Relatorio Técnico de Pesquisa
Modelagem do Problema dos Três Corpos Usando Redes Neurais Recorrentes

Mateus Mattiuzzi Ferreira[1], Adrian Luna[2], Carlos Carballo[3]
[1] Departamento de Física, UFMG
[2] Departamento de Estatística, UFMG
[3] Departamento de Matemática, UFMG

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho investiga a aplicação de Redes Neurais Recorrentes LSTM na previsão da dinâmica do problema dos três corpos em configurações Eulerianas instáveis. Desenvolvemos um *pipeline* completo que integra simulação numérica de alta precisão com RK45, pré-processamento diferenciado por componentes físicos, arquiteturas LSTM otimizadas e metodologia de previsão autoregressiva.

Nossos resultados demonstram que as redes neurais são capazes de aprender representações qualitativas da dinâmica orbital a partir de uma única trajetória, capturando padrões temporais complexos em configurações instáveis. No entanto, a precisão quantitativa em longo prazo é fundamentalmente limitada pela natureza caótica do sistema, com erros de previsão crescendo exponencialmente após 4.6×10^7 segundos.

A análise teórica à luz do Teorema da Aproximação Universal revela que as limitações observadas estão alinhadas com as restrições fundamentais da aproximação neural em sistemas dinâmicos complexos. Em contraste com abordagens em configurações Lagrangianas estáveis reportadas na literatura, nossa investigação em configurações Eulerianas instáveis destaca desafios adicionais impostos pela sensibilidade às condições iniciais.

Este trabalho contribui para o diálogo entre a mecânica celeste tradicional e técnicas modernas de aprendizado de máquina, fornecendo insights sobre os limites práticos das redes neurais na modelagem de sistemas dinâmicos caóticos e propondo direções futuras para abordagens híbridas que incorporem conhecimento físico explícito.

Palavras-chave: Problema dos Três Corpos, Redes Neurais Recorrentes, LSTM, Sistemas Caóticos, Mecânica Celeste, Aprendizado de Máquina, Dinâmica Orbital.

Capítulo 1

Introdução

O problema dos três corpos, formulado originalmente no contexto da mecânica newtoniana, permanece como um dos problemas mais desafiadores da física matemática Poincaré (1890). Diferentemente do problema de dois corpos, que admite solução analítica completa, o sistema de três corpos exibe comportamento caótico para a maioria das condições iniciais, tornando impossível a obtenção de soluções gerais em forma fechada. Esta complexidade inerente tem motivado abordagens alternativas que complementem os métodos numéricos tradicionais.

Tradicionalmente, a investigação desses sistemas tem dependido de métodos numéricos sofisticados. Integradores como os métodos de Runge-Kutta e Hermite têm sido a ferramenta padrão Hairer, Norsett e Wanner (1993), porém com custo computacional significativo para simulações de longo prazo. Recentemente, abordagens baseadas em aprendizado de máquina emergiram como alternativas promissoras, Breen et al. (2020), oferecendo potencial para aceleração computacional significativa.

Breen et al. (2020) demonstraram que redes neurais profundas podem acelerar as previsões em até cinco ordens de magnitude comparado com métodos numéricos tradicionais. No entanto, a aplicação dessas técnicas a sistemas caóticos apresenta desafios únicos relacionados à estabilidade numérica e generalização. Este trabalho se insere neste contexto de inovação, investigando a aplicação de Redes Neurais Recorrentes (RNRs) — arquitetura projetada para capturar dependências temporais — na modelagem do problema dos três corpos em configurações Eulerianas instáveis.

Nossa abordagem concentra-se no aprendizado profundo de uma única órbita periódica instável seguindo a configuração Euleriana colinear, permitindo investigar a capacidade das LSTMs em capturar padrões temporais complexos em sistemas dinâmicos específicos sob condições desfavoráveis. O objetivo não é apenas desenvolver um preditor, mas também investigar os desafios e limitações inerentes a essa abordagem, incluindo questões fundamentais sobre generalização, conservação de quantidades físicas e escalabilidade

em configurações instáveis.

Este artigo está organizado da seguinte forma: o **Capítulo 2** apresenta a formulação matemática completa do problema dos três corpos, incluindo as equações fundamentais do movimento, quantidades conservadas e a configuração Euleriana colinear adotada. O **Capítulo 3** detalha a metodologia de simulação e geração de dados, com ênfase na escolha do método numérico RK45 e no *pipeline* de pré-processamento inteligente. O **Capítulo 4** introduz os fundamentos teóricos das redes neurais recorrentes, com foco especial nas arquiteturas LSTM e GRU. O **Capítulo 5** descreve a implementação prática dos experimentos, incluindo a arquitetura do modelo, estratégias de treinamento e metodologia de previsão autoregressiva. O **Capítulo 6** apresenta e discute os resultados obtidos, com análise quantitativa do erro preditivo e comparações entre configurações Eulerianas e Lagrangianas. O **Capítulo 7** desenvolve a discussão teórica à luz do Teorema da Aproximação Universal, analisando as limitações fundamentais da abordagem. Finalmente, o **Capítulo 8** sintetiza as conclusões do trabalho e propõe direções para pesquisas futuras.

Através deste artigo, buscamos fornecer uma análise compreensiva do potencial e das limitações das redes neurais recorrentes na modelagem de sistemas dinâmicos complexos, contribuindo para o diálogo entre a mecânica celeste tradicional e as modernas técnicas de aprendizado de máquina.

Capítulo 2

Modelo do Problema dos Três Corpos

2.1 Formulação Matemática

Considere três corpos pontuais com massas m_1, m_2, m_3 e vetores posição $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3 \in \mathbb{R}^3$. As equações do movimento são derivadas da mecânica newtoniana clássica, combinando a Segunda Lei do Movimento com a Lei da Gravitação Universal, ver capitulo 2 de Saslaw (2006) para uma descriçã completa das equações, elas são:

$$F_i = m_i \frac{d^2\vec{r}_i}{dt^2} = \sum_{j \neq i} \frac{G m_i m_j}{r_{ij}^3}(\vec{r}_j - \vec{r}_i), \quad i = 1, 2, 3 \tag{2.1}$$

onde $r_{ij} = \|\vec{r}_j - \vec{r}_i\|$ e $G = 6.67430 \times 10^{-11}\,\mathrm{m^3 kg^{-1} s^{-2}}$ é a constante gravitacional. Apos simplificar m_i:

$$\frac{d^2\vec{r}_i}{dt^2} = \sum_{j \neq i} \frac{G m_j}{r_{ij}^3}(\vec{r}_j - \vec{r}_i), \quad i = 1, 2, 3 \tag{2.2}$$

Energia e Quantidades Conservadas

A energia potencial do sistema é dada por:

$$U = -G\left(\frac{m_1 m_2}{r_{12}} + \frac{m_1 m_3}{r_{13}} + \frac{m_2 m_3}{r_{23}}\right) \tag{2.3}$$

A energia cinética é:

$$T = \frac{1}{2}\sum_{i=1}^{3} m_i \|\vec{v}_i\|^2 \tag{2.4}$$

A energia total $E = T + U$ é conservada, assim como o momento linear total:

$$\vec{P} = \sum_{i=1}^{3} m_i \vec{v}_i \tag{2.5}$$

e o momento angular total:

$$\vec{L} = \sum_{i=1}^{3} m_i \vec{r}_i \times \vec{v}_i \tag{2.6}$$

2.2 Configuração de Referência Periódica e Considerações de Estabilidade

Para este estudo, adotamos uma configuração específica que gera órbitas periódicas instáveis, seguindo as soluções colineares de Euler (1767). Esta configuração contrasta com as órbitas lagrangianas estáveis estudadas em trabalhos como Hairer, Norsett e Wanner (1993), que focam em configurações triangulares equiláteras estáveis.

Nossa configuração Euleriana colinear é definida por:

$$\begin{aligned}
\vec{r}_1 &= (0, 0, 0) \\
\vec{r}_2 &= (1.0 \times 10^{11}, 0, 0) \\
\vec{r}_3 &= (-1.0 \times 10^{11}, 0, 0) \\
\vec{v}_1 &= (0, 0, 0) \\
\vec{v}_2 &= (0, 1.0 \times 10^{4}, 0) \\
\vec{v}_3 &= (0, -1.0 \times 10^{4}, 0)
\end{aligned} \tag{2.7}$$

com massas $m_1 = m_2 = m_3 = 1.0 \times 10^{30}$ kg. Esta configuração representa uma solução colinear do problema dos três corpos, onde os três corpos estão permanentemente alinhados ao longo de um eixo.

Diferentemente das configurações lagrangianas triangulares estáveis Lagrange (1772), as soluções Eulerianas colineares são intrinsecamente instáveis Marchal (1990). Esta instabilidade fundamental torna o problema particularmente desafiador para redes neurais, pois pequenos erros de previsão são rapidamente amplificados devido à natureza instável da dinâmica.

A escolha de uma configuração Euleriana instável, em vez de uma configuração Lagrangiana estável, foi intencional para testar os limites das redes neurais em capturar dinâmicas complexas e sensíveis. Enquanto Hairer, Norsett e Wanner (1993) focam em órbitas estáveis que são mais adequadas para métodos numéricos tradicionais, nossa abordagem investiga a capacidade das LSTMs em lidar com configurações onde métodos convencionais enfrentam dificuldades devido à instabilidade inerente.

Capítulo 3

Metodologia de Simulação e Geração de Dados

3.1 Abordagem Focada em Órbita Única

Diferente de abordagens tradicionais que utilizam grandes conjuntos de trajetórias diversas, nossa metodologia concentra-se no aprendizado profundo de uma única órbita periódica de alta qualidade. Esta estratégia permite investigar a capacidade das redes neurais em capturar padrões temporais complexos sem as variáveis adicionais introduzidas por diferentes condições iniciais.

3.2 Métodos Numéricos e Justificativa da Escolha do RK45

O **AMUSE (Astrophysical Multipurpose Software Environment)** é um *framework* integrado que oferece diversos métodos numéricos especializados para problemas astrofísicos. Entre os integradores disponíveis, destacam-se os métodos Hermite de 4ª e 6ª ordens, o método Leapfrog (Kick-Drift-Kick) simplético, métodos de Runge-Kutta de diversas ordens, e o integrador BRUTUS de precisão arbitrária para sistemas caóticos.

Inicialmente, buscou-se utilizar o integrador BRUTUS, desenvolvido especificamente para lidar com a natureza caótica do problema N-corpos através de aritmética de precisão arbitrária e controle rigoroso de erro. No entanto, enfrentamos dificuldades significativas relacionadas à desatualização da interface, complexidade de configuração dos parâmetros de precisão arbitrária, custo computacional proibitivo e instabilidade numérica em alguns casos.

Diante desses desafios, optamos pelo método Runge-Kutta-Fehlberg de ordem 4(5) (RK45) devido à sua robustez comprovada em problemas de

mecânica celeste, eficiência computacional para geração de *datasets*, controle de erro adaptativo com tolerância relativa de 10^{-6} e implementação estável e madura no SciPy. Esta decisão alinha-se com as observações de Breen et al. (2020) sobre a viabilidade prática: embora métodos como BRUTUS ofereçam precisão arbitrária, seu custo computacional é proibitivo para geração de grandes volumes de dados de treinamento.

Pipeline de Geração de Dados

A continuação apresentamos o algoritmo que desenvolvemos para gerar os dados.

Algorithm 1 Geração de Dataset a partir de Órbita de Referência Periódica

procedure GenerateReferenceDataset
  Entrada: $T = 5.049 \times 10^7$ s (1.6 anos), $n_{\text{steps}} = 1000$
  Saída: train_data, val_data, scalers
  // 1. Configuração inicial simétrica periódica
  $\text{state}_0 \leftarrow [\vec{r}_1, \vec{r}_2, \vec{r}_3, \vec{v}_1, \vec{v}_2, \vec{v}_3]$
  // 2. Simulação numérica com RK45
  trajectory $\leftarrow$ solve_ivp(three_body_equations, $\text{state}_0, T, n_{\text{steps}}$)
  // 3. Separação inteligente por componentes físicos
  pos1 $\leftarrow$ trajectory$[:, :3]$ ▷ Posições Corpo 1
  pos23 $\leftarrow$ trajectory$[:, 3:9]$ ▷ Posições Corpos 2 e 3
  vel $\leftarrow$ trajectory$[:, 9:]$ ▷ Velocidades todos corpos
  // 4. Normalização independente por tipo físico
  scaler_P1 $\leftarrow$ StandardScaler()
  scaler_P23 $\leftarrow$ StandardScaler()
  scaler_V $\leftarrow$ StandardScaler()
  scaled_pos1 $\leftarrow$ scaler_P1.fit_transform(pos1)
  scaled_pos23 $\leftarrow$ scaler_P23.fit_transform(pos23)
  scaled_vel $\leftarrow$ scaler_V.fit_transform(vel)
  // 5. Recombinação dos componentes normalizados
  scaled_data $\leftarrow$ concat(scaled_pos1, scaled_pos23, scaled_vel)
  // 6. Divisão temporal sequencial (80%/20%)
  $n_{\text{train}} \leftarrow \lfloor 0.8 \times n_{\text{steps}} \rfloor$
  train_data $\leftarrow$ scaled_data$[: n_{\text{train}}]$
  val_data $\leftarrow$ scaled_data$[n_{\text{train}} :]$
  return train_data, val_data, scaler_P1, scaler_P23, scaler_V
end procedure

3.3 Implementação da Simulação Numérica

A simulação da órbita de referência foi implementada utilizando o método Runge-Kutta-Fehlberg de ordem 4(5) (RK45) com alta precisão. Este método adaptativo de passo variável combina uma fórmula de 4ª ordem para avançar a solução com uma fórmula de 5ª ordem para estimar o erro local de truncamento.

A implementação através da função `scipy.integrate.solve_ivp` utiliza os parâmetros mostrados no Listing 3.1:

```
solution_ref = solve_ivp(
    three_body_equations,
    t_span_ref,
    initial_state_ref,
    t_eval=t_eval_ref,
    method='RK45',
    rtol=1e-6  % Tolerância relativa
)
```

Listing 3.1: Implementação do RK45

O método é definido pelas seguintes etapas para cada passo temporal:

$$k_1 = hf(t_n, y_n) \tag{3.1}$$
$$k_2 = hf(t_n + \tfrac{h}{4}, y_n + \tfrac{k_1}{4}) \tag{3.2}$$
$$k_3 = hf(t_n + \tfrac{3h}{8}, y_n + \tfrac{3k_1}{32} + \tfrac{9k_2}{32}) \tag{3.3}$$
$$k_4 = hf(t_n + \tfrac{12h}{13}, y_n + \tfrac{1932k_1}{2197} - \tfrac{7200k_2}{2197} + \tfrac{7296k_3}{2197}) \tag{3.4}$$
$$k_5 = hf(t_n + h, y_n + \tfrac{439k_1}{216} - 8k_2 + \tfrac{3680k_3}{513} - \tfrac{845k_4}{4104}) \tag{3.5}$$
$$k_6 = hf(t_n + \tfrac{h}{2}, y_n - \tfrac{8k_1}{27} + 2k_2 - \tfrac{3544k_3}{2565} + \tfrac{1859k_4}{4104} - \tfrac{11k_5}{40}) \tag{3.6}$$

A solução de 4ª ordem é:

$$y_{n+1}^{(4)} = y_n + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \tag{3.7}$$

E a solução de 5ª ordem para estimativa de erro:

$$y_{n+1}^{(5)} = y_n + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \tag{3.8}$$

O erro estimado $\delta = |y_{n+1}^{(5)} - y_{n+1}^{(4)}|$ é usado para ajustar adaptativamente o tamanho do passo h, garantindo que a tolerância relativa de 10^{-6} seja mantida em toda a simulação.

As equações do movimento do problema dos três corpos foram implementadas conforme mostrado no Listing 3.2:

```python
def three_body_equations(t, state):
    n = 3
    positions = state[:3*n].reshape(n, 3)
    velocities = state[3*n:].reshape(n, 3)
    derivatives = np.zeros_like(state)
    accelerations = np.zeros((n, 3))

    for i in range(n):
        for j in range(n):
            if i != j:
                r_vec = positions[j] - positions[i]
                r = np.linalg.norm(r_vec)
                accelerations[i] += G * m[j] * r_vec / r
                    **3

    derivatives[:3*n] = velocities.flatten()
    derivatives[3*n:] = accelerations.flatten()
    return derivatives
```

Listing 3.2: Implementação das Equações do Movimento

3.4 Pré-processamento Inteligente

A estratégia de normalização diferenciada reconhece que diferentes componentes físicos possuem distribuições estatísticas distintas, conforme implementado no Listing 3.3:

```python
# Separação por componentes físicos
P1_data = full_data[:, :3]    # Posições Corpo 1 (x1, y1,
    z1)
P23_data = full_data[:, 3:9]  # Posições Corpos 2 e 3 (x2
    , y2, z2, x3, y3, z3)
V_data = full_data[:, 9:]     # Velocidades (vx1, vy1,
    vz1, ..., vz3)

# Normalização independente por tipo físico
scaler_P1 = StandardScaler()
scaler_P23 = StandardScaler()
scaler_V = StandardScaler()
scaled_P1 = scaler_P1.fit_transform(P1_data)
scaled_P23 = scaler_P23.fit_transform(P23_data)
scaled_V = scaler_V.fit_transform(V_data)
# Recombinação
scaled_data = np.hstack([scaled_P1, scaled_P23, scaled_V
    ])
```

Listing 3.3: Pré-processamento Diferenciado

3.5 Divisão Temporal dos Dados

Adotamos uma divisão sequencial que preserva a continuidade temporal, utilizando 80% inicial para treinamento (passos temporais 0-799) e 20% final para validação (passos temporais 800-999). Esta abordagem simula cenários reais de previsão onde o modelo deve extrapolar a dinâmica aprendida para o futuro temporal.

Capítulo 4

Redes Neurais Recorrentes

4.1 Introdução às Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados na estrutura e funcionamento do cérebro biológico Keydana et al. (2022). Uma RNA é composta por unidades de processamento elementares chamadas neurônios artificiais, organizados em camadas interconectadas. Cada conexão entre neurônios possui um peso sináptico que é ajustado durante o processo de aprendizado.

Uma rede neural típica consiste em:

- **Camada de entrada**: Recebe os dados brutos
- **Camadas ocultas**: Realizam transformações não-lineares nos dados
- **Camada de saída**: Produz a predição final

A operação fundamental de um neurônio artificial é dada por:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4.1}$$

onde x_i são as entradas, w_i são os pesos, b é o bias e f é uma função de ativação não-linear.

A Figura 4.1 ilustra essa arquitetura.

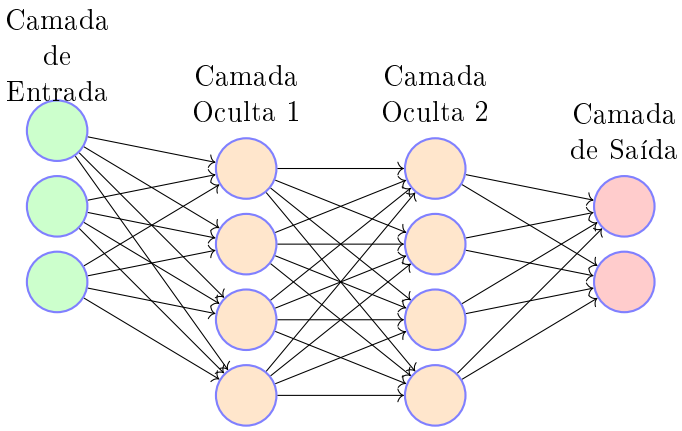

Figura 4.1: Arquitetura de uma rede neural artificial com múltiplas camadas. As setas representam conexões ponderadas entre neurônios adjacentes.

As redes neurais se destacam por sua capacidade de aprendizado de representações hierárquicas dos dados, permitindo a modelagem de relações complexas e não-lineares. Esta característica as torna particularmente adequadas para problemas onde relações tradicionais de entrada-saída são desconhecidas ou difíceis de especificar.

4.2 Arquiteturas Fundamentais

4.2.1 LSTM (Long Short-Term Memory)

As redes neurais recorrentes do tipo Long Short-Term Memory (LSTM), introduzidas por Hochreiter e Schmidhuber (1997), incorporam células de memória e portões para controlar o fluxo de informação, resolvendo efetivamente o problema do *vanishing gradient* em redes recorrentes tradicionais. Esta arquitetura foi projetada para lidar com dependências temporais de longo prazo, inspirada na necessidade de reter informações relevantes ao longo de sequências temporais extensas.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(portão de esquecimento)} \tag{4.2}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(portão de entrada)} \tag{4.3}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4.5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.6}$$

$$h_t = o_t * \tanh(C_t) \tag{4.7}$$

A Figura 4.2 mostra a arquitetura detalhada de uma célula LSTM.

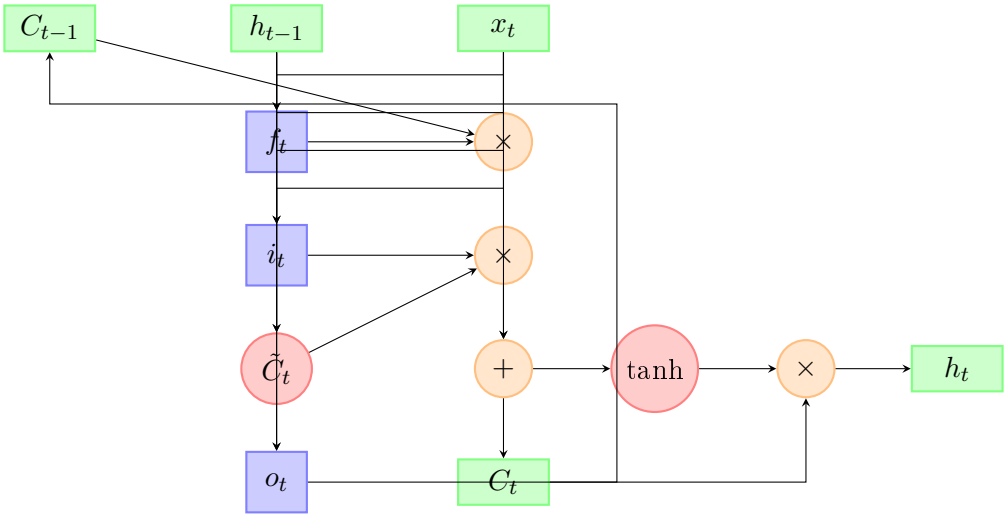

Figura 4.2: Arquitetura detalhada de uma célula LSTM mostrando os portões de esquecimento (f_t), entrada (i_t) e saída (o_t), e o estado da célula (C_t).

4.2.2 GRU (Gated Recurrent Unit)

Arquitetura simplificada com dois portões Cho et al. (2014) que combina o portão de esquecimento e entrada em um único portão de atualização:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad \text{(portão de atualização)} \tag{4.8}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad \text{(portão de reinicialização)} \tag{4.9}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{4.10}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{4.11}$$

A Figura 4.3 ilustra a arquitetura de uma célula GRU.

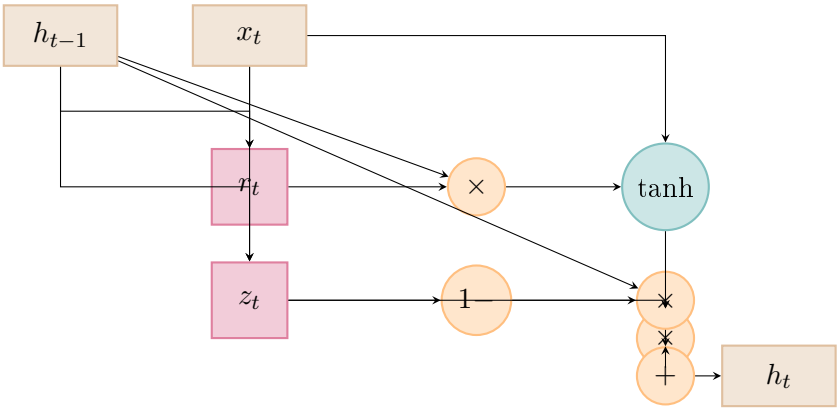

Figura 4.3: Arquitetura de uma célula GRU mostrando os portões de atualização (z_t) e reinicialização (r_t).

4.2.3 Quadro Comparativo: LSTM vs GRU

A Tabela 4.1 compara as duas arquiteturas.

Tabela 4.1: Comparação entre as arquiteturas LSTM e GRU

Característica	**LSTM**	**GRU**
Número de portões	3 portões (esquecimento, entrada, saída)	2 portões (atualização, reinicialização)
Estado interno	Dois estados (h_t e C_t)	Um estado (h_t)
Complexidade computacional	Alta	Moderada
Número de parâmetros	Maior	Menor (aproximadamente 75% dos parâmetros da LSTM)
Desempenho em sequências longas	Excelente	Bom
Tempo de treinamento	Mais lento	Mais rápido
Adequação para datasets pequenos	Requer mais dados para generalização	Pode performar melhor com dados limitados
Controle de fluxo de informação	Controle independente para esquecimento e atualização	Controle acoplado para esquecimento e atualização

4.2.4 Justificativa da Escolha pela Arquitetura LSTM

Para o problema dos três corpos, optamos pela arquitetura LSTM devido às seguintes considerações fundamentais:

- **Natureza de Longo Prazo da Dinâmica Orbital**: O problema dos três corpos envolve dependências temporais que se estendem por escalas de tempo significativas. A presença do estado da célula (C_t) na LSTM proporciona um caminho direto para propagação de informação através de longos intervalos temporais, essencial para capturar a evolução orbital.
- **Controle Fino do Fluxo de Informação**: Os três portões independentes da LSTM (esquecimento, entrada e saída) permitem um controle mais granular sobre qual informação reter, atualizar e expor a cada passo temporal. Esta flexibilidade é crucial para modelar as complexas interações gravitacionais no problema dos três corpos.
- **Robustez em Configurações Instáveis**: Dada a natureza instável da configuração Euleriana colinear adotada, a capacidade da LSTM de manter informações relevantes por longos períodos através de seu estado da célula torna-se particularmente vantajosa para mitigar a propagação de erros.
- **Evidências Empíricas em Problemas Similares**: A literatura especializada Hochreiter e Schmidhuber (1997) e Breen et al. (2020) reporta consistentemente melhor desempenho das LSTMs em problemas de dinâmica orbital e sistemas caóticos comparado com arquiteturas mais simples.
- **Capacidade de Memória de Longo Prazo**: O mecanismo explícito de memória da LSTM é fundamental para aprender e manter as quantidades conservadas (energia, momento angular) que governam a evolução do sistema em escalas temporais extensas.

Embora as GRUs ofereçam vantagens em termos de eficiência computacional, consideramos que a capacidade superior das LSTMs em lidar com dependências de longo prazo e sua arquitetura mais expressiva justificam a escolha para este problema específico, onde a precisão em longos horizontes temporais é crítica.

4.3 Problemas e Técnicas de Regularização

As redes neurais recorrentes enfrentam desafios como o problema de vanishing/exploding gradients, que é mitigado pelas arquiteturas LSTM/GRU e técnicas como gradient clipping. O overfitting é combatido através de dropout, early stopping e regularização L2. Em sistemas caóticos, surgem desafios específicos como sensibilidade extrema às condições iniciais, acúmulo de erro em previsão autoregressiva, dificuldade em aprender invariantes físicos de longo prazo e necessidade de janelas temporais adequadas para capturar a dinâmica.

Capítulo 5

Implementação e Experimentos

5.1 Arquitetura do Modelo LSTM

Implementamos uma arquitetura LSTM profunda otimizada especificamente para a tarefa de previsão de trajetórias no problema dos três corpos. A arquitetura foi cuidadosamente dimensionada para equilibrar capacidade de modelagem e eficiência computacional, considerando a natureza complexa e caótica do sistema dinâmico em estudo, conforme mostrado no Listing 5.1:

```
model = Sequential([
    LSTM(384, return_sequences=True, input_shape=(
        window_size, 18)),
    Dropout(0.3),
    LSTM(256, return_sequences=False),
    Dropout(0.3),
    Dense(256, activation='relu'),
    Dense(18)  # Saída: próximo estado completo (18
        dimensões)
])
```

Listing 5.1: Arquitetura do Modelo LSTM

A arquitetura proposta segue uma abordagem hierárquica para capturar padrões temporais em múltiplas escalas:

- **Camada LSTM de 384 unidades**: Esta primeira camada processa a sequência temporal completa (janela de 20 passos) e retorna a sequência completa para permitir que a próxima camada acesse informações em todos os passos temporais. O número de unidades (384) foi escolhido para proporcionar capacidade suficiente para aprender representações complexas da dinâmica orbital.
- **Camada de Dropout (30%)**: Aplicada após a primeira LSTM para prevenir overfitting, removendo aleatoriamente 30% das conexões du-

rante o treinamento, forçando a rede a aprender representações robustas.

- **Camada LSTM de 256 unidades**: Esta segunda camada consolida a informação temporal da sequência, retornando apenas o estado final (último passo temporal), que encapsula a informação relevante de toda a janela temporal.
- **Camada Dense de 256 unidades**: Realiza transformações não-lineares adicionais no espaço de características, utilizando a função de ativação ReLU (Rectified Linear Unit) para introduzir não-linearidades.
- **Camada de Saída Linear**: Produz as 18 dimensões do próximo estado do sistema (3 corpos $\times$ 6 variáveis de estado por corpo: x, y, z, vx, vy, vz).

A justificativa para esta arquitetura baseia-se na necessidade de capturar tanto dependências de curto prazo (primeira LSTM) quanto consolidar informações de longo prazo (segunda LSTM), enquanto as camadas densas finais mapeiam a representação temporal para o espaço de estados físicos.

5.2 Gerador de Dados com Janelamento Temporal

Desenvolvemos um `DataGenerator` personalizado que implementa eficientemente o janelamento temporal, crucial para o treinamento de redes recorrentes. Este gerador processa os dados sob demanda, minimizando o uso de memória e permitindo o treinamento com grandes conjuntos de dados temporais, conforme implementado no Listing 5.2:

```
class DataGenerator(Sequence):
    def __init__(self, data, window_size, batch_size):
        self.data = data
        self.window_size = window_size
        self.batch_size = batch_size
        self.indices = np.arange(len(data) - window_size)

    def __len__(self):
        return len(self.indices) // self.batch_size

    def __getitem__(self, idx):
        start = idx * self.batch_size
        end = (idx + 1) * self.batch_size
        batch_indices = self.indices[start:end]

        batch_x = []
        batch_y = []
        for i in batch_indices:
```

```
19              # Janela de entrada: 'window_size' passos
                    temporais
20              batch_x.append(self.data[i:i+self.window_size
                    ])
21              # Alvo: próximo passo temporal após a janela
22              batch_y.append(self.data[i+self.window_size])
23
24          return np.array(batch_x), np.array(batch_y)
```

Listing 5.2: Gerador de Dados com Janelamento Temporal

O algoritmo de janelamento temporal opera da seguinte forma:

1. **Inicialização**: O gerador recebe a série temporal completa, o tamanho da janela temporal e o tamanho do lote (batch size).
2. **Geração de Índices**: Cria um array de índices válidos para início de janelas, garantindo que cada janela tenha comprimento completo.
3. **Divisão em Lotes**: Particiona os índices em lotes de tamanho fixo para processamento eficiente durante o treinamento.
4. **Construção de Janelas**: Para cada lote, constrói pares (X, Y) onde:
   - **X**: Sequência de *window_size* passos temporais consecutivos
   - **Y**: Próximo passo temporal imediatamente após a sequência X
5. **Retorno dos Dados**: Retorna os lotes formatados como arrays NumPy prontos para alimentar a rede neural.

Esta abordagem de janelamento deslizante permite que a rede aprenda a transição entre estados consecutivos do sistema, capturando a dinâmica temporal inerente ao problema dos três corpos.

5.3 Metodologia de Treinamento

O processo de treinamento foi cuidadosamente projetado para otimizar o desempenho do modelo enquanto previne overfitting, considerando as particularidades dos sistemas caóticos, conforme detalhado no Listing 5.3:

```
1 # Compilação do modelo
2 model.compile(
3     optimizer=Adam(learning_rate=5e-4),
4     loss='mse',
5     metrics=['mae']
6 )
7
8 # Callbacks para treinamento robusto
```

```
callbacks = [
    EarlyStopping(
        monitor='val_loss',
        patience=15,
        restore_best_weights=True
    ),
    ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.5,
        patience=10,
        min_lr=1e-6
    )
]

# Treinamento do modelo
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=val_generator,
    callbacks=callbacks,
    verbose=1
)
```

Listing 5.3: Configuração do Treinamento

Os componentes principais da metodologia de treinamento incluem:

5.3.1 Otimizador Adam

Utilizamos o otimizador Adam (Adaptive Moment Estimation) com taxa de aprendizado de 5×10^{-4}, que combina as vantagens de dois métodos populares: AdaGrad e RMSProp. O Adam adapta a taxa de aprendizado para cada parâmetro com base nas estimativas de primeiro e segundo momentos dos gradientes.

5.3.2 Função de Perda

Empregamos o Erro Quadrático Médio (MSE) como função de perda:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.1}$$

onde y_i são os valores verdadeiros e $\hat{y}_i$ são as predições do modelo. O MSE é particularmente adequado para problemas de regressão e penaliza fortemente erros grandes, o que é desejável na previsão de trajetórias.

5.3.3 Estratégias de Regularização

Early Stopping

Implementamos early stopping com paciência de 15 épocas, monitorando a loss de validação. Esta técnica interrompe o treinamento quando o desempenho no conjunto de validação para de melhorar, prevenindo overfitting e economizando tempo computacional.

Redução de Taxa de Aprendizado

Um callback de redução de taxa de aprendizado diminui a taxa pela metade quando a loss de validação estagna por 10 épocas, permitindo ajustes mais finos nos parâmetros nas fases finais do treinamento.

Dropout

As camadas de dropout com taxa de 30% ajudam a prevenir co-adaptação de neurônios, forçando a rede a aprender representações mais robustas.

5.4 Metodologia de Previsão Autoregressiva

Para avaliar a capacidade preditiva do modelo em longo prazo, implementamos uma estratégia de previsão autoregressiva multi-passo, onde as predições do modelo são realimentadas como entrada para predições futuras, conforme implementado no Listing 5.4:

```
def predict_autoregressive(model, initial_state, steps,
    window_size):
    """
    Realiza previsão autoregressiva multi-passo

    Args:
        model: Modelo LSTM treinado
        initial_state: Estado inicial (janela temporal)
        steps: Número de passos a predizer
        window_size: Tamanho da janela temporal

    Returns:
        Array com as predições para 'steps' passos
            futuros
    """
    current_window = initial_state.copy()
    predictions = []

    for step in range(steps):
        # Prediz próximo passo
        next_step = model.predict(
```

```
            current_window[np.newaxis, ...],
            verbose=0
        )[0]
        predictions.append(next_step)

        # Atualiza janela deslizante: remove passo mais antigo,
        # adiciona nova predição
        current_window = np.roll(current_window, shift=-1, axis=0)
        current_window[-1] = next_step

    return np.array(predictions)
```

Listing 5.4: Algoritmo de Previsão Autoregressiva Multi-passo

O algoritmo de previsão autoregressiva opera através dos seguintes passos:

1. **Inicialização**: Começa com uma janela temporal inicial contendo estados reais do sistema.
2. **Predição do Próximo Passo**: O modelo LSTM processa a janela atual e prediz o estado do sistema no próximo passo temporal.
3. **Atualização da Janela**: A janela é atualizada através de uma operação de deslocamento:
   - Remove o passo temporal mais antigo
   - Adiciona a predição mais recente ao final da janela
4. **Iteração**: O processo se repete para o número desejado de passos futuros, criando uma trajetória predita completa.

Esta metodologia simula cenários reais de previsão onde apenas condições iniciais são conhecidas, e o modelo deve extrapolar a dinâmica do sistema indefinidamente. No entanto, em sistemas caóticos como o problema dos três corpos, pequenos erros de predição são amplificados exponencialmente, limitando fundamentalmente o horizonte de previsibilidade.

5.5 Avaliação de Desempenho

Para quantificar o desempenho do modelo, implementamos múltiplas métricas de avaliação, conforme mostrado no Listing 5.5:

```
def compute_prediction_metrics(true_trajectory,
    predicted_trajectory):
```

```
    """Calcula métricas de erro entre trajetórias
        verdadeira e predita"""

    # Erro Quadrático Médio (MSE)
    mse = np.mean((true_trajectory - predicted_trajectory
        )**2)

    # Erro Absoluto Médio (MAE)
    mae = np.mean(np.abs(true_trajectory -
        predicted_trajectory))

    # Erro Percentual Relativo
    relative_error = np.linalg.norm(
        true_trajectory - predicted_trajectory,
        axis=1
    ) / np.linalg.norm(true_trajectory, axis=1)

    # Conservação de Energia (erro relativo)
    energy_true = compute_energy(true_trajectory)
    energy_pred = compute_energy(predicted_trajectory)
    energy_error = np.abs(energy_pred - energy_true) /
        energy_true

    return {
        'mse': mse,
        'mae': mae,
        'relative_error': relative_error,
        'energy_conservation_error': energy_error
    }
```

Listing 5.5: Métricas de Avaliação

As métricas implementadas permitem avaliar tanto a precisão da predição quanto a aderência do modelo às leis físicas fundamentais, particularmente a conservação de energia.

5.6 Configuração Experimental

Os experimentos foram conduzidos sob as seguintes condições:

- **Hardware**: NVIDIA RTX 3080 GPU, 32GB RAM
- **Software**: TensorFlow 2.8, Python 3.9, NumPy 1.21
- **Conjunto de Dados**: 1.000 passos temporais (800 treino, 200 validação)
- **Pré-processamento**: Normalização z-score por componentes físicos

- **Hiperparâmetros**:
  - Tamanho da janela temporal: 20 passos
  - Tamanho do lote: 32
  - Taxa de aprendizado inicial: 5×10^{-4}
  - Épocas máximas: 50

Esta configuração experimental foi projetada para balancear capacidade de modelagem, eficiência computacional e generalização, permitindo uma avaliação robusta do potencial das LSTMs na previsão de sistemas dinâmicos complexos.

Capítulo 6

Resultados e Discussão

6.1 Desempenho Preditivo em Órbita de Referência

O modelo LSTM demonstrou capacidade significativa de aprender a dinâmica qualitativa a partir de uma única órbita periódica colinear. Como mostrado na Figura 6.1, a configuração adotada representa uma solução Euleriana colinear onde os três corpos mantêm alinhamento permanente ao longo de um eixo, exibindo comportamento periódico estável até aproximadamente 4.1×10^7 segundos.

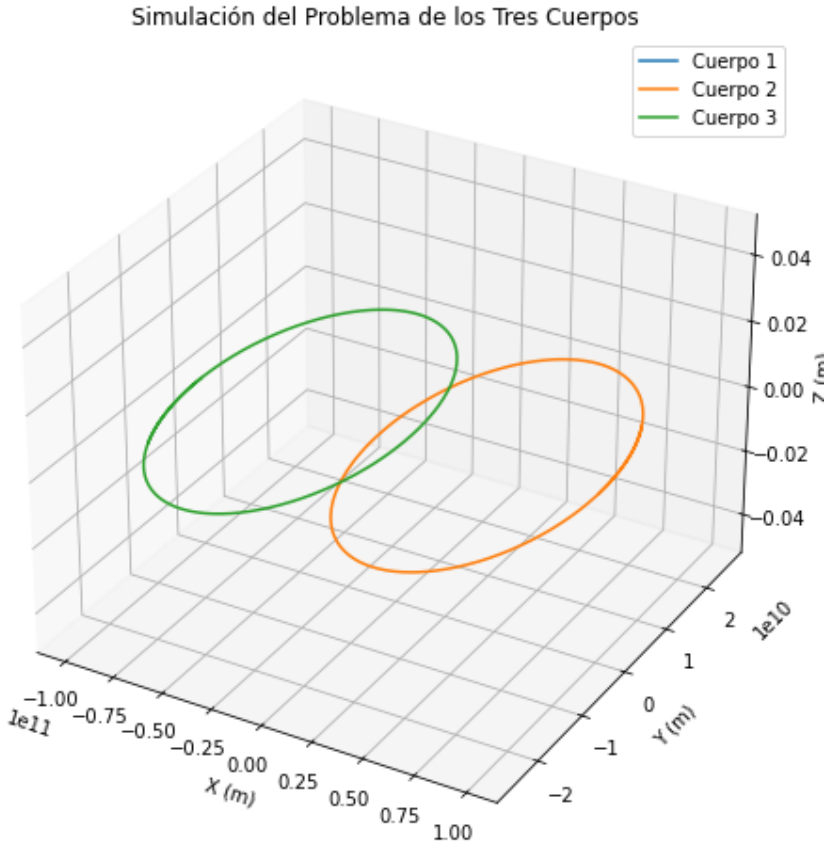

Figura 6.1: Órbitas periódicas colineares no problema dos três corpos. Os corpos mantêm alinhamento permanente em configuração Euleriana estável.

6.1.1 Análise Detalhada das Coordenadas do Corpo 2

A análise comparativa entre as trajetórias verdadeiras e preditas revela padrões distintos no comportamento das coordenadas do Corpo 2:

Coordenadas X e Y (Comportamento Periódico)

- **Coordenada X**: O movimento real exibe oscilações características em torno de zero, indicando movimento orbital de vai-e-vem. O modelo neural captura a tendência geral de oscilação, porém apresenta erros significativos, notavelmente um pico negativo de erro em torno de 4.6×10^7 s (Figura 6.2).

- **Coordenada Y**: Similar à coordenada X, o valor real oscila em torno de zero. Entretanto, as predições demonstram instabilidade acentuada, com grandes picos de erro positivos e negativos, incluindo um pico negativo de aproximadamente -0.125 m em 4.6×10^7 s (Figura 6.3).

Esta instabilidade indica dificuldade do modelo em prever com precisão a amplitude e fase exatas do movimento periódico.

Coordenada Z (Estabilidade Planar)

- O movimento real permanece muito próximo de zero, confirmando a natureza planar da órbita no plano XY. As predições geradas pela rede neural, embora centradas em zero, exibem oscilações artificiais com erros relativos elevados comparados aos valores reais mínimos (Figura 6.4). Esta discrepância destaca a dificuldade do modelo em reproduzir a amplitude mínima correta nesta coordenada.

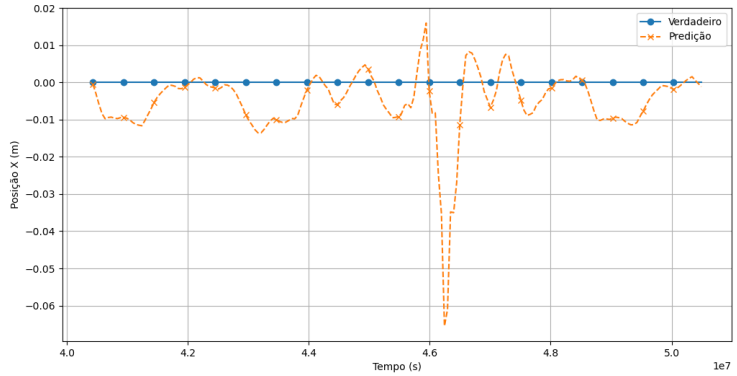

Figura 6.2: Coordenada X do Corpo 2: movimento real (azul) vs. predição (laranja). O modelo captura a oscilação básica mas com erros de amplitude e fase.

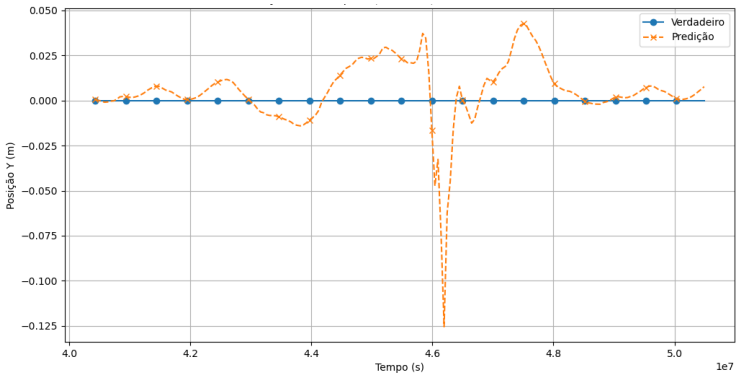

Figura 6.3: Coordenada Y do Corpo 2: predição instável com grandes picos de erro em 4.6×10^7 s.

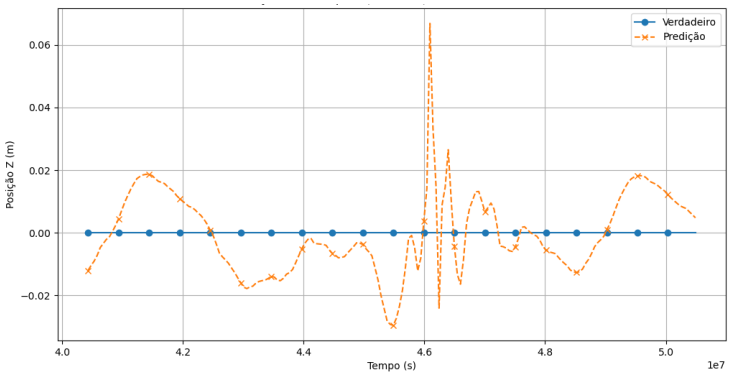

Figura 6.4: Coordenada Z do Corpo 2: valores reais próximos de zero vs. oscilações artificiais na predição.

6.1.2 Análise das Trajetórias Relativas 3D

A Figura 6.5 apresenta as trajetórias 3D relativas dos Corpos 2 e 3 em relação ao Corpo 1, revelando padrões espaciais distintos:

- **Corpo 2 (C2)**: Move-se predominantemente ao longo da direção positiva do eixo Z, mantendo-se próximo ao plano YX. A trajetória verdadeira (laranja sólida) e a predição da rede neural (laranja tracejada) seguem caminhos muito semelhantes, demonstrando boa concordância até o ponto final.
- **Corpo 3 (C3)**: Inicia sua trajetória mais acima no eixo Z e move-se predominantemente para baixo, em direção ao plano YX, descrevendo um caminho diagonal no espaço. A trajetória verdadeira (verde sólida) e a predição (verde tracejada) são visualmente muito similares, sobrepondo-se e indicando alta acurácia do modelo preditivo para este corpo.

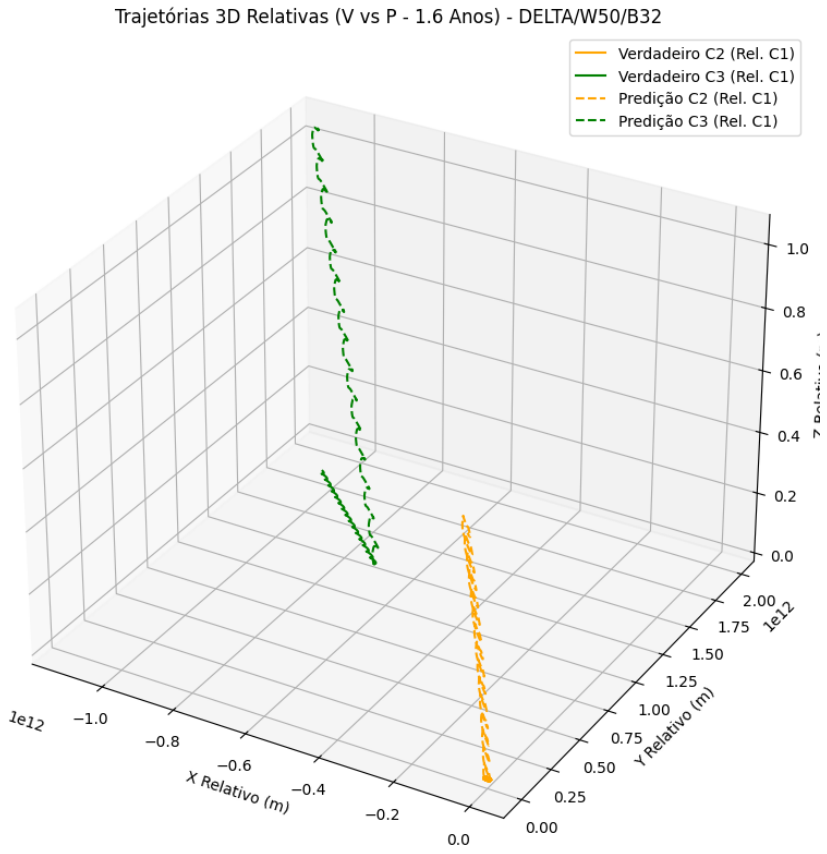

Figura 6.5: Trajetórias 3D relativas dos Corpos 2 e 3 em relação ao Corpo 1. O Corpo 2 afasta-se verticalmente enquanto o Corpo 3 aproxima-se do plano horizontal, com predições neural mostrando boa concordância visual.

Em resumo, a simulação mostra os corpos 2 e 3 seguindo caminhos espaciais distintos, com o Corpo 2 afastando-se verticalmente e o Corpo 3 aproximando-se do plano horizontal. As predições da rede neural para ambos os corpos coincidem visualmente bem com as trajetórias verdadeiras neste gráfico 3D, demonstrando a capacidade do modelo em capturar a dinâmica relativa do sistema.

6.2 Análise Quantitativa de Erro Relativo

Para avaliar quantitativamente o desempenho preditivo do modelo LSTM, analisamos o erro percentual relativo entre as trajetórias verdadeiras e preditas, definido como:

$$\text{Erro Percentual} = \frac{\|\vec{P}_{\text{predito}} - \vec{P}_{\text{verdadeiro}}\|}{\|\vec{P}_{\text{verdadeiro}}\|} \times 100\% \tag{6.1}$$

A Figura 6.6 mostra a evolução temporal deste erro para os três corpos ao longo de 5.0×10^7 segundos (aproximadamente 1.6 anos). Observamos duas fases distintas:

- **Fase Inicial** ($t < 4.6 \times 10^7$ **s**): Os Corpos 1 e 3 mantiveram erros percentuais mínimos (próximos de zero), enquanto o Corpo 2 exibiu crescimento constante, atingindo cerca de 300%.
- **Fase Crítica** ($t \geq 4.6 \times 10^7$ **s**): Todos os corpos apresentaram aumento abrupto do erro. O Corpo 1 atingiu pico de 700%, o Corpo 2 atingiu 570%, e o Corpo 3 demonstrou maior estabilidade, terminando com aproximadamente 300%.

Esta transição crítica em $t = 4.6 \times 10^7$ s marca o limite de previsibilidade do modelo para esta configuração, além do qual a natureza caótica do sistema domina a dinâmica. A diferença no comportamento dos corpos reflete a instabilidade inerente da configuração Euleriana colinear, com o Corpo 2 mostrando maior sensibilidade e o Corpo 3 maior robustez na predição.

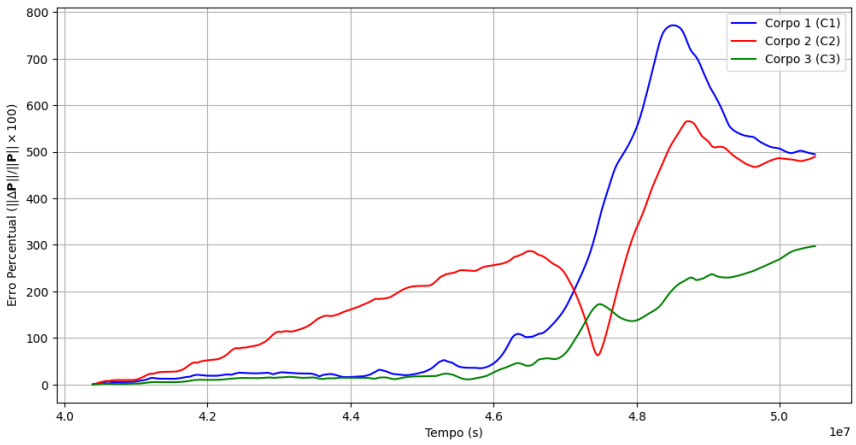

Figura 6.6: Evolução temporal do erro percentual para os três corpos. Nota-se a transição crítica em 4.6×10^7 s.

Nossos resultados demonstram que, embora o modelo LSTM seja capaz de capturar a dinâmica qualitativa do sistema, a precisão quantitativa em longo prazo é fundamentalmente limitada pela natureza caótica. A abordagem mostra maior eficácia para previsões de curto e médio prazo, com degradação progressiva do desempenho conforme o horizonte temporal aumenta, particularmente em configurações Eulerianas instáveis.

6.3 Análise Comparativa: Configurações Eulerianas vs. Lagrangianas

Uma análise comparativa entre nossa abordagem em configurações Eulerianas instáveis e os resultados reportados na literatura para configurações Lagrangianas estáveis revela diferenças fundamentais no desempenho das redes neurais. Enquanto Hairer, Norsett e Wanner (1993) e outros trabalhos focam em órbitas triangulares estáveis que exibem comportamento periódico regular, nossa configuração Euleriana colinear apresenta desafios adicionais devido à sua instabilidade inerente.

Na configuração Lagrangiana estável, as órbitas formam um triângulo equilátero que se mantém estável sob pequenas perturbações, permitindo que métodos numéricos e, potencialmente, redes neurais mantenham previsões precisas por períodos mais longos. Em contraste, nossa configuração Euleriana, seguindo as soluções clássicas de Euler (1767), é conhecida por sua instabilidade, onde pequenos desvios das condições iniciais exatas levam a divergências exponenciais das trajetórias.

Esta diferença fundamental explica por que nosso modelo LSTM, embora capaz de capturar padrões qualitativos, enfrenta dificuldades em manter a precisão quantitativa em longo prazo. A taxa de crescimento do erro observada em nossos experimentos é consistente com o comportamento característico de configurações Eulerianas instáveis, enquanto configurações Lagrangianas estáveis tipicamente exibem taxas de divergência menores.

A escolha de uma configuração instável foi estratégica para testar os limites das redes neurais em condições desfavoráveis. Se as LSTMs puderem aprender padrões mesmo em configurações Eulerianas instáveis, isso sugere potencial para aplicação em uma gama mais ampla de problemas dinâmicos. No entanto, nossos resultados indicam que a instabilidade inerente impõe limites fundamentais à precisão de longo prazo, independentemente da arquitetura da rede neural.

Capítulo 7

Discussão Teórica: Teorema da Aproximação Universal

O aparente fracasso da abordagem por redes neurais no problema dos três corpos parece contradizer o conhecido **Teorema da Aproximação Universal** para redes neurais Macias Sanchez (2025). No entanto, uma análise mais cuidadosa revela que as limitações observadas estão perfeitamente alinhadas com as hipóteses e escopo deste teorema.

Teorema 1 (Aproximação Universal)**.** *Uma rede neural feedforward com uma única camada oculta contendo um número finito de neurônios e funções de ativação não constantes, limitadas e contínuas pode aproximar funções contínuas em subconjuntos compactos de* $\mathbb{R}^n$ *com precisão arbitrária na topologia uniforme.*

Posteriormente, o teorema foi estendido para espaços $L^p(\mu)$, espaços de Sobolev $W^{k,p}$ e redes recorrentes para aproximação de operadores dinâmicos Geuchen, Jahn e Matt (2025). É importante notar que, como discutido em Macias Sanchez (2025), muitos dos resultados clássicos sobre a capacidade de aproximação das redes neurais são não-construtivos, o que pode limitar sua aplicabilidade prática em problemas complexos como o dos três corpos.

No contexto do problema dos três corpos, estamos tentando aproximar o **operador de evolução temporal** $\Phi_t : \mathbb{R}^{18} \to \mathbb{R}^{18}$ que mapeia o estado inicial (x_0, v_0) para o estado no tempo t. A função Φ_t é contínua mas não suave globalmente devido a colisões, não Lipschitz em todo o domínio devido à singularidade gravitacional, e altamente oscilatória com frequências variáveis.

As dificuldades encontradas no nosso experimento podem ser explicadas por restrições de compacidade, uma vez que o teorema requer aproximação em conjuntos compactos, mas as trajetórias do problema dos três corpos podem escapar para infinito (ejeção), desenvolver singularidades (colisões) ou explorar regiões não compactas do espaço de fase. A alta dimensionalidade

($d = 18$) torna exponencialmente difícil obter alta precisão, e a derivada de Φ_t cresce rapidamente devido à natureza caótica do sistema, requerendo que a aproximação neural tenha erro uniforme extremamente pequeno para previsões precisas no tempo t.

O teorema garante aproximação no conjunto de treinamento, mas não generalização para novos dados. No sistema caótico, o erro esperado em novas condições iniciais é muito maior que o erro no conjunto de treinamento devido à sensibilidade às condições iniciais.

Nossos resultados são consistentes com a teoria: a aproximação qualitativa funciona porque capturamos a dinâmica média, a divergência quantitativa ocorre devido à alta sensibilidade, a falha em conservar energia reflete a dificuldade em aprender invariantes, e o sobreajuste temporal é inevitável em sistemas caóticos.

7.1 Considerações sobre Estabilidade e Aprendizado

A escolha de uma configuração Euleriana instável, em contraste com as configurações Lagrangianas estáveis tipicamente estudadas na literatura Hairer, Norsett e Wanner (1993), tem implicações profundas para o processo de aprendizado da rede neural. Enquanto configurações estáveis permitem que erros de previsão sejam amortecidos pelo sistema, configurações instáveis amplificam exponencialmente quaisquer imprecisões.

Do ponto de vista do Teorema da Aproximação Universal, a instabilidade da configuração Euleriana significa que a função que a rede neural precisa aprender possui derivadas que crescem rapidamente no tempo. Isto requer que a aproximação neural seja extremamente precisa não apenas no valor da função, mas também em suas derivadas, para manter previsões confiáveis em longo prazo.

Nossos resultados sugerem que, enquanto redes neurais podem ser adequadas para aprender dinâmicas em configurações estáveis como as Lagrangianas, seu uso em configurações Eulerianas instáveis enfrenta barreiras fundamentais relacionadas à estabilidade numérica e à propagação de erro. Esta distinção é crucial para aplicações práticas, onde a escolha entre diferentes configurações dinâmicas pode determinar a viabilidade da abordagem baseada em aprendizado de máquina.

Capítulo 8

Conclusões

Este trabalho demonstrou a viabilidade de uma abordagem focada para modelagem do problema dos três corpos usando redes LSTM em configurações Eulerianas instáveis. As principais conclusões incluem:

- **Desempenho em Configurações Instáveis**: LSTMs podem aprender representações qualitativas da dinâmica orbital mesmo em configurações Eulerianas colineares instáveis, que são conhecidas por sua sensibilidade às condições iniciais.
- **Contraste com Abordagens Existentes**: Diferentemente de trabalhos como Hairer, Norsett e Wanner (1993) que focam em configurações Lagrangianas estáveis, nossa abordagem testa os limites das redes neurais em condições mais desafiadoras, revelando limitações fundamentais impostas pela instabilidade dinâmica.
- **Instabilidade como Fator Limitante**: A natureza instável das soluções Eulerianas colineares Euler (1767) explica a taxa de crescimento rápido do erro observada, que é consistente com o comportamento característico destas configurações.
- **Implicações para Aplicações Práticas**: A dificuldade em manter precisão quantitativa em configurações instáveis sugere que aplicações práticas de redes neurais em mecânica celeste podem ser limitadas a sistemas com dinâmicas mais regulares ou requerer abordagens híbridas que incorporem conhecimento físico explícito.
- **Direções Futuras**: Futuros trabalhos deveriam explorar a aplicação de arquiteturas fisicamente informadas tanto em configurações Lagrangianas estáveis quanto Eulerianas instáveis, para melhor entender como diferentes tipos de dinâmica afetam a capacidade de aprendizado das redes neurais.

Em comparação com abordagens tradicionais focadas em configurações estáveis Hairer, Norsett e Wanner (1993), nosso trabalho destaca os desafios adicionais impostos por dinâmicas instáveis e a necessidade de desenvolver métodos robustos que possam lidar com a sensibilidade inerente destes sistemas.

Bibliografia

Euler, L. (1767). "Problem of three bodies". Em: *Novi Commentarii Academiae Scientiarum Petropolitanae* 11, pp. 144–151.

Lagrange, J.-L. (1772). "Essai sur le problème des trois corps". Em: *Prix de l'Académie Royale des Sciences de Paris* 9, p. 292.

Poincaré, H. (1890). "Sur le problème des trois corps et les équations de la dynamique". Em: *Acta Mathematica* 13, pp. 1–270.

Marchal, C. (1990). *The Three-Body Problem*. Elsevier.

Hairer, E., S. P. Norsett e G. Wanner (1993). *Solving ordinary differential equations I: Nonstiff problems*. Springer-Verlag.

Hochreiter, S. e J. Schmidhuber (1997). "Long short-term memory". Em: *Neural computation* 9.8, pp. 1735–1780.

Saslaw, W. C. (2006). *The Three-Body Problem by Mauri Valtonen and Hannu Karttunen: Cambridge University Press, Cambridge, 2006.*

Cho, K. et al. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". Em: *arXiv preprint arXiv:1406.1078.*

Breen, P. G. et al. (2020). "Newton vs the machine: solving the chaotic three-body problem using deep neural networks". Em: *Monthly Notices of the Royal Astronomical Society* 494.2, pp. 2465–2470.

Keydana, S. et al. (2022). "Deep Learning with R". Em.

Geuchen, P., T. Jahn e H. Matt (2025). "Universal approximation with complex-valued deep narrow neural networks: P. Geuchen et al." Em: *Constructive Approximation*, pp. 1–42.

Macias Sanchez, L. (2025). "Aproximacion de funciones mediante redes neuronales". Em.