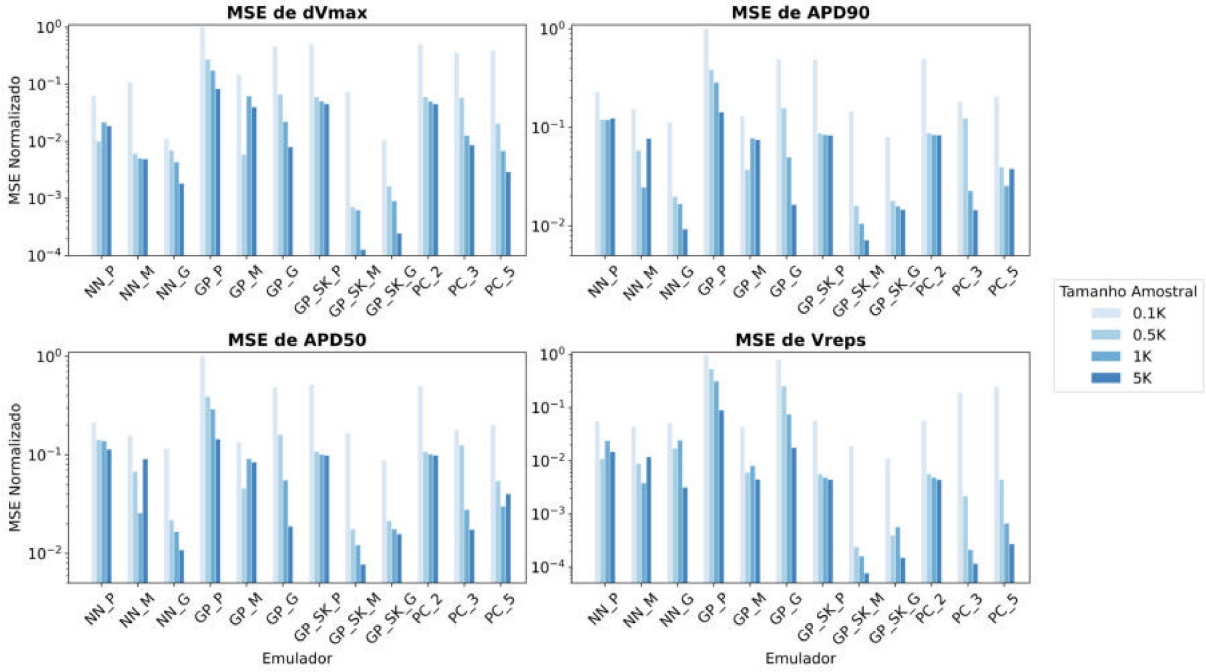

Figura 42 – Gráfico de barras, agrupando os diferentes tamanhos amostrais de treino por emulador e por quantidade de interesse (QoI) do Modelo B, mostrando a métrica de erro quadrático médio (MSE) normalizado obtida pelo emulador ao inferir 100 mil amostras.

Após a análise do MSE, nota-se que há uma grande variabilidade da acurácia conforme se altera o emulador. Visto isso, é importante realizar uma investigação para notar se esse comportamento da acurácia está ligado à formulação matemática dos dados simulados, que alimentaram o emulador, ou se é algo relacionado aos emuladores. Para essa análise, selecionou-se o Modelo A, por conter menos variáveis de entrada, facilitando assim o comparativo, e a quantidade de interesse (QoI) dVmax, que apresentou, em valores absolutos, o maior erro.

Na Figura 43, não se identificam padrões evidentes que indiquem anomalias na simulação dos dados da QoI dVmax. O único padrão observado é a correlação negativa entre a QoI e a variável de entrada Hipercalemia, pois à medida que o valor de entrada aumenta, há uma diminuição no valor simulado dVmax. Esse comportamento é corroborado pelo gráfico de correlação dessas variáveis mostrado na Figura 6.

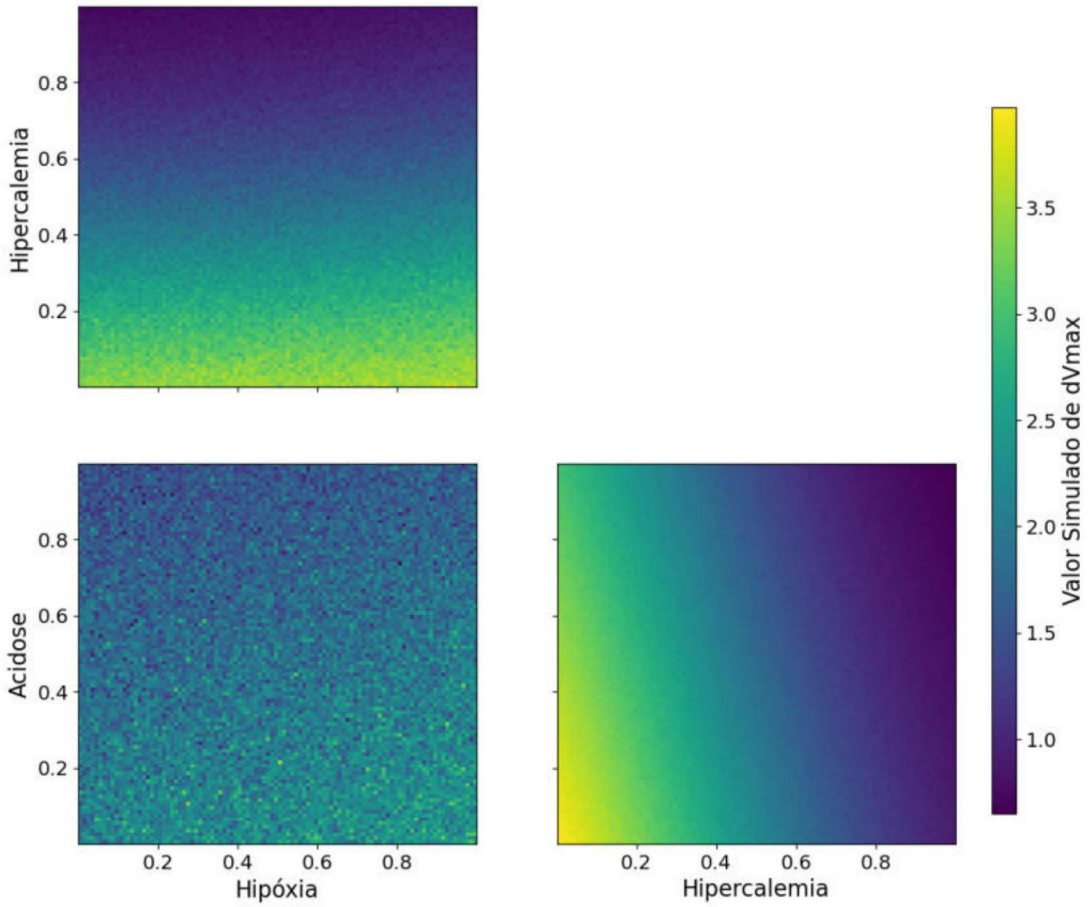

Figura 43 – Gráfico de calor par a par das variáveis de entrada do Modelo A com valores simulados da quantidade de interesse dVmax.

Como não há indícios de problemas nos dados simulados, a análise será direcionada aos emuladores. Para evidenciar os pontos fortes e fracos de predição de cada emulador, será realizado um comparativo entre o menos e o mais acurado, quando se utilizou o maior tamanho amostral de treino. Para os quatro emuladores, o menor e o maior MSE encontrados corresponderam, respectivamente, ao de menor e ao de maior complexidade.

Analisando as Figuras 44 e 45, nota-se na escala do erro absoluto que a NN_G é mais acurada que a NN_P. Além disso, as regiões de baixa acurácia variam muito entre os dois modelos, onde destaca-se, no emulador de baixa complexidade, Figura 44, que valores elevados de hipercalemia com valores extremos, positivo e negativo, de acidose possuem um alto erro absoluto. Esse comportamento também é observado com valores altos de hipóxia combinados com valores baixos de acidose. Já no emulador NN_G, Figura 45, há faixas de erro, em vez de valores mais pontuais, como pode ser notado quando se variam os valores de acidose ou hipóxia combinados com valores altos de hipercalemia.

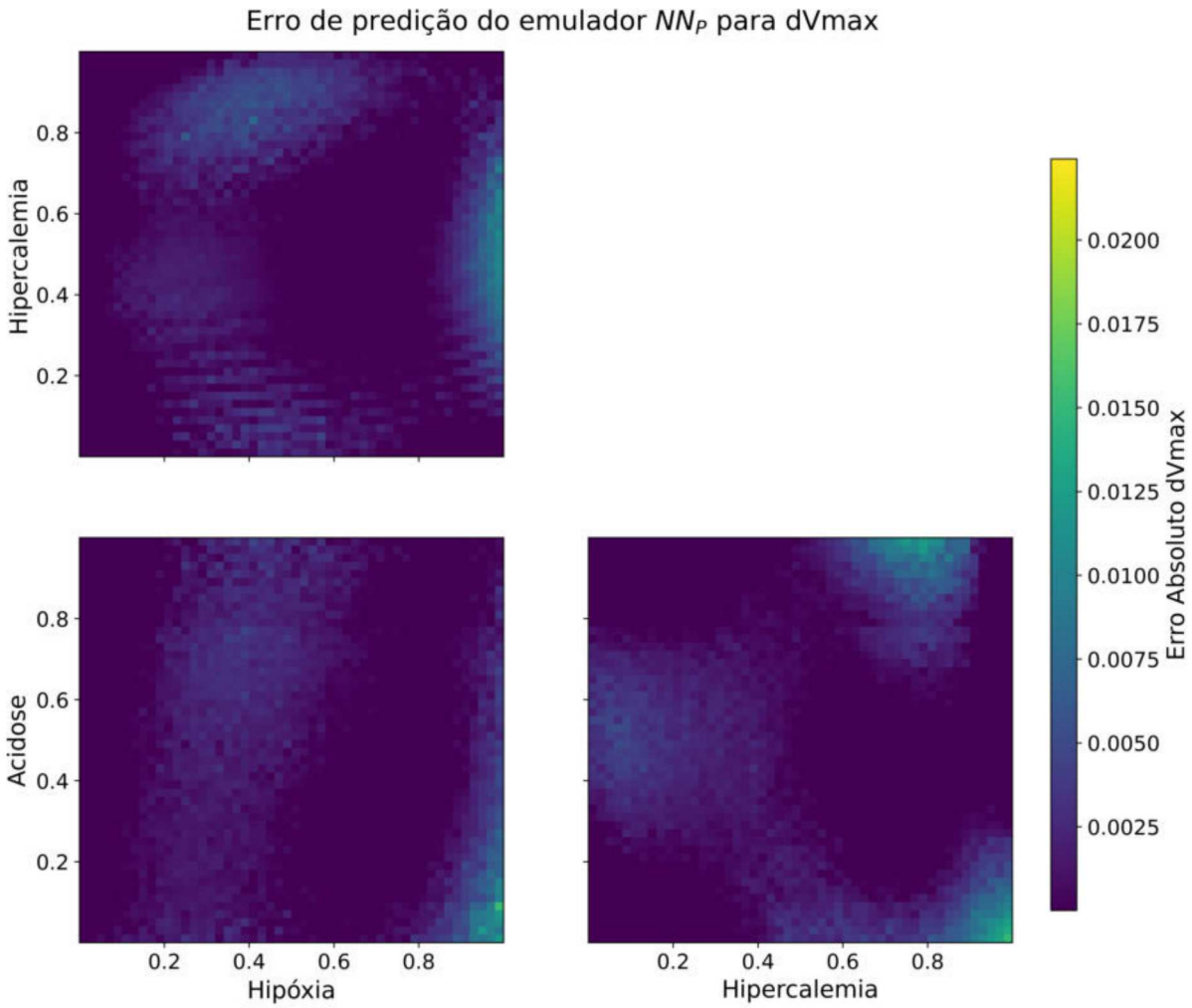

Figura 44 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador NN_P.

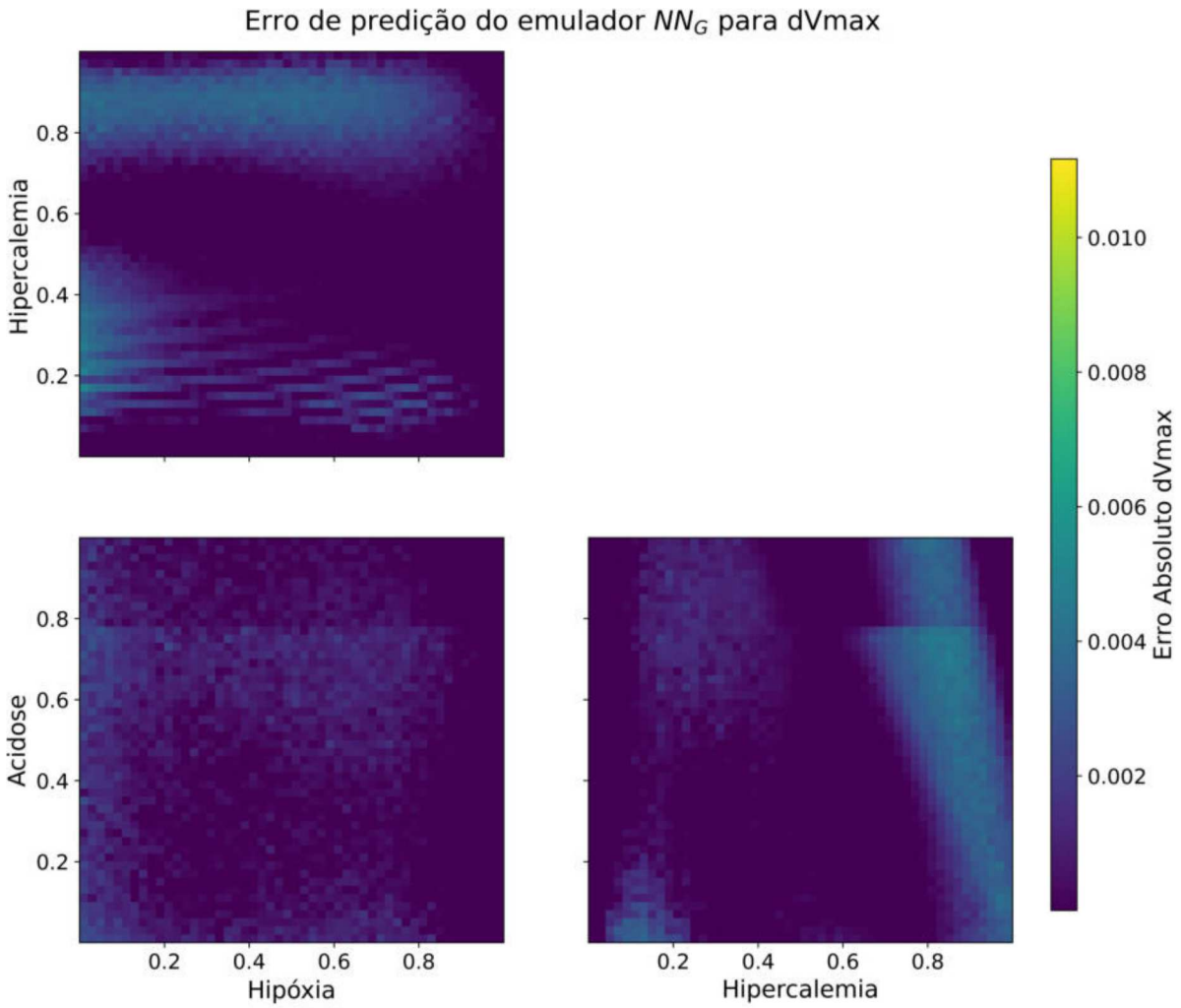

Figura 45 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador NN_G.

Observando as Figuras 46 e 47, verifica-se na escala do erro absoluto, que o emulador mais complexo é mais acurado que o menos complexo. No GP_P, Figura 46, os erros são evidentes em determinadas faixas de valores de hipercalemia, dando destaque para os valores elevados onde o emulador foi menos acurado. Já no GP_G, Figura 47, os erros ficam mais concentrados, sendo maiores nos valores extremos de hipercalemia.

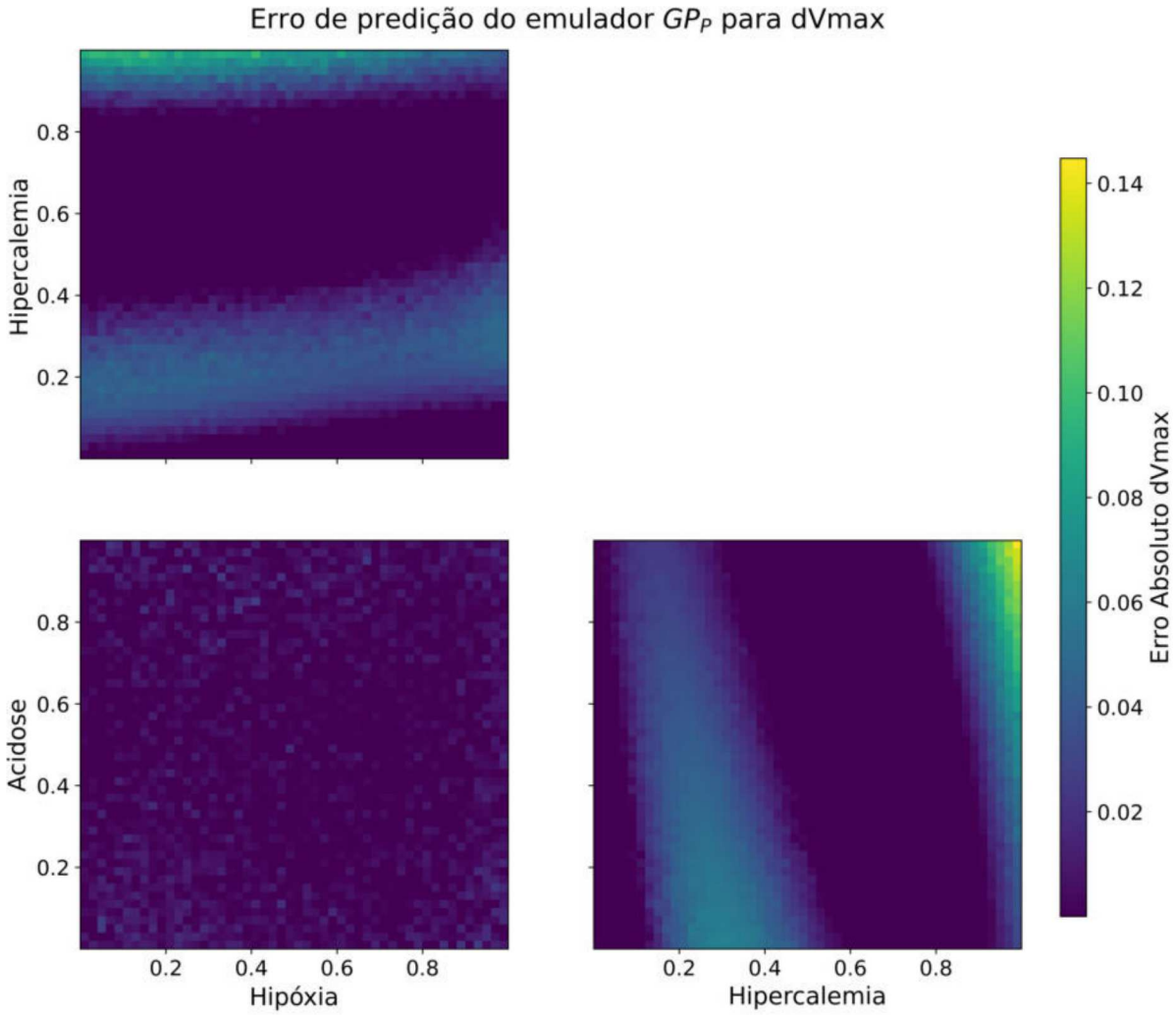

Figura 46 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador GP_P.

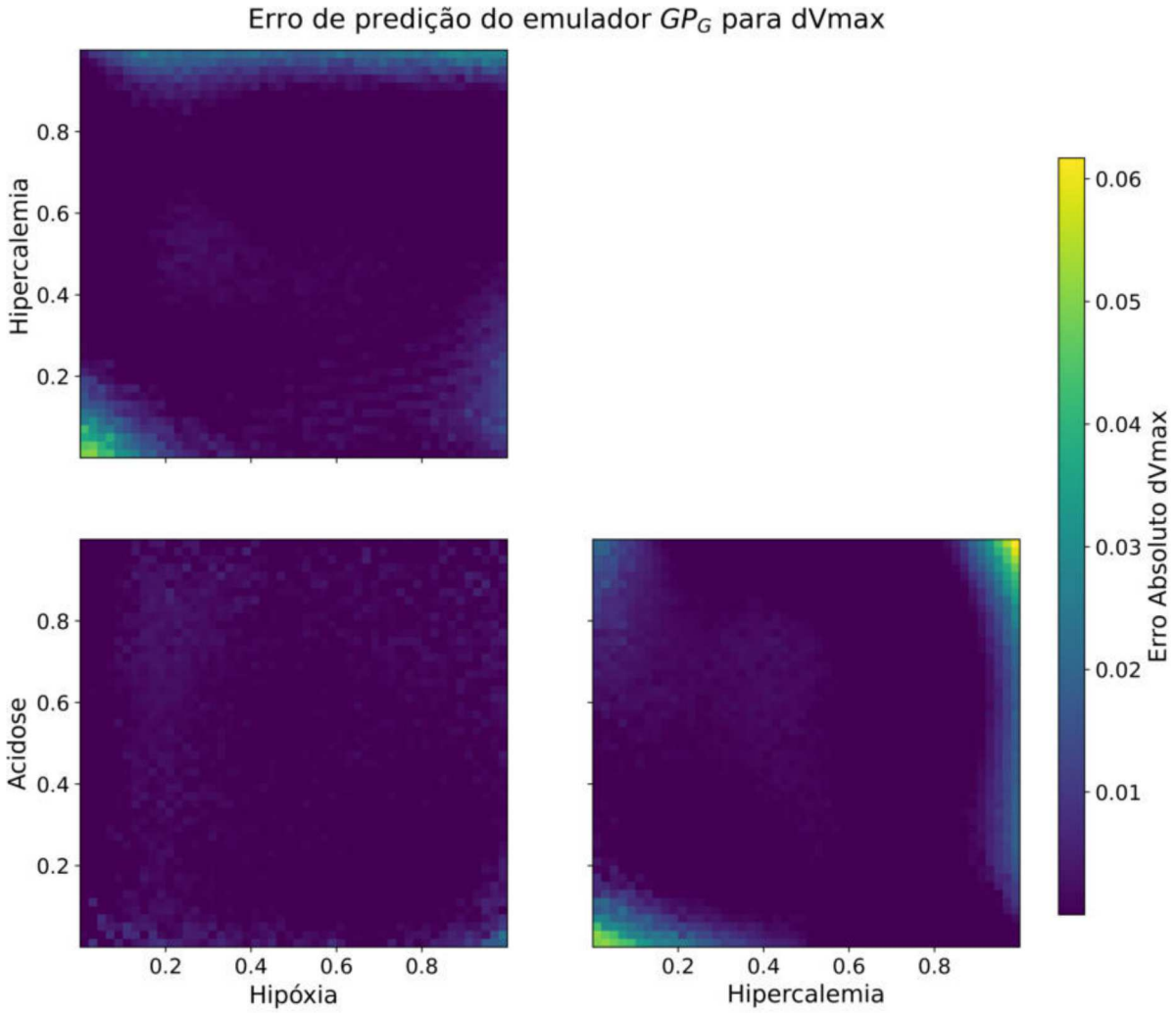

Figura 47 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador GP_G.

Investigando as Figuras 48 e 49, verifica-se na escala do erro absoluto que o emulador mais complexo é muito mais acurado. No GP_SK_P, Figura 48, os erros são bem determinados por certas faixas de valores de hipercalemia, já no GP_SK_G, Figura 49, eles são bem pontuais e não possuem um padrão definido. Além disso, é observado como as demais regiões apresentam erros absolutos próximos a zero, justificando o valor de MSE observado na Figura 41, onde este emulador se destaca por apresentar o menor erro quadrático médio dentre todos os outros.

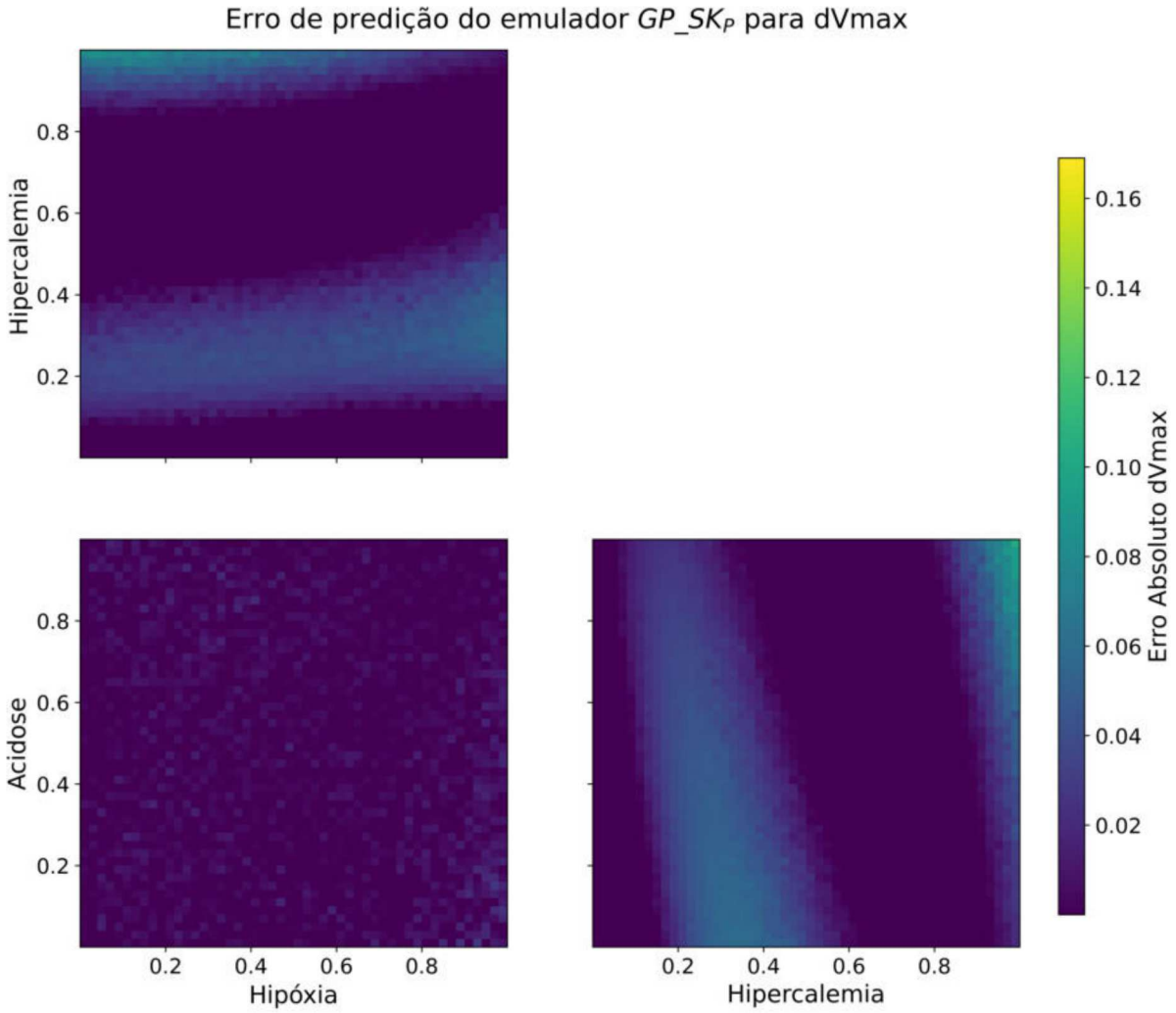

Figura 48 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador GP_SK_P.

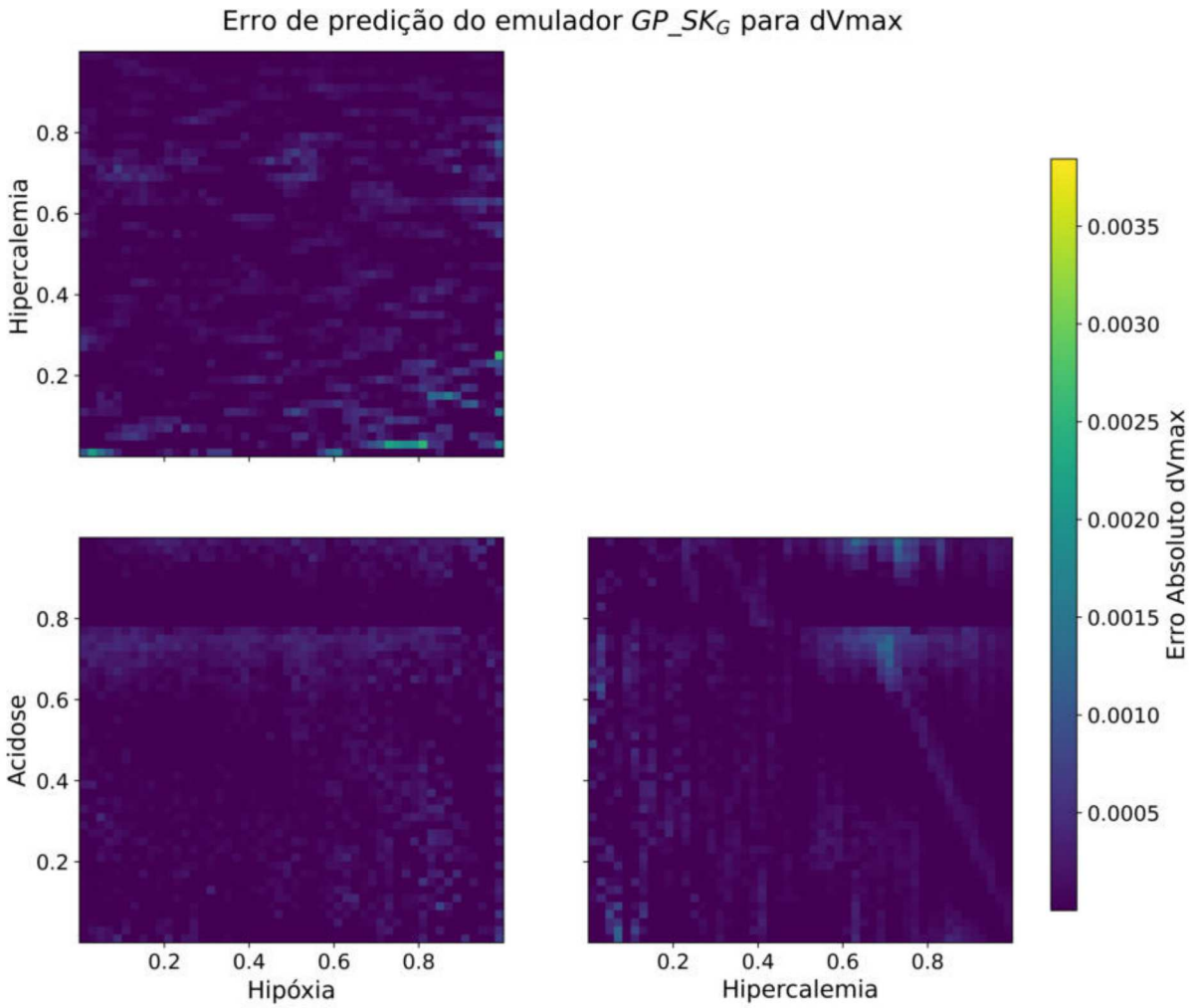

Figura 49 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador GP_SK_G.

Analisando as Figuras 50 e 51, nota-se pela escala do erro absoluto que o emulador mais complexo é muito mais acurado. Além disso, as regiões de baixa acurácia variam entre os dois modelos, sendo que no PC_2, Figura 50, ela se restringe a faixas específicas de hipercalemia, apresentando maiores erros quando essa variável tem valores mais altos. Já no PC_5, Figura 51, os erros não possuem um padrão definido, sendo observada diversas faixas variando alta e baixa acurácia em todas as variáveis de entrada.

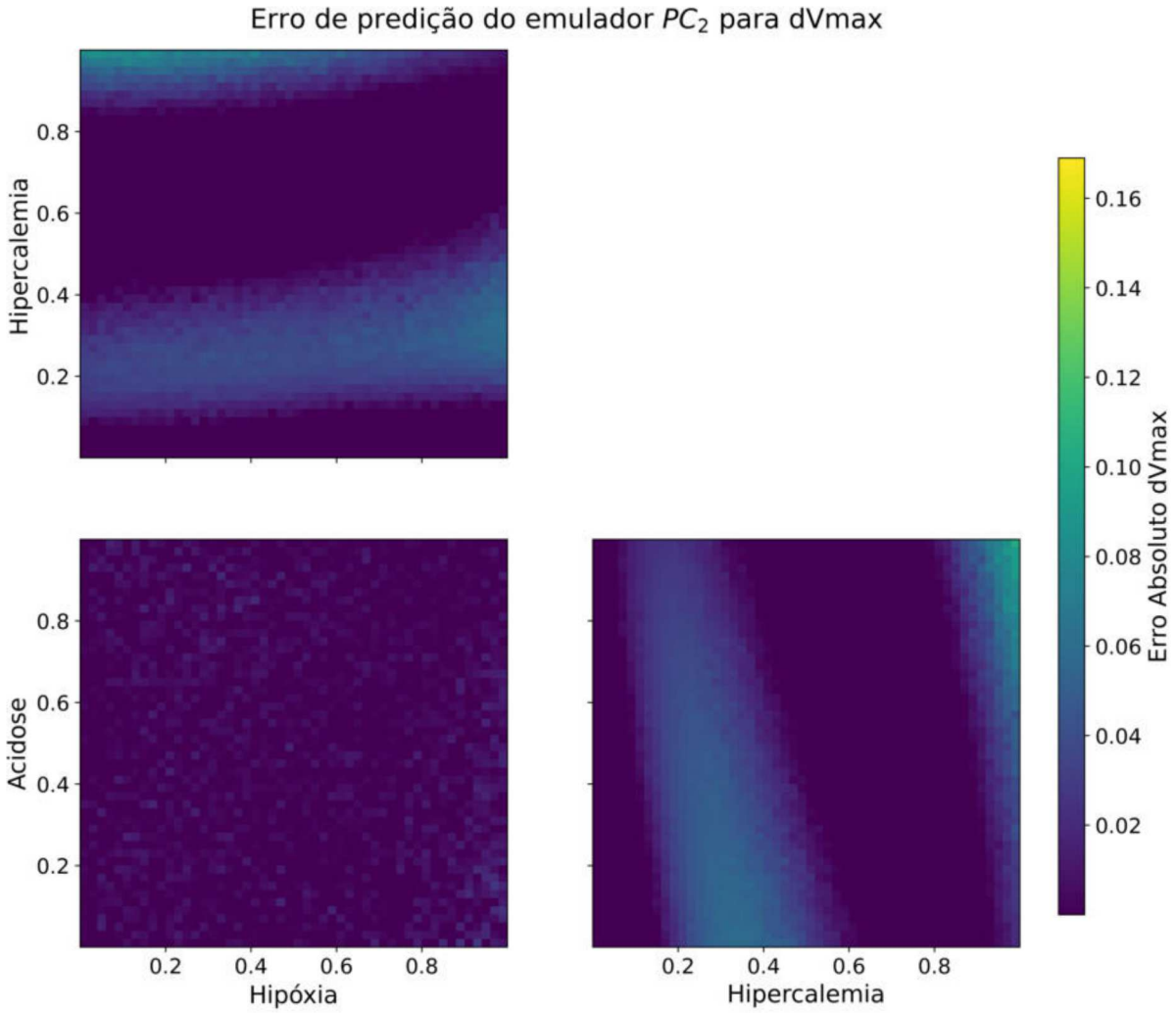

Figura 50 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador PC_2.

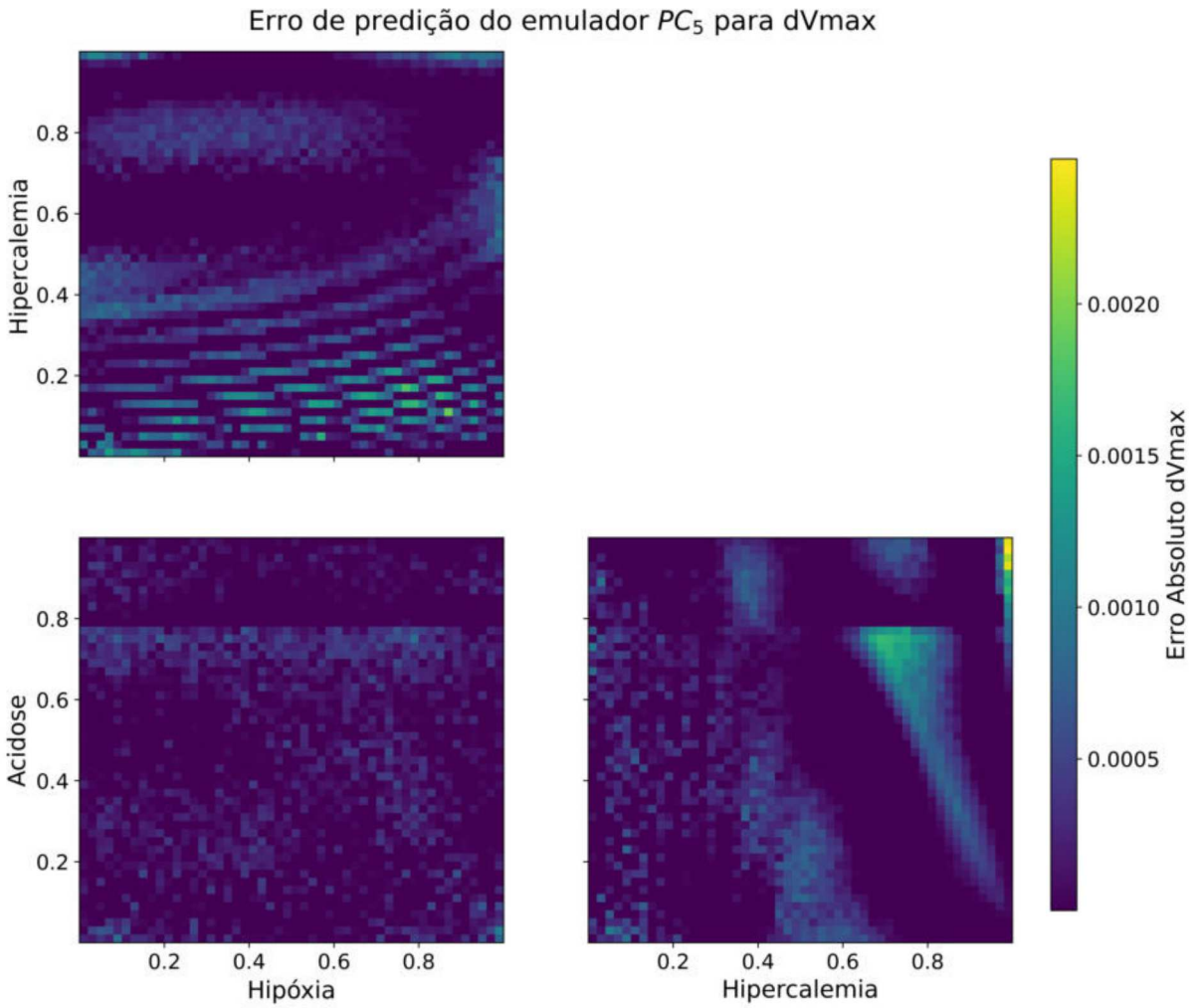

Figura 51 – Gráfico de calor par a par das variáveis de entrada do Modelo A com o erro absoluto ao predizer a quantidade de interesse dVmax, utilizando o emulador PC_5.

Ao comparar os diversos gráficos de calor que apresentam o erro absoluto na predição da QoI dVmax, observa-se que cada emulador possui características próprias na captura das interações entre as variáveis de entrada e saída, o que explica a variação de acurácia em diferentes faixas de valores.

5.5.5 Análise da correlação do MSE com o tempo de inferência

Nas Figuras 52 e 53, é mostrado o comportamento de cada emulador, para cada uma das quantidades de interesse (QoI), em relação à correlação entre o MSE e o tempo necessário para inferir 100 mil amostras.

Analisando a Figura 52, que apresenta os resultados do Modelo A, observa-se uma semelhança entre dois pares de QoIs dVmax e Vreps, APD50 e APD90. Analisando o primeiro par, nota-se que as NNs possuem o melhor tempo de inferência, porém, não apresentam valores tão baixos de MSE quanto é observado nos PCs e GP_SKs. Isso

evidencia que, cada um desses emuladores, possuem um atributo mais proeminente, variando entre acurácia e eficiência. Já os GPs, apresentam desempenho mediano para o tempo de inferência, mas valores medianos a elevados de MSE.

Observando o segundo par de QoIs, APD50 e APD90, destaca-se o aglomerado de pontos azuis, correspondentes às NNs, com baixos valores tanto de MSE quanto de tempo de inferência. Isso destaca as NNs como os melhores candidatos para serem o melhor emulador dessas QoIs desse modelo, combinando boa acurácia e eficiência. Embora os PCs não atinjam os mesmos baixos tempos de inferência, eles apresentam valores de MSE próximos aos das NNs, o que os torna competitivos em termos de acurácia. Quanto aos emuladores GPs e GP_SKs, ambos apresentam tempos de inferência e valores de MSE elevados. No entanto, os GPs têm um desempenho superior aos GP_SKs, evidenciando que, para o Modelo A, os GPs são ligeiramente mais eficientes em termos de acurácia e tempo de inferência, apesar de não se destacarem tanto quanto as NNs ou os PCs.

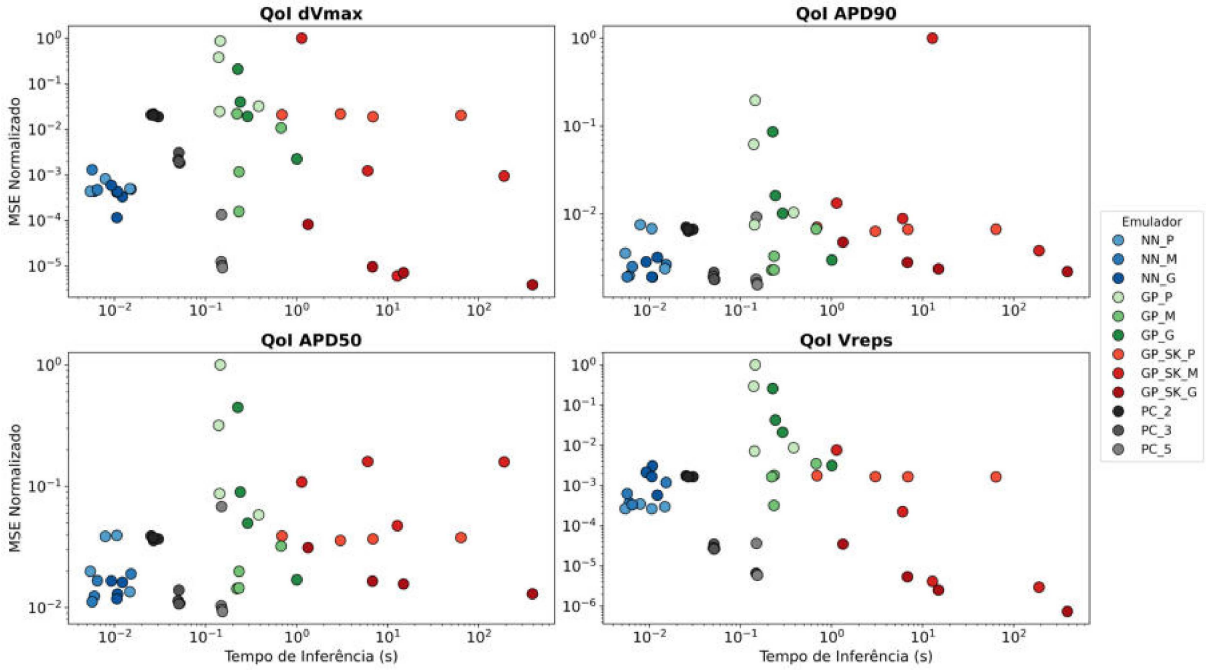

Figura 52 – Gráfico de dispersão evidenciando a correlação entre o MSE (Normalizado) e o tempo necessário, em segundos, para cada emulador do Modelo A inferir 100 mil dados para cada quantidade de interesse (QoI). Cada emulador é demarcado quatro vezes, representando os diferentes tamanhos amostrais.

Analisando a Figura 53, que apresenta os resultados do Modelo B, observa-se, em todas as quantidades de interesse (QoI), que há um aglomerado de pontos azuis, correspondentes às NNs, com valores baixos de tempo de inferência. Nas QoIs APD50 e APD90, as NNs também apresentam valores de baixo a moderados de MSE. Em seguida, destacam-se os emuladores GP_SK_P e GP_SK_M, que apresentam valores baixos de

MSE, embora com tempos de inferência mais elevados. Por outro lado, o GP_SK_G se destaca negativamente, apresentando baixa acurácia e valores de MSE mais altos. Em relação aos emuladores GPs e PCs, ambos apresentam tempos de inferência e valores de MSE elevados, com seus pontos no gráfico próximos entre si. A exceção é o PC_5, que está mais distante dos outros, apresentando valores de MSE semelhantes, mas com tempos de inferência mais elevados. Esse desempenho faz com que os emuladores GPs e PCs não sejam boas opções para o Modelo B.

Nas QoIs dVmax e Vreps, nota-se um comportamento semelhante às demais variáveis, porém as NNs se aglutinam mais em torno de um MSE moderado, evidenciando ainda mais os emuladores GP_SK_P, GP_SK_M e alguns tamanhos amostrais de PC_3 e PC_5, por apresentarem valores baixos de erro. Os demais PCs e todos os GPs apresentam tempos de inferência e valores de MSE elevados, destacando negativamente o desempenho desses emuladores.

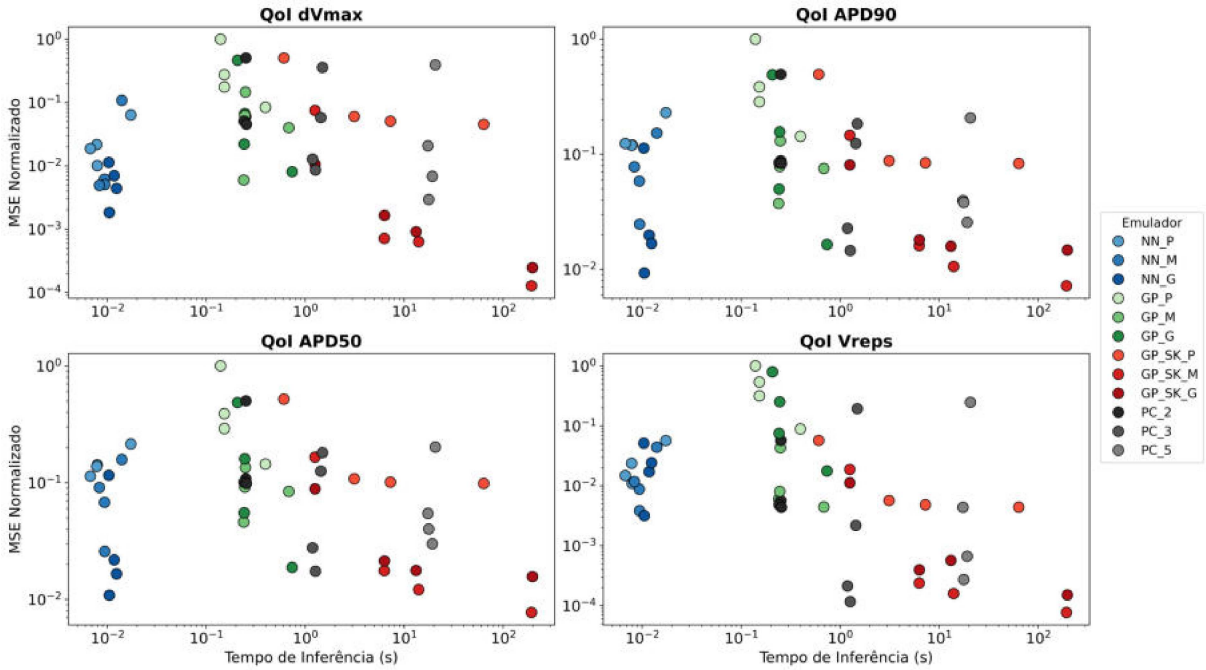

Figura 53 – Gráfico de dispersão evidenciando a correlação entre o MSE (Normalizado) e o tempo necessário, em segundos, para cada emulador do Modelo B inferir 100 mil dados para cada quantidade de interesse (QoI). Cada emulador é demarcado quatro vezes, representando os diferentes tamanhos amostrais.

5.5.6 Análise do desempenho pelo score de Pareto

Como mostrado nos gráficos anteriores, que apresentam as métricas, tempo de treino, tempo de inferência e MSE, de diversos emuladores, para cada uma das quantidades de interesse (QoI), é possível observar que há momentos em que um se destaca em comparação com os outros. Nesse contexto, é essencial definir o desempenho de cada

emulador por meio de uma pontuação, com o intuito de encontrar o melhor emulador para cada modelo.

Diante desse cenário, analisando a Figura 54, que apresenta as pontuações dos emuladores para as QoIs do Modelo A, observa-se que para a quantidade dVmax os PC_2 obtiveram os melhores resultados, mas destaca-se também o GP_SK_G com tamanho amostral 5K, que foi o único Processo Gaussiano presente em todas as QoIs. Para as variáveis APD50 e APD90 o melhor resultado foi obtido pela NN_M com tamanho de amostra de treino de 1K, as demais posições ficaram variando entre NNs e PCs. Importante ressaltar que na QoI APD90 os dois primeiros emuladores não obtiveram diferença significativa entre eles, então, também deve-se destacar o PC_3 com tamanho amostral de 1K. Por fim, para a variável Vreps, os PCs se destacaram ocupando as primeiras 7 posições, sendo que o PC_3 com tamanho amostral de 1K e 0.5K se sobressaíram, já que não houve diferença significativa entre eles.

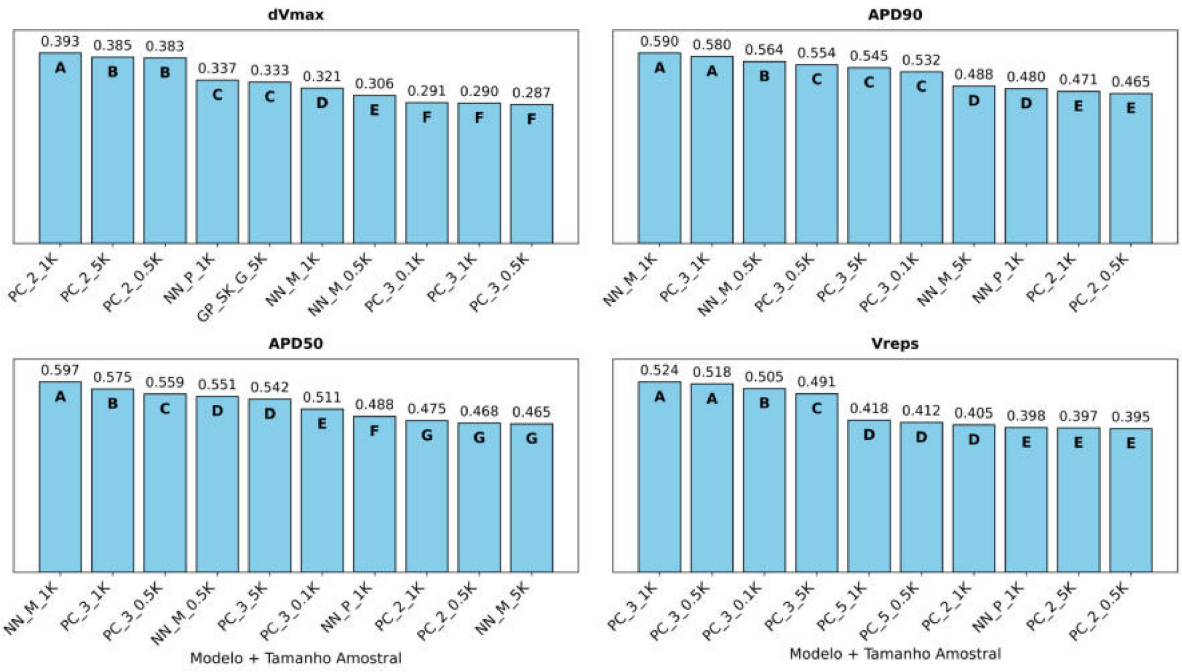

Figura 54 – Gráfico de barras mostrando os 10 melhores emuladores do Modelo A de acordo com o score de pareto. As colunas apresentam letras que, se diferentes, indicam que as notas são significativamente diferentes entre si.

Observando a Figura 55, que apresenta as pontuações dos emuladores para as QoIs do Modelo B, nota-se que para a variável dVmax e Vreps houve um empate estatístico entre as três primeiras posições, onde a NN_P com tamanho amostral de 5K se sobressaiu diante dos GP_SK_P com tamanho de treino de 0.1K e GP_SK_M com 5K. As demais posições da QoI dVmax foram ocupadas por NNs e para Vreps houve uma maior diversidade onde houve NNs, PC_3 e GP_SK_G. Para as variáveis APD50 e APD90, o melhor emulador foi

a NN_G com tamanho de amostra de treino de 5K, obtendo uma nota bem mais elevada que os demais emuladores. O restante das posições seguiu-se muito similar, com NN_P com tamanho amostral de 5K em segundo, seguido do GP_SK_P com 0.1K dados de treino.

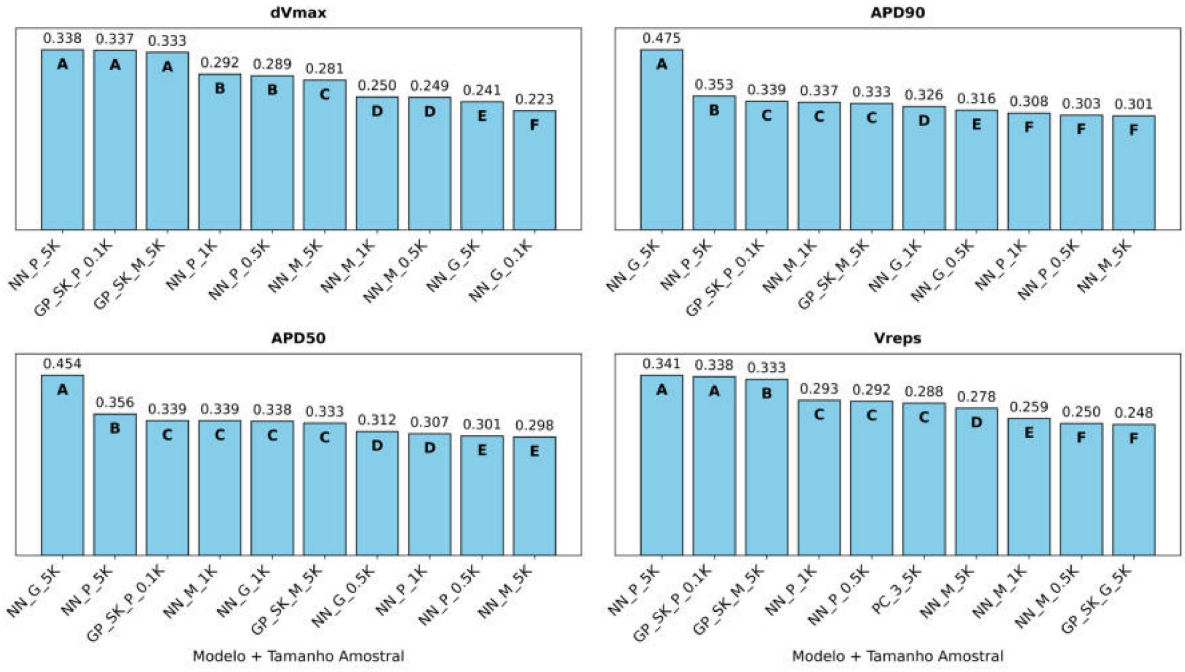

Figura 55 – Gráfico de barras mostrando os 10 melhores emuladores do Modelo B de acordo com o score de pareto. As colunas apresentam letras que, se diferentes, indicam que as notas são significativamente diferentes entre si.

Nota-se que os melhores emuladores do Modelo B, Figura 55, possuem tamanho da amostra de treino mais elevados do que os melhores do Modelo A, Figura 54, indicando que o aumento na dimensão dos dados de entrada deixou a emulação mais complexa, necessitando assim de mais dados para obter o melhor resultado. Ressalta-se que, o score de Pareto pondera acurácia e eficiência de treino e inferência, então mesmo que um emulador se destaque muito em um quesito, ele precisa se sair bem nos demais para ter uma boa nota.

6 CONCLUSÃO

O trabalho teve como objetivo comparar o desempenho de diferentes emuladores para dois modelos celulares da eletrofisiologia cardíaca, verificando como os Processos Gaussianos se saem nesta tarefa em comparação às Redes Neurais e às Expansões de Caos Polinomial.

Após a análise de quatro métricas de acurácia e eficácia para cada um dos 13 kernels estudados, verificou-se que, em ambos os modelos, os três melhores kernels foram Poly, Poly+RBF e Poly+Matern32. Esses kernels foram, portanto, selecionados para comparar os Processos Gaussianos com os demais emuladores, representando, respectivamente, os emuladores de baixa, média e alta complexidade. Observou-se que o kernel Poly+RBF obteve o melhor score de Pareto no Modelo A, enquanto no Modelo B foi o kernel Poly.

Comparando os diferentes emuladores, verificou-se que, para ambos os modelos, o tempo de treino das PCEs, sobretudo de grau 2, é o mais baixo. Então, se o critério de escolha fosse somente o tempo necessário para realizar o treinamento dos emuladores, o PCE de grau 2 é o melhor.

Analisando o tempo de inferência dos emuladores, os melhores tempos ficam para as NNs, não havendo grande diferença entre as diferentes complexidades e tamanhos amostrais desses emuladores. Logo, considerando apenas o tempo necessário para inferir novos dados, as NNs são as melhores opções.

Contrapondo os tempos de treino e inferência dos Processos Gaussianos, concluímos que há um overhead que deve ser considerado em tamanhos amostrais pequenos, onde a utilização de CPU se mostra mais eficaz do que GPU, mas, à medida que mais dados são utilizados, o tempo de inicialização é compensado.

Para o valor de MSE das variáveis dVmax e Vreps se destacaram os emuladores por Processos Gaussianos, utilizando a biblioteca scikit-learn, sendo que o de alta complexidade foi o melhor para o Modelo A e o de média complexidade o melhor para o Modelo B. Para as variáveis APD50 e APD90 os emuladores mais acurados para o Modelo A foram o PCE de grau 5, enquanto para o Modelo B foi o GP de média complexidade, utilizando scikit-learn.

Comparando o ranking final dos dois modelos, conclui-se que o Modelo A possui, como melhores emuladores, considerando o score de Pareto, as NNs e os PCEs, em que os tamanhos amostrais menores se sobressaíram aos maiores. Para as variáveis dVmax e Vreps, a melhor escolha, respectivamente, são os PCEs de grau 2 e grau 3. Para a variável APD50, o emulador que se destaca é a NN de média complexidade, e, para a APD90, há um empate estatístico entre a NN de média complexidade e o PCE de grau 3.

Obtém-se que, para o Modelo B, houve uma grande melhora nos GPs utilizando a biblioteca scikit-learn, em que esse emulador, com as complexidades baixa e média, dividiu

o pódio com a NN de baixa complexidade para as variáveis dVmax e Vreps, evidenciando que o GP de baixa complexidade obteve esse resultado com o menor tamanho amostral de treino. Para as QoIs APD50 e APD90, a NN de alta complexidade é a melhor opção ao se considerar o score de Pareto.

Comparando o score de Pareto dos dois modelos, verifica-se que o Modelo B apresenta maior complexidade devido ao aumento da dimensionalidade dos dados de entrada. Nesse caso, entre os emuladores de melhor desempenho já se encontram aqueles treinados com 5000 amostras, indicando que esse modelo demanda arquiteturas mais robustas e conjuntos de dados mais extensos para alcançar níveis de acurácia superiores em relação ao Modelo A.

As considerações finais deste trabalho indicam que, embora os resultados obtidos forneçam informações valiosas sobre o desempenho dos emuladores em modelos cardíacos, algumas limitações devem ser reconhecidas. Primeiramente, mão foram avaliados diferentes otimizadores nos Processos Gaussianos, o que poderia influenciar significativamente o desempenho desses emuladores. Adicionalmente, o estudo foi restrito a um modelo eletrofisiológico específico, limitando a generalização dos resultados para outros tipos de modelos cardíacos. Além disso, não foram realizados testes com outras configurações de NNs e PCEs, semelhante ao que foi feito para a escolha dos kernels dos GPs.

Considerando essas limitações, trabalhos futuros podem ser direcionados para expandir as configurações de cada emulador, experimentando novas combinações de camadas e neurônios para redes neurais, além de testar novos kernels e otimizadores para os Processos Gaussianos e variações nos graus de PCE. Esses passos podem levar a novas conclusões sobre o desempenho dos emuladores quando aplicados a dados provenientes de outros modelos cardíacos, possibilitando uma compreensão mais ampla de sua eficácia e limitações em cenários distintos.

REFERÊNCIAS

ALIZADEH, R.; ALLEN, J. K.; MISTREE, F. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, v. 31, n. 3, p. 275–298, 2020. ISSN 1435-6066. Disponível em: <https://doi.org/10.1007/s00163-020-00336-7>.

BERG, L. A. *Generation and uncertainty quantification of patient-specific purkinje network models*. Tese (Tese de Doutorado) — Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, Brasil, agosto 2022. Programa de Pós-graduação em Modelagem Computacional.

BIFULCO, S. F.; AKOUM, N.; BOYLE, P. M. Translational applications of computational modelling for patients with cardiac arrhythmias. *Heart*, BMJ Publishing Group Ltd, v. 107, n. 6, p. 456–461, 2021. ISSN 1355-6037. Disponível em: <https://heart.bmj.com/content/107/6/456>.

CAMPOS, J. et al. Polynomial chaos expansion surrogate modeling of passive cardiac mechanics using the holzapfel–ogden constitutive model. *Journal of Computational Science*, v. 71, p. 102039, 2023. ISSN 1877-7503. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877750323000996>.

CAMPOS, J. O. et al. Uncertainty quantification and sensitivity analysis of left ventricular function during the full cardiac cycle. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 378, n. 2173, p. 20190381, 2020. Disponível em: <https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2019.0381>.

CHANG, E. T. Y.; STRONG, M.; CLAYTON, R. H. Bayesian sensitivity analysis of a cardiac cell model using a gaussian process emulator. *PLOS ONE*, Public Library of Science, v. 10, n. 6, p. 1–20, 06 2015. Disponível em: <https://doi.org/10.1371/journal.pone.0130252>.

COVENEY, S. et al. Bayesian calibration of electrophysiology models using restitution curve emulators. *Frontiers in Physiology*, v. 12, 2021. ISSN 1664-042X. Disponível em: <https://www.frontiersin.org/journals/physiology/articles/10.3389/fphys.2021.693015>.

DOMINGUEZ-GOMEZ, P. et al. Real-time prediction of drug-induced proarrhythmic risk with sex-specific cardiac emulators. *bioRxiv*, Cold Spring Harbor Laboratory, 2024. Disponível em: <https://www.biorxiv.org/content/early/2024/10/02/2024.09.30.615798>.

DUNN, O. J. Multiple comparisons using rank sums. *Technometrics*, Taylor & Francis, v. 6, n. 3, p. 241–252, 1964.

FEINBERG, J.; LANGTANGEN, H. P. Chaospy: An open source tool for designing methods of uncertainty quantification. *Journal of Computational Science*, Elsevier, v. 11, p. 46–57, 2015.

FOSTER, L. et al. Stable and efficient gaussian process calculations. *Journal of Machine Learning Research*, v. 10, n. 4, 2009.

FRAZIER, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

GARDNER, J. et al. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in Neural Information Processing Systems*, v. 31, 2018.

GIBBONS, J. D.; CHAKRABORTI, S. *Nonparametric statistical inference: revised and expanded*. [S.l.]: CRC press, 2014.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2001.

HODGKIN, A. L.; HUXLEY, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, v. 117, n. 4, p. 500, 1952.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080.

KINGMA, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KLÉBER, A. G.; RUDY, Y. Basic mechanisms of cardiac impulse propagation and associated arrhythmias. *Physiological Reviews*, v. 84, n. 2, p. 431–488, 2004. PMID: 15044680. Disponível em: <https://doi.org/10.1152/physrev.00025.2003>.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, Taylor & Francis, v. 47, n. 260, p. 583–621, 1952.

LAWSON, B. A. J. et al. Variability in electrophysiological properties and conducting obstacles controls re-entry risk in heterogeneous ischaemic tissue. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Royal Society Publishing, v. 378, n. 2173, p. 20190341, May 2020.

MARGARA, F. et al. In-silico human electro-mechanical ventricular modelling and simulation for drug-induced pro-arrhythmia and inotropic risk assessment. *Progress in Biophysics and Molecular Biology*, v. 159, p. 58–74, 2021. ISSN 0079-6107. Mechanobiology of the Cardiovascular System. Disponível em: <https://www.sciencedirect.com/science/article/pii/S007961072030064X>.

Merriam-Webster Dictionary. *Simulation*. 2024. <https://www.merriam-webster.com/dictionary/simulation>. Acesso em: 16 jul. 2025.

MONTGOMERY, D. C. *Design and analysis of experiments*. [S.l.]: John wiley & sons, 2017.

NEAL, R. M. Bayesian learning for neural networks. *Lecture Notes in Statistics*, Springer Science & Business Media, v. 118, 1996.

NIEDERER, S.; LUMENS, J.; TRAYANOVA, N. A. Computational models in cardiology. *Nature Reviews Cardiology*, v. 16, n. 2, p. 100–111, 2019.

OLIVEIRA, R. S. et al. Ectopic beats arise from micro-reentries near infarct regions in simulations of a patient-specific heart model. *Scientific Reports*, v. 8, n. 1, p. 16392, 2018. ISSN 2045-2322. Disponível em: <https://doi.org/10.1038/s41598-018-34304-y>.

O'HAGAN, A. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering System Safety*, v. 91, n. 10, p. 1290–1300, 2006. ISSN 0951-8320. The Fourth International Conference on Sensitivity Analysis of Model Output (SAMO 2004). Disponível em: <https://www.sciencedirect.com/science/article/pii/S0951832005002383>.

PAGANI, S.; MANZONI, A. Enabling forward uncertainty quantification and sensitivity analysis in cardiac electrophysiology by reduced order modeling and machine learning. *International Journal for Numerical Methods in Biomedical Engineering*, v. 37, n. 6, p. e3450, 2021.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PIGOZZO, R. B.; SANTOS, R. W. dos; ROCHA, B. M. Sensitivity analysis of cardiac alternans in electrophysiology cell models: A monte carlo filtering approach. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, v. 35, n. 4, p. 043111, 04 2025. ISSN 1054-1500. Disponível em: <https://doi.org/10.1063/5.0258095>.

PYTORCH TEAM. *CUDA semantics — PyTorch 2.7 documentation*. 2024. Disponível em: <https://pytorch.org/docs/stable/notes/cuda.html>.

RASMUSSEN, C. E.; WILLIAMS, C. K. *Gaussian processes for machine learning*. Cambridge, MA: The MIT Press, 2006.

REGAZZONI, F. et al. A machine learning method for real-time numerical simulations of cardiac electromechanics. *Computer Methods in Applied Mechanics and Engineering*, v. 393, p. 114825, 2022. ISSN 0045-7825. Disponível em: <https://www.sciencedirect.com/science/article/pii/S004578252200144X>.

RUPP, L. C. et al. Using uncertainsci to quantify uncertainty in cardiac simulations. In: IEEE. *2020 Computing in Cardiology*. [S.l.], 2020. p. 1–4.

TRAYANOVA, N. A. Whole-heart modeling: Applications to cardiac electrophysiology and electromechanics. *Circulation Research*, v. 108, n. 1, p. 113–128, 2011.

TUSSCHER, K. H. T. et al. A model for human ventricular tissue. *American Journal of Physiology-Heart and Circulatory Physiology*, American Physiological Society, v. 286, n. 4, p. H1573–H1589, 2004.

WILLIAMS, C.; RASMUSSEN, C. Gaussian processes for regression. In: TOURETZKY, D.; MOZER, M.; HASSELMO, M. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1995. v. 8. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/1995/file/7cce53cf90577442771720a370c3c723-Paper.pdf>.

World Health Organization. *The top 10 causes of death*. 2023. <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death>. Accessed: 30 June 2025.

XIU, D.; KARNIADAKIS, G. E. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, v. 24, n. 2, p. 619–644, 2002.

YANG, C. et al. Computational modeling of cardiac electrophysiology with human realistic heart–torso model. *Bioengineering*, v. 12, n. 4, 2025. ISSN 2306-5354. Disponível em: <https://www.mdpi.com/2306-5354/12/4/392>.