

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO**

GUSTAVO ROSSI MARTINS

**USO DE REDES NEURAIS PARA MODELAGEM DE
PROPAGAÇÃO NÃO LINEAR DE PULSOS ÓTICOS**

VITÓRIA
2022

GUSTAVO ROSSI MARTINS

USO DE REDES NEURAIS PARA MODELAGEM DE PROPAGAÇÃO NÃO LINEAR DE PULSOS ÓTICOS

Parte manuscrita do Projeto de Graduação do aluno **Gustavo Rossi Martins**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Carlos Eduardo Schmidt Castellani

Coorientador: Prof. Dr. Helder Roberto de Oliveira Rocha

VITÓRIA
2022

GUSTAVO ROSSI MARTINS

USO DE REDES NEURAIS PARA MODELAGEM DE PROPAGAÇÃO NÃO LINEAR DE PULSOS ÓTICOS

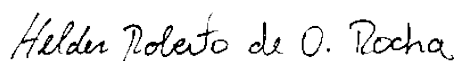
Parte manuscrita do Projeto de Graduação do aluno **Gustavo Rossi Martins**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 21 de março de 2022.

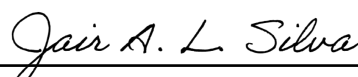
COMISSÃO EXAMINADORA:



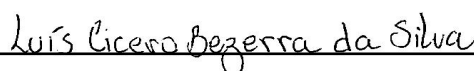
**Prof. Dr. Carlos Eduardo Schmidt
Castellani**
Universidade Federal do Espírito Santo
Orientador



**Prof. Dr. Helder Roberto de Oliveira
Rocha**
Universidade Federal do Espírito Santo
Coorientador



Prof. Dr. Jair Adriano Lima Silva
Universidade Federal do Espírito Santo
Examinador



Prof. Dr. Luís Cícero Bezerra da Silva
Universidade Federal do Espírito Santo
Examinador

*Ao meu pai, João Martins, que morreu sem saber que entrei na Universidade.
A tantos outros, vítimas da pandemia de Covid-19.*

AGRADECIMENTOS

Gostaria de agradecer a todas essas pessoas especiais que apareceram na minha vida e que me presentearam com sua amizade.

À minha família, e as mulheres que me fizeram e fazem crescer: minha avó Maria Lúcia pela sabedoria, minha irmã Iasmim pelo amor e carinho, e minha mãe Elisangela por tudo. E um agradecimento especial à minha tia Joelcia Fanchiotti por todo suporte nesses cinco anos.

À turma 2017/1 que tive o prazer de fazer parte, e aos amigos que conheci na Universidade, em especial a Aiury Jureswski, Ramon de Angeli e Pedro Henrique, que ainda me ajudam e aturam, mesmo a distância. Também àqueles de fora (nem tanto), que dividiram o apartamento 301 comigo: Carlos Augusto, o grande matemático Rafael Almeida e José Matheus, quem começou essa jornada comigo.

Aos meus professores da UFES por todo conhecimento passado, em especial aos meus orientadores: ao Prof. Helder Rocha, sempre solícito para tirar dúvidas, e ao Prof. Carlos Eduardo, pelo acompanhamento de anos e pelos desafios que aos poucos se transformaram nesse trabalho de graduação.

À banca examinadora pelo tempo investido para leitura e avaliação desse trabalho.

Agradeço por fim à Universidade Federal do Espírito Santo pela minha formação.

*“Lave o rosto nas águas sagradas da pia,
Nada como um dia após o outro dia.”*

Racionais mc's

RESUMO

As fibras óticas formam o *backbone* do sistema de telecomunicações moderno, interligando países e continentes via extensos cabos submarinos. Mesmo com uma atenuação de apenas 0,2 dB/km, esse meio de comunicação não é tão transparente ou passivo, e seus efeitos de dispersão e auto-modulação de fase podem ser destrutivos para os sistemas de comunicação. Tais efeitos são demasiado complexos e exigem simulações computacionais sofisticadas baseadas na equação não linear de Schrödinger (NLSE). O algoritmo estado-da-arte *split-step Fourier* é uma solução simples para avaliar a propagação de pulsos óticos, contudo, esse método pode ser um gargalo para experimentos em tempo real ou com um grande número de simulações numéricas. O presente projeto de graduação faz uso de redes neurais para prever a evolução temporal no caso de propagação não linear de pulsos óticos, dispensando o uso de soluções numéricas no modelo.

Devido ao caráter temporal do problema de propagação, faz sentido o uso de redes recorrentes como a rede *long short-term memory* (LSTM). No trabalho foram testadas duas redes, a LSTM e a rede *convolutional LSTM* (ConvLSTM). A primeira apresentou como vantagem sua rapidez, sendo em média 2,9 vezes mais rápida que o algoritmo SSF na predição da base de testes. Já a rede ConvLSTM mitigou os erros, tais como a raiz do erro quadrático médio, $RMSE = 0,25\%$, e o erro absoluto médio, $MAE = 0,15\%$.

As redes também foram estendidas para um conjunto com *range* distinto do conjunto de testes, sem retreiná-las. O comportamento das redes foi similar, mas as métricas de erros pioraram, o RMSE aumentou 60% e o MAE aumentou 40%.

Palavras-chave: Fibra ótica; Propagação de pulsos; Equação não linear de Schrödinger; Método *split-step Fourier*; Redes neurais; LSTM; ConvLSTM.

ABSTRACT

Optical fibers have become the backbone of modern telecommunication systems, linking countries, and continents through long submarine cables. Even with an attenuation of only 0.2 dB/km, this communication channel is not so transparent or passive, since its self-phase modulation and dispersive effects can be destructive for the communication system. Such effects are highly complex and require computationally demanding simulations based on the nonlinear Schrödinger equation (NLSE). The split-step Fourier state-of-the-art algorithm is a simple solution to evaluate the propagation of optical pulses, however, this method creates a severe bottleneck for real-time experiments or with a great number of numerical simulations. This undergraduate project uses neural networks to predict the temporal evolution in case of nonlinear pulse propagation, bypassing the need for numerical solutions.

Due to temporal characteristics of the propagation problem, it makes sense the use of recurrent networks likewise long short-term memory (LSTM). In this work were tested two networks, the LSTM and the convolutional LSTM (ConvLSTM) network were. The first one presented as an advantage its speed, being on average 2.9 times faster than the SSF algorithm for the prediction of the test dataset. The ConvLSTM network diminished the errors, such as the root mean squared error, $RMSE = 0.25\%$, and the mean absolute error, $MAE = 0.15\%$.

The networks also were extended to a dataset with a different range from the test dataset, without retraining. The network's behavior was similar, but the error metrics got worse, the RMSE increased 60% and the MAE increased 40%.

Keywords: Optical fiber; Pulse propagation; Nonlinear Schrödinger Equation; *Split-step* Fourier method; Neural networks; LSTM; ConvLSTM.

LISTA DE FIGURAS

Figura 1 – Fibra ótica	17
Figura 2 – Formatos dos pulsos.	19
Figura 3 – Curvas do modo de propagação $\beta(\omega)$.	21
Figura 4 – Efeito do alargamento de pulso.	22
Figura 5 – Efeito do alargamento espectral.	23
Figura 6 – Esquemático do método split-step Fourier.	24
Figura 7 – Nó Perceptron	26
Figura 8 – Exemplo de redes neurais clássicas.	27
Figura 9 – Nó Recorrente.	28
Figura 10 – Módulo interno RNN.	28
Figura 11 – Módulo interno LSTM.	29
Figura 12 – Modelo de rede ConvLSTM.	31
Figura 13 – Intensidade temporal para diferentes distâncias de um soliton de ordem superior com $P_0 = 26,3$ W e $T_{FWHM} = 1,1$ ps.	34
Figura 14 – Intensidade temporal para diferentes distâncias de um soliton de ordem superior com $P_0 = 34,19$ W e $T_{FWHM} = 0,77$ ps.	34
Figura 15 – Matriz 3D dos Dados.	39
Figura 16 – Divisões do <i>dataset</i> .	42
Figura 17 – <i>Loss</i> durante o treinamento da rede LSTM.	44
Figura 18 – <i>Loss</i> durante o treinamento da rede ConvLSTM.	45
Figura 19 – Soliton com $P_0 = 26,3$ W e $T_{FWHM} = 1,1$ ps. SSF \times LSTM.	48
Figura 20 – Soliton com $P_0 = 34,19$ W e $T_{FWHM} = 0,77$ ps. SSF \times LSTM.	48
Figura 21 – Soliton com $P_0 = 40,0$ W e $T_{FWHM} = 1,46$ ps. SSF \times LSTM.	50
Figura 22 – Soliton com $P_0 = 38,68$ W e $T_{FWHM} = 1,96$ ps. SSF \times LSTM.	50

LISTA DE TABELAS

Tabela 1 – Componentes da LSTM.	29
Tabela 2 – Parâmetros da fibra.	32
Tabela 3 – Dados gerados.	36
Tabela 4 – Dados normalizados.	37
Tabela 5 – Visualizando problema de regressão nos dados.	38
Tabela 6 – Parâmetros de rede e treinamento.	40
Tabela 7 – Arquitetura da rede LSTM.	40
Tabela 8 – Arquitetura da rede ConvLSTM2D.	41
Tabela 9 – Comparação das redes construídas.	45
Tabela 10 – Arquitetura final da rede LSTM.	46
Tabela 11 – Arquitetura final da rede ConvLSTM2D.	47
Tabela 12 – Comparação final das redes construídas.	47
Tabela 13 – Comparação das redes para novos dados.	49
Tabela 14 – Comparação de valores em dB.	61

LISTA DE ABREVIATURAS E SIGLAS

CNN	<i>Convolutional Neural Networks</i>
ConvLSTM	<i>Convolutional LSMT</i>
DL	<i>Deep Learning</i>
FFT	<i>Fast Fourier Transform</i>
FWHM	<i>Full Width at Half Maximum</i>
GVD	<i>Group-Velocity Dispersion</i>
IIS	Interferência Intersimbólica
LSTM	<i>Long Short-Term Memory</i>
ML	<i>Machine Learning</i>
MAE	Erro Absoluto Médio
MSE	Erro Quadrático Médio
NLSE	<i>Nonlinear Schrödinger Equation</i>
R^2	Coefficiente de Determinação
RMSE	Raiz do Erro Quadrático Médio
RNA	Redes Neurais Artificiais
SPM	<i>Self-Phase Modulation</i>
SSF	<i>Split-Step Fourier</i>
TOD	<i>Third-Order Dispersion</i>
UFES	Universidade Federal do Espírito Santo
UV	Ultravioleta
WDM	<i>Wavelength-Division Multiplexing</i>

LISTA DE SÍMBOLOS

a	Raio do núcleo
b	Raio da casca
n_1	Índice de refração do núcleo
n_2	Índice de refração da casca
Φ	Ângulo de incidência
Φ_c	Ângulo crítico
$A(z, T)$	Amplitude do pulso (W)
α	Atenuação total da fibra ótica
β_2	Parâmetro de dispersão GVD (s^2m^{-1})
β_3	Parâmetro de dispersão TOD (s^3m^{-1})
γ	Parâmetro não linear ($W^{-1}m^{-1}$)
$U(z, T)$	Amplitude normalizada do pulso (W)
α_{dB}	Atenuação total da fibra ótica (dB/km)
P_0	Potência de pico do pulso incidente (W)
T_0	Largura do pulso incidente (s)
T_{FWHM}	Largura à meia altura do pulso (s)
N	Ordem do soliton
\mathcal{F}	Transformada direta de Fourier
\mathcal{F}^{-1}	Transformada inversa de Fourier
\odot	Produto Hamadart
\star	Operador de convolução
L_D	Comprimento da dispersão
L'_D	Comprimento da dispersão associado ao TOD
L_{NL}	Comprimento não linear

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Objetivo Geral	16
1.2	Objetivos Específicos	16
2	REFERENCIAL TEÓRICO	17
2.1	Fibra ótica	17
2.2	Equação de Propagação de Pulso	18
2.2.1	Atenuação	19
2.2.2	Dispersão	20
2.2.3	Auto-modulação de fase	22
2.3	Método <i>Split-Step</i> Fourier	23
2.4	Redes Neurais Artificiais	26
2.4.1	Redes Neurais <i>Feed-Forward</i>	27
2.4.2	Redes Neurais Recorrentes	28
2.4.3	Redes Neurais Recorrentes <i>Long Short-Term Memory</i>	29
2.4.4	Redes Neurais Convolucionais LSTM	31
3	METODOLOGIA E ETAPAS DE DESENVOLVIMENTO	32
3.1	Modelagem da Fibra	32
3.2	Algoritmo <i>Split-Step</i>	35
3.3	Tratamento dos Dados	36
3.3.1	Normalização	37
3.3.2	Série Temporal × Regressão	37
3.3.3	Dimensionamento dos dados	38
3.4	Rede Neural	39
3.4.1	Construção	39
3.4.2	Treinamento	42
3.4.3	Métricas	42
4	RESULTADOS	44
4.1	Treinamento e Redes Construídas	44
4.2	Resultado da Melhor Arquitetura	46
4.3	Novos Testes	48
5	CONCLUSÕES E PROJETOS FUTUROS	51
5.1	Conclusões	51
5.2	Projetos futuros	52

REFERÊNCIAS	53
APÊNDICES	56
APÊNDICE A – EQUAÇÕES DE MAXWELL	57
APÊNDICE B – EQUAÇÃO DE PROPAGAÇÃO NÃO LINEAR . . .	59
APÊNDICE C – DECIBEL	61
ANEXOS	62
ANEXO A – CÓDIGO PRINCIPAL	63
ANEXO B – PARAMETRIZAÇÃO	66
ANEXO C – FIBRA (SPLIT-STEP)	67
ANEXO D – REDE NEURAL	68

1 INTRODUÇÃO

As fibras óticas formam o *backbone* do sistema de telecomunicações moderno, interligando países e continentes com maior qualidade e taxa de dados via extensos cabos submarinos^[1]. O primeiro cabo telefônico transatlântico, o TAT-1, surgiu em 1956 levando 36 circuitos de voz por cabos coaxiais arcaicos. Em 1976, o TAT-6 dispunha de 5.200 circuitos e o TAT-7 adicionava mais 4.000 novos circuitos em 1983. O primeiro da série a ser projetado para operar com fibras foi o TAT-8 em 1988, com apenas dois pares monomodos e capacidade de 280 Mbps. O mais moderno desses cabos submarinos é o TAT-14^[2] de 2001, com 4 pares de fibra percorrendo 15.000 km e capacidade projetada de 9,38 Tbps, usando amplificadores óticos baseados em fibras dopadas com érbio e multiplexação por divisão de comprimento de onda (WDM, de *wavelength-division multiplexing*). Como imaginado por Hecht (1999), as fibras óticas se tornaram a tecnologia principal das telecomunicações.

Sistemas de comunicação quilométricos exigem baixas perdas na transmissão, e em 1965 a melhor fibra tinha uma atenuação de 1.000 decibéis por quilômetro. A marca de 20 dB/km foi alcançada no início da década de 1970, um avanço importante para tornar possível sua aplicação em grandes distâncias, papel dividido com redes satélites e guias de ondas metálicos milimétricos, mas insuficiente para predominar sobre as duas tecnologias (HECHT, 1999). A vantagem veio logo em 1976, com novas técnicas de fabricação como deposição de vapor químico e aplicação de germânio como dopante do núcleo, alcançando uma atenuação total de apenas 0,2 dB/km para comprimentos de onda de 1,55 μm (CASTELLANI, 2013).

Mesmo com uma atenuação de 0,2 dB/km, esse meio de comunicação não é tão transparente ou passivo. O *design* do sistema leva em conta os efeitos dispersivos e não lineares cumulativos da fibra, controlados devido as longas distâncias envolvidas. A potência dos *lasers* na transmissão e o perfil de dispersão são otimizados para combater esses efeitos potencialmente destrutivos para a informação (AGRAWAL, 2021).

Os efeitos dispersivos e não lineares são demasiado complexos e exigem simulações computacionais baseadas na equação de Schrödinger não linear (NLSE, de *nonlinear Schrödinger equation*). O método mais comum para tratar a equação de propagação (equação de Schrödinger) é o algoritmo *split-step* Fourier (SSF) que, através de transformações rápidas de Fourier, direta e inversa, calcula, em um número de passos, a evolução do pulso ótico (ALESHAMS; ZARIFKAR; SHEIKHI, 2005; OLIARI et al., 2020). Contudo, esse método pode ser um gargalo severo na otimização de experimentos e situações em tempo real

¹ <<https://www.submarinecablemap.com/>>

² <<https://www.infrapedia.com/app/subsea-cable/tat-14-eol-dec-2020>>

devido ao número elevado de iterações, por exemplo na transmissão de sinais por centenas de quilômetros, ou na simulação de lasers, que naturalmente exige milhares de voltas em uma cavidade (SALMELA et al., 2021). O presente projeto de graduação faz uso de redes neurais para o caso de propagação não linear de pulsos óticos, dispensando o uso de soluções numéricas no modelo, buscando dessa forma uma solução rápida e robusta para o problema.

O campo de *machine learning* tem evoluído rapidamente e suas aplicações em controle de veículos autônomos, processamento de linguagem, *health-care*, visão computacional, entre outros, tem mostrado a capacidade e adaptabilidade para uma variedade de problemas complexos (JORDAN; MITCHELL, 2015). Em fotônica, trabalhos fazendo o uso dessa poderosa ferramenta em diversas aplicações tem se destacado (GENTY et al., 2021). Por exemplo, problemas clássicos de classificação por meio de redes profundas feito por Närhi et al. (2018), ou ainda problemas mais complexos de propagação feito por Salmela et al. (2021), que inspira o presente trabalho.

Aqui, o uso do algoritmo SSF tem como finalidade gerar o banco de dados usado na construção do modelo neural. Esses dados são obtidos simulando (com o SSF) uma fibra com características físicas mantidas inalteradas (sua atenuação e outras constantes explicadas no corpo do texto), e variando o perfil do pulso limitado pela transformada de Fourier (*transform-limited*) de entrada na fibra ótica, sua potência de pico, P_0 , e sua largura de pulso, T_0 ou T_{FWHM} . O interesse do problema se resume à evolução temporal, e não à sua contraparte no espectro. A matriz de dados gerados com o algoritmo SSF serão mais tarde organizados e normalizados para servirem de entrada no modelo de rede neural construída.

Devido ao caráter dinâmico do problema de propagação, faz sentido o uso de redes recorrentes como a *long short-term memory* (LSTM) criada por Hochreiter e Schmidhuber (1997), com o fator tempo sendo de fundamental importância. Ainda, a busca de um modelo adequado ao problema de evolução temporal do pulso ótico não se limita a abordagem com a LSTM, e podemos combiná-la com camadas de redes densas ou ainda camadas de redes convolucionais (CNN, de *convolutional neural network*).

Para avaliar a performance do modelo, fazemos uso de métricas como o erro quadrático médio (MSE, de *mean squared error*), e sua raiz quadrada (RMSE, de *root mean squared error*), como forma de comparação dos dados obtidos do SSF com os valores previstos pelo modelo neural para quantificar sua aproximação com o método estado-da-arte SSF.

1.1 Objetivo Geral

O objetivo desse trabalho é elaborar um modelo neural para o problema de propagação não linear de pulsos óticos, com desempenho similar ao algoritmo estado-da-arte SSF empregado, e menor custo computacional.

1.2 Objetivos Específicos

- Elaborar o algoritmo SSF para o problema de propagação de pulso e de acordo com as características da fibra;
- Gerar os dados a partir do algoritmo SSF;
- Obter um modelo neural que capture os efeitos da transmissão na fibra;
- Validar e comparar os resultados obtidos com o modelo neural;

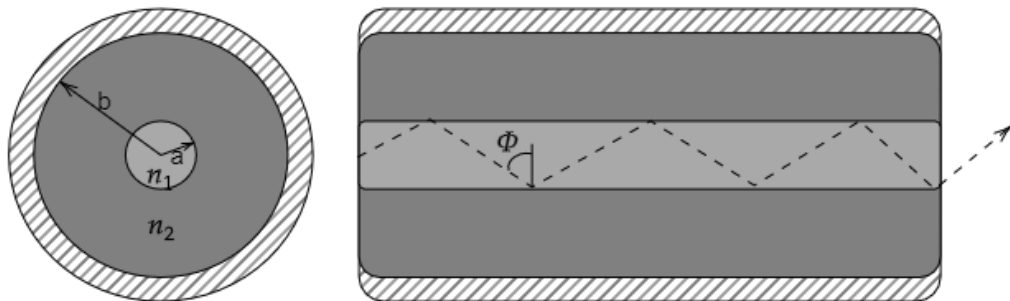
2 REFERENCIAL TEÓRICO

2.1 Fibra ótica

A fibra ótica é formada por um núcleo (ou *core*) cilíndrico vítreo de raio a e índice de refração n_1 envolto em uma casca (ou *cladding*) tubular com raio b e índice n_2 , como pode ser visto na Figura 1. Ambos núcleo e casca são constituídos de dióxido de silício (SiO_2), ou sílica, e são adicionados dopantes na sua estrutura para alterar os índices de refração de modo que $n_1 > n_2$. Dopantes como GeO_2 e P_2O_5 aumentam o índice da sílica enquanto outros como o boro o diminui. O núcleo possui raio a de ordem micrométrica, tipicamente $25 \mu\text{m}$ para fibras multimodo e raio $a < 5 \mu\text{m}$ para fibras monomodo (AGRAWAL, 2007).

Apresentam características importantes como taxa de transmissão elevada, baixas perdas, imunidade a interferência eletromagnética, isolamento elétrico da sílica (dielétrico), robustez mecânica e, devido ao núcleo e casca micrométricos, são compactas (CERTÓRIO, 2009).

Figura 1 – Fibra ótica



Fonte: Próprio autor.

A relação entre os índices é fundamental para que ocorra a propagação de luz através da reflexão interna total mostrada na Figura 1 acima. Para isso, o ângulo de incidência Φ deve ser maior que o ângulo crítico Φ_c descrito na Equação (2.1) obtido da lei de Snell,

$$\Phi_c = \arcsin\left(\frac{n_2}{n_1}\right). \quad (2.1)$$

A reflexão interna total é útil na compreensão da fibra como um *pipeline* para a luz que se propaga por ela, mas o entendimento completo exige uma análise das equações de Maxwell (Apêndice A) em guia de ondas.

2.2 Equação de Propagação de Pulso

A equação que governa a propagação de pulsos óticos em fibra (Apêndice [B](#)) é conhecida como equação de Schrödinger não linear (NLSE) e é dado por

$$i\frac{\partial A}{\partial z} = \frac{\beta_2}{2}\frac{\partial^2 A}{\partial T^2} + i\frac{\beta_3}{6}\frac{\partial^3 A}{\partial T^3} - i\frac{\alpha}{2}A - \gamma|A|^2A \quad (2.2)$$

onde A é o formato ou envelope do pulso ótico, β_2 e β_3 governam os efeitos de dispersão, α engloba as perdas na transmissão, e γ é o parâmetro das não linearidades da interação fibra-pulso. Ainda na equação, i é a unidade imaginária, $i = \sqrt{-1}$, z é a variável de comprimento do eixo longitudinal da fibra, e T é a medida de tempo em relação ao *frame* de referência do envelope ótico. As seções relacionadas a fotônica sem referência explícita foram retiradas do livro-texto do [Agrawal \(2007\)](#).

Todas as constantes da equação acima estão relacionados a fibra ótica, sua fabricação e dopantes usados no núcleo ou casca. O termo A é referente ao envelope do pulso ótico, e a Equação [\(2.2\)](#) governa pulsos ultra curtos ($T_0 < 1\text{ps}$). Por simplificação, consideramos apenas três soluções recorrentes na literatura: pulso em formato gaussiano, $\text{sech}(t)$, ou $\text{sech}^2(t)$.

Além do seu formato, duas informações são importantes para descrever um pulso temporal, T_0 é sua largura inicial e P_0 a potência de pico incidente. Ainda, as equações a seguir são normalizadas, onde $U(0, T)$ é o pulso normalizado, obtido por $A(0, T) = \sqrt{P_0} U(0, T)$.

A Equação [\(2.3\)](#) a seguir descreve um pulso gaussiano

$$U(0, T) = \exp\left(-\frac{T^2}{2T_0^2}\right) \quad (2.3)$$

é comum caracterizar o pulso por sua largura à meia altura T_{FWHM} (FWHM, de *full width at half maximum*), que se relaciona com T_0 pela equação abaixo no caso de pulso gaussiano

$$T_{\text{FWHM}} = 2(\ln 2)^{1/2}T_0 \approx 1,665 T_0. \quad (2.4)$$

Outras duas equações de grande interesse são descritas em [\(2.5\)](#), secante hiperbólica e sech^2 , respectivamente:

$$U(0, T) = N\text{sech}\left(\frac{T}{T_0}\right) \quad U(0, T) = N\text{sech}^2\left(\frac{T}{T_0}\right) \quad (2.5)$$

onde N no contexto de sólitons é sua ordem e é dado por

$$N^2 = \frac{\gamma P_0 T_0^2}{|\beta_2|} \quad (2.6)$$

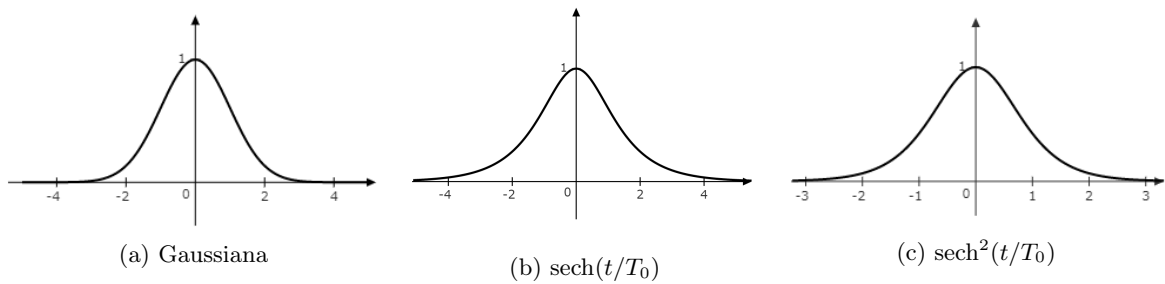
e T_{FWHM} se relaciona com T_0 pela Equação (2.7) abaixo

$$T_{\text{FWHM}} = 2 \ln(1 + \sqrt{2})T_0 \approx 1,763 T_0 \quad (2.7)$$

O interesse pelo pulso da forma de uma secante hiperbólica é por ele surgir naturalmente no contexto de sóliton óticos, isto é, uma onda solitária que se propaga, em condições ideais, com formato constante. O caso apresentado neste trabalho é o da propagação de um sóliton. O sóliton de ordem $N = 1$ é também chamado de fundamental, e seu formato natural é o de uma sech^2 . Idealmente, assumindo γ , P_0 e β_2 constantes, o sóliton fundamental pode ser propagado sem sofrer alterações. Para valores de N maiores, o sóliton de maior ordem apresenta mudanças periódicas temporal e espacialmente (CASTELLANI, 2013).

As equações descritas acima podem ser visualizadas na Figura 2 a seguir, os pulsos são normalizados e tem largura $T_0 = 1$.

Figura 2 – Formatos dos pulsos.



Fonte: Próprio autor.

As subseções 2.2.2 e 2.2.3 seguintes tratam dos efeitos de dispersão e auto-modulação de fase separadamente de forma a entender a natureza dos fenômenos que ocorrem na propagação de pulsos em fibra ótica, a ação conjunta desses efeitos lineares e não lineares não é tratado analiticamente neste capítulo, visto que o algoritmo SSF (seção 2.3) é adotado para essa situação.

2.2.1 Atenuação

A sílica usada na fabricação das fibras é um material transparente com grau de pureza elevado. Ainda na década de 70, medições feitas por Kapron, Keck e Maurer (1970) mencionam fibras de 20 decibéis por quilômetro. Avanços nas técnicas empregadas permitiram reduzir ainda mais essas perdas, a maior parte devido a absorção ultravioleta (UV) em comprimentos de onda mais curtos e por íons de OH em comprimentos de onda maiores. Nas região de 0,5 a 2 μm outro fator que contribui para a atenuação é a dispersão Rayleigh, que devido a flutuações aleatórias da densidade da sílica, e do seu índice de refração, é intrínseca ao processo de fabricação, da temperatura e tensões exercidas no processo.

A perda provocada por Rayleigh pode ser estimada (em dB/km) pela equação seguinte

$$\alpha_R = \frac{C_R}{\lambda^4}, \quad (2.8)$$

onde λ é o comprimento de onda e C_R depende dos componentes da fibra e varia nos valores entre 0,7 e 0,9 dB/(km- μm^4). Pela Equação (2.8), fica evidente que o espalhamento Rayleigh torna inviável o uso de comprimentos de onda abaixo de 0,5 μm .

Atualmente, em fibras comerciais a atenuação para o comprimento de 1,55 μm é cerca de 0,2 dB/km, o que significa dizer que um sinal ótico teria 1% da sua potência inicial após propagar por 100 km. Para a melhor fibra dos anos 70 seria uma distância de apenas 1 km.

Se P_0 é a potência na entrada da fibra de comprimento L , a potência na saída P_T é dada por

$$P_T = P_0 \exp(-\alpha L) \quad (2.9)$$

onde α é a constante de atenuação total. Normalmente α é dado em dB/km pela relação

$$\alpha = -\frac{1}{L} \ln \left(\frac{P_T}{P_0} \right), \quad (2.10)$$

$$\alpha_{dB} = -\frac{10}{L} \log \left(\frac{P_T}{P_0} \right) \quad (2.11)$$

A razão de α e α_{dB} , tomando $R = P_T/P_0$, é

$$\frac{\alpha_{dB}}{\alpha} = 10 \frac{\log R}{\ln R} = 10 \log e = 4,343 \quad (2.12)$$

Nas informações comerciais das fibras, é comumente usado α_{dB} , mas nos cálculos matemáticos é necessário fazer a conversão $\alpha_{dB} = 4,343\alpha$.

2.2.2 Dispersão

A dispersão cromática, ou dispersão, é um fenômeno linear que surge da interação da onda eletromagnética com a fibra e que se manifesta através da dependência em frequência do índice de refração $n(\omega)$. O índice de refração relaciona a velocidade de uma onda no canal de propagação por $v = c/n$. Em razão da dispersão, um sinal ótico tem seus diferentes componentes espectrais viajando cada um em uma velocidade $c/n(\omega)$.

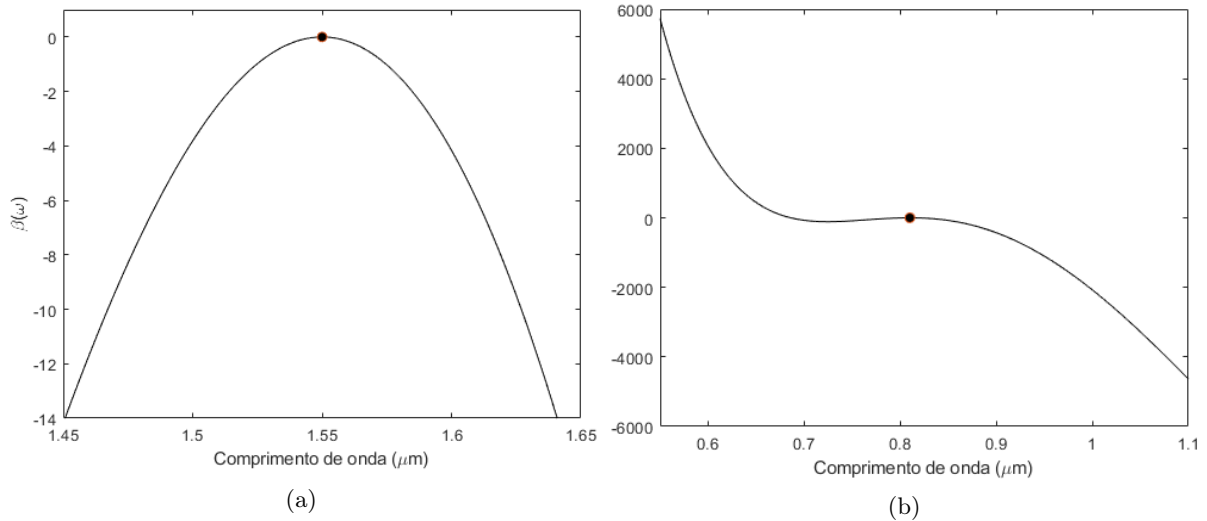
Matematicamente, os efeitos da dispersão são dados pelos componentes da constante de propagação β escrita numa série de Taylor:

$$\beta(\omega) = n(\omega) \frac{\omega}{c} = \beta_0 + \beta_1(\omega - \omega_0) + \frac{1}{2} \beta_2(\omega - \omega_0)^2 + \dots \quad (2.13)$$

onde

$$\beta_m = \left(\frac{d^m \beta}{d\omega^m} \right)_{\omega=\omega_0} \quad (2.14)$$

Figura 3 – Curvas do modo de propagação $\beta(\omega)$.



Fonte: Próprio autor.

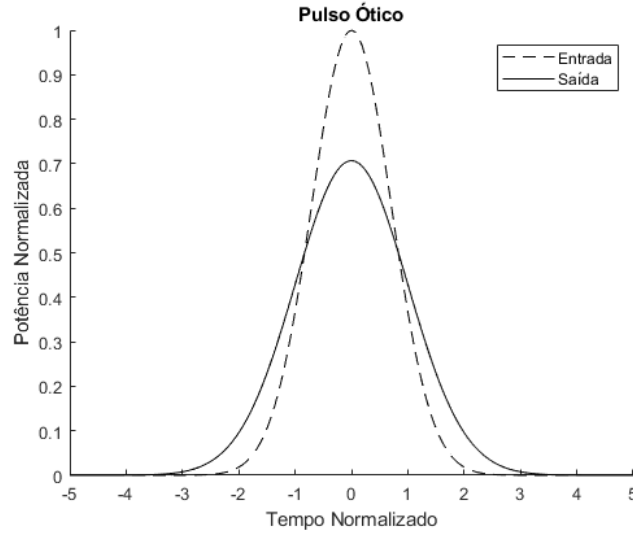
Na equação de propagação (2.2) levamos em conta a participação de dois componentes da série de Taylor, β_2 e β_3 . O parâmetro β_2 é responsável pelo alargamento dos pulsos e é conhecido como parâmetro de dispersão de velocidade de grupo (GVD, de *group-velocity dispersion*). E o parâmetro β_3 , incluído para propagação de pulsos ultra curtos ($T_0 < 1$ ps), é chamado de parâmetro de dispersão de terceira ordem (TOD, de *third-order dispersion*). Componentes de maior ordem são necessários somente em casos específicos.

A Figura 3 mostra dois casos da curva $\beta(\omega)$. A série de Taylor aproxima uma curva em torno da frequência central (pontos nas figuras). Na curva 3(a), com $\lambda_0 = 1.55 \mu\text{m}$ e usando os componentes β_2 e β_3 , a confiança para comprimentos de onda distantes é pequena. Na curva 3(b), com $\lambda_0 = 0.81 \mu\text{m}$ e usando os componentes β_2 a β_7 , a confiança para comprimentos de onda distantes é maior, esse é o caso de super-contínuos por exemplo, que ocupam uma larga faixa de frequências.

O efeito mais nocivo da dispersão nos canais de telecomunicações é o alargamento do pulso visto na Figura 4, que aumenta a duração do pulso e diminui sua amplitude, e que por esse motivo pode provocar interferência intersimbólica (IIS). Nesse caso, o efeito é geralmente compensado por meio fibras com dispersão contrária.

O alargamento é devido a diferença de velocidade entre as componentes de frequência, a essa diferença de fases dá-se o nome de *chirp*. Quando o parâmetro GVD é positivo ($\beta_2 > 0$) o regime de propagação é normal e as componentes “vermelhas” do espectro

Figura 4 – Efeito do alargamento de pulso.



Fonte: Próprio autor.

viajam mais rápido que as “azuis”. Quando o parâmetro GVD é negativo ($\beta_2 < 0$) o regime de propagação é anômalo e as componentes “azuis” do espectro viajam mais rápido que as “vermelhas”.

2.2.3 Auto-modulação de fase

Anteriormente foi dito que o índice de refração da fibra ótica é dependente da frequência, $n(\omega)$. Aqui, será abordado os efeitos da sua dependência também na intensidade. Esta subseção trata da auto-modulação de fase (SPM, de *self-phase modulation*) que se manifesta na propagação de pulsos de potência elevada e produz um alargamento espectral desses pulsos.

Simplificando a equação de propagação de modo que tenhamos apenas o efeito do termo de não linearidade γ ($\beta_2, \beta_3, \alpha = 0$), ficamos com

$$\frac{\partial A}{\partial z} = i\gamma|A|^2A. \quad (2.15)$$

A Equação (2.15) pode ser resolvida pela solução geral $A = V \exp(i\phi_{NL})$,

$$e^{i\phi_{NL}} \frac{\partial V}{\partial z} + iV e^{i\phi_{NL}} \frac{\partial \phi_{NL}}{\partial z} = i\gamma V^3 e^{i\phi_{NL}}. \quad (2.16)$$

e tomando então

$$\frac{\partial V}{\partial z} = 0; \quad \frac{\partial \phi_{NL}}{\partial z} = \gamma V^2. \quad (2.17)$$

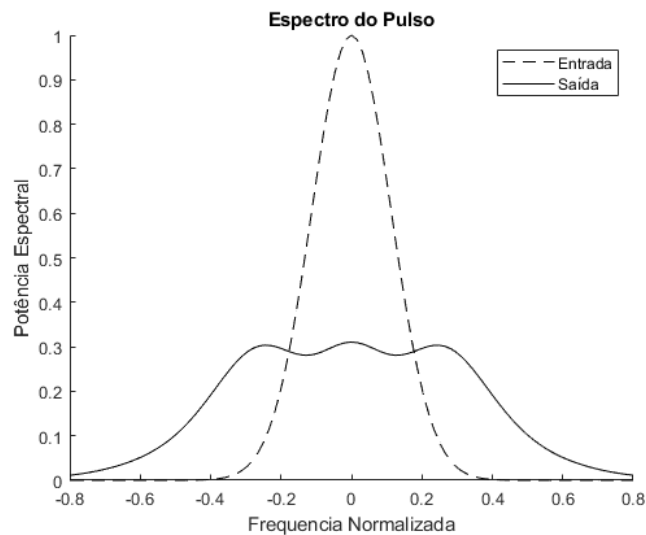
Como a derivada parcial de V é nula em relação a z , podemos afirmar que V é constante ao longo do comprimento da fibra, nesse caso a amplitude do pulso não altera durante a propagação, $V(T) = A(0, T)$,

$$V(T) = A(0, T); \quad \phi_{NL}(z, T) = \gamma|A(0, T)|^2 z. \quad (2.18)$$

E por isso $A(z, T) = A(0, T) \exp[i\phi_{NL}(z, T)]$.

Com as equações (2.18) demonstra-se que o SPM tem como consequência uma fase ϕ_{NL} dependente da intensidade, mas que sozinha não altera o pulso. O efeito direto dessa fase ϕ_{NL} é a variação no espectro do pulso. Na prática, não existe meio em que o SPM ocorra isoladamente, e sua interação com a dispersão pode fazer com que o pulso alargue ainda mais rapidamente. A Figura 5 apresenta um exemplo do espectro de um pulso gaussiano após se propagar na fibra.

Figura 5 – Efeito do alargamento espectral.



Fonte: Próprio autor.

Nesse caso, além de alargar a banda do pulso, o *chirp de frequência* também altera o formato do espectro.

2.3 Método *Split-Step Fourier*

Equações diferenciais parciais não lineares como a NLSE geralmente não possuem soluções analíticas triviais e portanto exigem métodos numéricos sofisticados. O algoritmo estado-da-arte empregado nesse caso é o método *Split-Step Fourier* (SSF) que se utiliza da transformada rápida de Fourier (FFT, de *fast Fourier transform*), conhecida por sua rapidez

no cálculo de transformadas de Fourier de sinais discretos, para calcular rapidamente os efeitos da propagação ótica (FRIGO; JOHNSON, 2005).

Para entender o método, escrevemos a Equação (2.2) na forma

$$\frac{\partial A}{\partial z} = (\hat{D} + \hat{N})A \quad (2.19)$$

onde \hat{D} é o operador das dispersões e perdas lineares que acontecem na fibra e \hat{N} é o operador não linear da propagação. Essa notação simplifica a equação de Schrödinger por meio do operador hamiltoniano, $\hat{H} = \hat{D} + \hat{N}$, que é uma medida de energia do sistema (SCHLOSS, 2018).

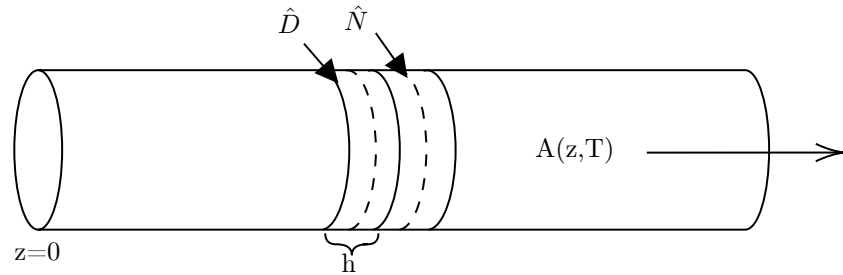
Esses operadores são dados então por

$$\hat{D} = -\frac{i\beta_2}{2} \frac{\partial^2}{\partial T^2} + \frac{\beta_3}{6} \frac{\partial^3}{\partial T^3} - \frac{\alpha}{2} \quad (2.20)$$

$$\hat{N} = i\gamma|A|^2 \quad (2.21)$$

Para o método, assumimos que dispersão e não linearidade atuam de forma independente e uma de cada vez.

Figura 6 – Esquemático do método split-step Fourier.



Fonte: Próprio autor.

Na Figura 6, a linha sólida representa a atuação independente do efeito da dispersão (e perdas), e a linha tracejada representa as não linearidades atuando sozinhas na fibra. Com isso, a propagação de z para $z + h$ acontece em dois passos, primeiro $\hat{N} = 0$ onde se avalia apenas os efeitos lineares e depois $\hat{D} = 0$ onde se avalia apenas os efeitos não lineares.

Então, a propagação $z + h$ é dada por

$$A(z + h, T) = \exp[h(\hat{D} + \hat{N})]A(z, T) \quad (2.22)$$

que podemos aproximar na expressão abaixo usando a fórmula de Baker-Hausdorff para dois operadores que não comutam (WEISS; MARADUDIN, 1962)

$$A(z + h, T) \approx \exp(h\hat{D}) \exp(h\hat{N})A(z, T) \quad (2.23)$$

O operador exponencial $\exp(h\hat{D})$ pode ser tratado no domínio de Fourier através das transformações seguintes

$$A(z+h, T) \approx \mathcal{F}^{-1} \left\{ \exp[h\hat{D}(-i\omega)] \times \mathcal{F} \left[\exp(h\hat{N})A(z, T) \right] \right\} \quad (2.24)$$

Onde \mathcal{F} e \mathcal{F}^{-1} são as transformadas direta e inversa de Fourier, respectivamente, e $\hat{D}(-i\omega)$ é obtido substituindo o operador $\partial/\partial T$ por $-i\omega$ na Equação (2.20). Transformações similares são feitas no calculo de $\exp(h\hat{N})A(z, T)$ para incluir efeitos mais complexos como *self-steepening* ou espalhamento Raman (DEITERDING et al., 2013).

O método completo, aplicando as transformações rápidas de Fourier a cada passo resulta em

$$A(L, T) \approx \left(\prod_{m=1}^M e^{h\hat{D}} e^{h\hat{N}} \right) A(0, T) \quad (2.25)$$

onde L é o comprimento da fibra, M o número total de passos calculados, e $A(0, T)$ e $A(L, T)$ o pulso ótico na entrada e saída da fibra, respectivamente.

O algoritmo SSF pode ser reescrito de forma a aumentar a eficiência computacional (tornando-o mais complexo). No Algoritmo 1 tem-se o recorte de um código-exemplo com a parte essencial do método, nele a expressão (2.25) é alterada para

$$A(L, T) \approx e^{-h\hat{N}/2} \left(\prod_{m=1}^M e^{h\hat{N}} e^{h\hat{D}} \right) e^{h\hat{N}/2} A(0, T) \quad (2.26)$$

que inverte a ordem dos operadores, e os efeitos são computados na ordem $(h/2)\hat{N} \rightarrow h\hat{D} \rightarrow (h/2)\hat{N}$, com os efeitos não lineares ocorrendo na metade do passo.

Data:

- | | | |
|---|--------------------------------|-------------------------|
| 1 | $A = \text{pulso}(T/T0);$ | ▷ Pulso ótico |
| 2 | $D = i(1/2)\beta_2\omega^2;$ | ▷ Operador de dispersão |
| 3 | $N = i\gamma \text{abs}(A)^2;$ | ▷ Operador não linear |

Start:

- ```

4 $U = A \exp(N h/2);$
5 for $m = 1:M$ do
6 $U = \mathcal{F}^{-1}\{\exp(Dh) \times \mathcal{F}(U)\};$
7 $N = i\gamma \text{abs}(U)^2;$
8 $U = U \exp(Nh);$
9 end
10 $N = i\gamma \text{abs}(U)^2;$
11 $A = U \exp(-N h/2);$

```

**Algoritmo 1:** Algoritmo SSF.

## 2.4 Redes Neurais Artificiais

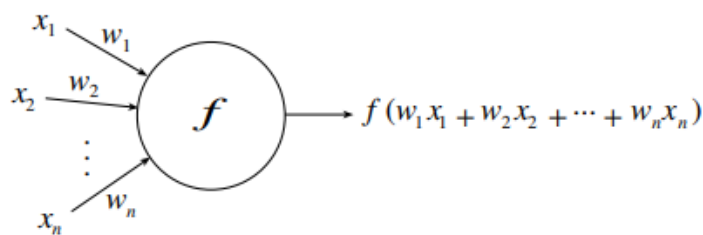
As redes neurais artificiais (RNA), ou redes neurais, são técnicas baseadas no processamento do cérebro humano. Os neurônios são capazes de transmitir, armazenar, e processar informação de maneira complexa e não totalmente compreendida (ROJAS, 1996).

Vale diferenciar dois conceitos que são comuns na literatura. O primeiro, *machine learning* (ML), mais geral, engloba o conceito de *deep learning* (DL):

- Aprendizagem de Máquina (*Machine Learning*): técnica que a partir da extração manual de características dos dados, treina-se um modelo capaz de associar padrões, classificação, regressão e fazer previsões (MATHWORKS, 2019);
- Aprendizagem Profunda (*Deep Learning*): remove-se a necessidade de pré-processamento manual dos dados, tornando o modelo mais adaptável. As redes neurais simulam o processo de aprendizagem do cérebro humano, com inúmeras camadas de neurônios artificiais e nível de abstração capaz de lidar com problemas complexos (MATHWORKS, 2019).

Nas RNAs, as funções de ativação  $f$  estão contidas nos nós (neurônios artificiais), cada entrada  $x_i$  tem um peso  $w_i$  associado, e computa-se uma única saída a partir dessas entradas ponderadas ( $\sum w_i x_i$ ), a Figura 7 exemplifica isto. A saída segue para os neurônios seguintes, se houver, de acordo com a topologia da rede neural.

Figura 7 – Nó Perceptron



Fonte: Rojas (1996).

Durante o processo de aprendizagem os pesos são atualizados automaticamente, o conhecimento da rede é armazenado nesses pesos. O aprendizado pode ser dividido entre: supervisionado ou não supervisionado.

- Aprendizado Supervisionado: baseado em conhecimento a priori, a rede neural recebe as entradas e as saídas esperadas (*targets*). O objetivo é ajustar os pesos e parâmetros

da rede e adequar a saída prevista pela rede ao *target* (VARGAS, 2021).

A cada iteração, as saídas são comparadas e sua diferença é usada para calcular o erro. Geralmente, o erro quadrático médio (MSE, de *mean squared error*) é usado para medir o desempenho da rede. Os pesos das conexões entre os neurônios são alterados gradativamente, por isso, esse processo de aprendizagem é lento e exige um grande número de dados de treinamento. O algoritmo de aprendizado supervisionado mais famoso é o *backpropagation* (ROJAS, 1996);

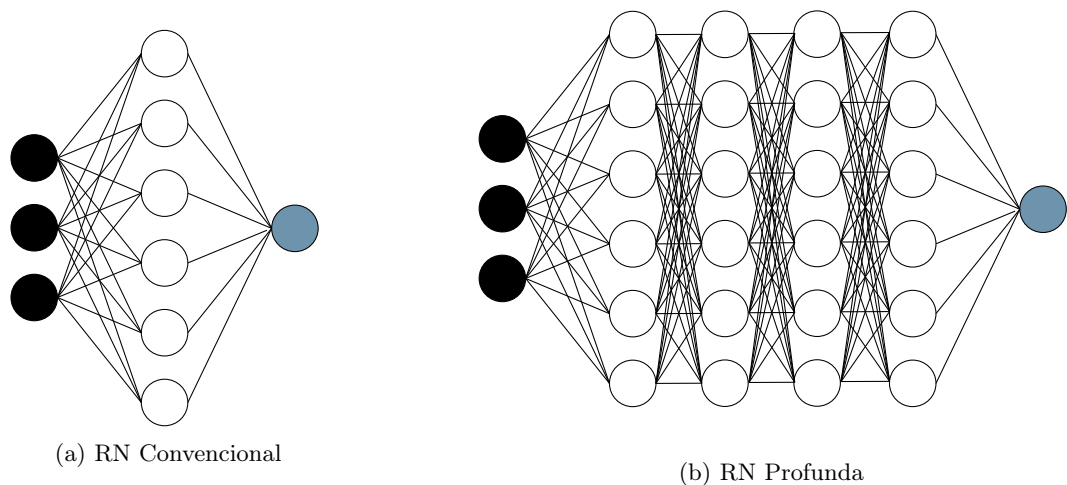
- Aprendizado Não Supervisionado (*model-free*): Não há conhecimento a priori das respostas desejadas. A cada iteração, a rede busca identificar e agrupar os padrões dos dados de entrada. O mapa de Kohonen é um exemplo típico (KOHONEN, 1982).

O presente projeto de graduação visa utilizar redes neurais artificiais supervisionadas.

### 2.4.1 Redes Neurais *Feed-Forward*

Modelo mais básico de redes neurais, usado principalmente para problemas de classificação e regressão. Destacam-se duas estruturas principais, a rede convencional, com uma única camada intermediária, e a chamada rede neural profunda, composta de inúmeras camadas intermediárias. Estas últimas tem incrível poder de processamento e preserva a simplicidade de construção da primeira, cada camada se conectando apenas na camada seguinte, sem *feedbacks*, por isso chamada de *feed-forward*.

Figura 8 – Exemplo de redes neurais clássicas.



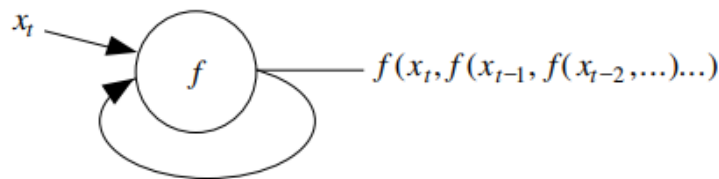
Fonte: Próprio autor.

Na Figura 8, as entradas se conectam na primeira camada (em preto), as conexões seguem as camadas intermediárias (em branco) e por fim a camada de saída (em azul).

### 2.4.2 Redes Neurais Recorrentes

Diferentemente das redes *feed-forward*, onde cada camada se conecta apenas com a próxima, as redes neurais recorrentes (RNN, de *recurrent neural networks*) apresentam *loops* de realimentação conforme visto na Figura 9. Essa estrutura de realimentação permite armazenar informações passadas e torna adequada para sequências temporais.

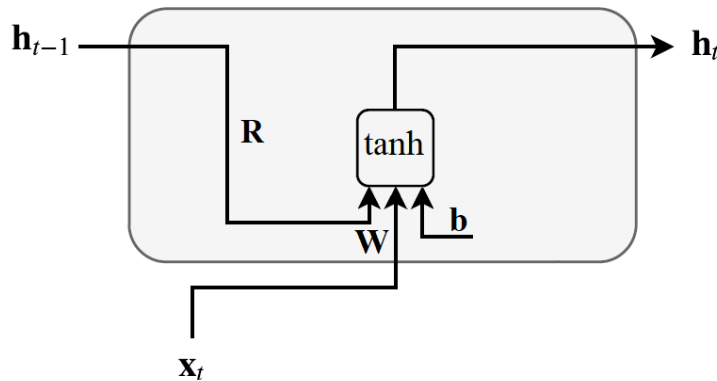
Figura 9 – Nó Recorrente.



Fonte: Rojas (1996).

O módulo interno da RNN pode ser representada como na Figura 10, onde o estado oculto anterior,  $\mathbf{h}_{t-1}$  e a entrada  $\mathbf{x}_t$  são usados no cálculo do estado oculto atual,  $\mathbf{h}_t$ . Nesse caso,  $\mathbf{h}_t$  também é a saída do módulo,  $\mathbf{h}_t = \tanh(\mathbf{R}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b})$ , onde,  $\mathbf{W}$  é o peso das entradas,  $\mathbf{R}$  é o peso do estado oculto, e  $\mathbf{b}$  o viés.

Figura 10 – Módulo interno RNN.



Fonte: Próprio autor.

O módulo RNN padrão tem simplesmente uma função de ativação tangente hiperbólico, descrita pela Equação (2.27) seguinte

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.27)$$

### 2.4.3 Redes Neurais Recorrentes *Long Short-Term Memory*

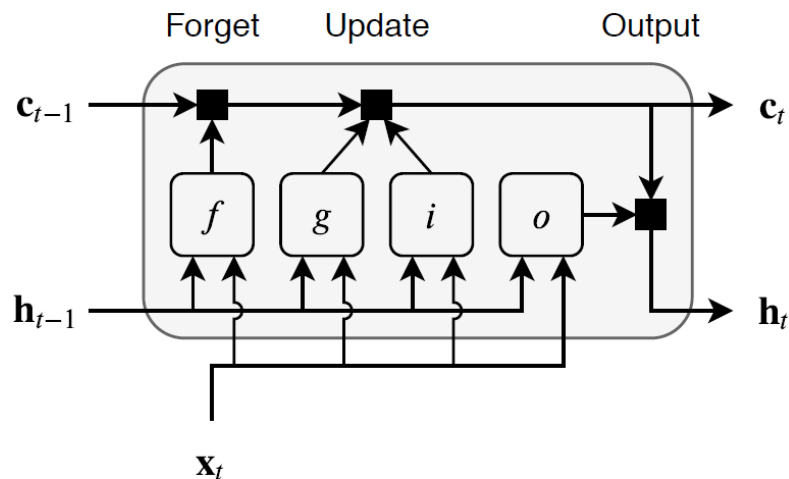
A rede neural recorrente *long short-term memory* (LSTM) foi desenvolvida por Hochreiter e Schmidhuber (1997) com o intuito de mitigar o efeito de desaparecimento de gradiente, comum em problemas dinâmicos, com dependência temporal. A LSTM resolve isso com o uso de *portões* que atualizam os estados da rede, inclusive esquecendo informações (VARGAS, 2021).

Tabela 1 – Componentes da LSTM.

| Componente                 | Função                                          |
|----------------------------|-------------------------------------------------|
| Portão de entrada (i)      | Atualiza o estado da célula                     |
| Portão de esquecimento (f) | Controla quais informações descartar ou excluir |
| Célula candidata (g)       | Adiciona informação no estado da célula         |
| Portão de saída (o)        | Decide o próximo estado                         |

Fonte: Próprio autor.

Figura 11 – Módulo interno LSTM.



Fonte: <<https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html>>.

Na Figura 11, os blocos  $f$ ,  $g$ ,  $i$  e  $o$  são o portão de esquecimento (*forget*), célula candidata (*cell candidate*), portão de entrada (*input*) e portão de saída (*output*), respectivamente. As entradas são dadas pelos vetores estado anterior da célula  $\mathbf{c}_{t-1}$ , estado oculto anterior,  $\mathbf{h}_{t-1}$  e entrada  $\mathbf{x}_t$ . Por fim, os vetores saída são o estado atual da célula  $\mathbf{c}_t$ , e o estado oculto atual,  $\mathbf{h}_t$  (VARGAS, 2021).

De maneira sucinta, a função dos blocos e o cálculo dos elementos são descritos a seguir:

Portão de entrada (*input gate*): Atualiza o estado da célula. Vamos considerar a célula candidata  $g$  nos cálculos para atualizar a célula. O estado oculto anterior  $\mathbf{h}_{t-1}$  e a entrada atual  $\mathbf{x}_t$  são passadas para os blocos  $g$  e  $i$ , seu resultado é dado pela tangente hiperbólica,

$\mathbf{g}_t = \tanh(\mathbf{h}_{t-1}, \mathbf{x}_t)$ , e sigmoide,  $\mathbf{i}_t = \sigma(\mathbf{h}_{t-1}, \mathbf{x}_t)$ . Agora, para o novo estado da célula, modificamos o estado da célula por meio de  $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$ ;

Portão de esquecimento (*forget gate*): Elimina ou mantém informações, esse é o primeiro bloco que aparece na Figura 11. O estado oculto anterior  $\mathbf{h}_{t-1}$  e a entrada atual  $\mathbf{x}_t$  são passadas no portão  $f$ , seu resultado é dado pela sigmoide  $\mathbf{f}_t = \sigma(\mathbf{h}_{t-1}, \mathbf{x}_t)$ , que posteriormente afeta o estado da célula  $\mathbf{c}_t$ ;

Portão de saída (*output gate*): Atualiza o estado oculto seguinte. O estado oculto anterior  $\mathbf{h}_{t-1}$  e a entrada atual  $\mathbf{x}_t$  são passadas no portão  $o$ , seu resultado é dado pela sigmoide  $\mathbf{o}_t = \sigma(\mathbf{h}_{t-1}, \mathbf{x}_t)$ ; O estado oculto  $\mathbf{h}_t$  é obtido a partir do estado da célula e de  $\mathbf{o}_t$ ,  $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$ .

Foram suprimidos os pesos e *biases* das funções acima. A função  $\sigma(\mathbf{h}_{t-1}, \mathbf{x}_t)$  por exemplo, é calculado como  $\sigma(\mathbf{W}\mathbf{x}_t + \mathbf{R}\mathbf{h}_{t-1} + \mathbf{b})$ . Similar para os demais cálculos. E  $\odot$  é o produto Hamadart (multiplicação elemento-a-elemento) dos vetores.

A função de ativação tangente hiperbólica é a mesma da Equação (2.27) e a função de ativação sigmoide é escrita na Equação (2.28) abaixo,

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}. \quad (2.28)$$

O resumo das equações do módulo LSTM encontra-se a seguir:

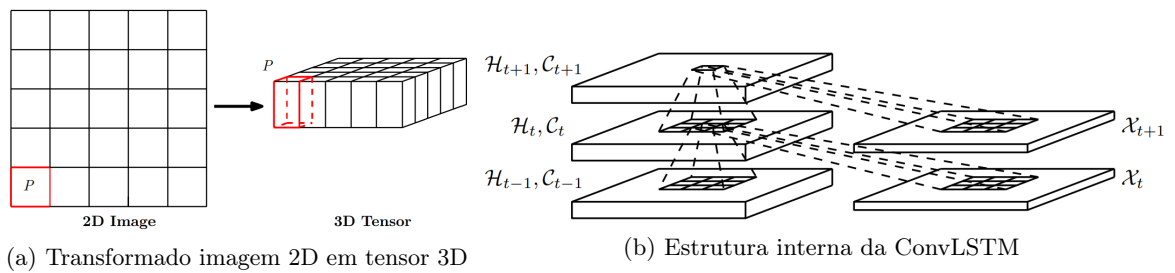
$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{R}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{R}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{R}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\ \mathbf{g}_t &= \tanh(\mathbf{W}_g \mathbf{x}_t + \mathbf{R}_g \mathbf{h}_{t-1} + \mathbf{b}_g) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \end{aligned} \quad (2.29)$$

O estado da célula  $\mathbf{c}_t$  é alterado pelos portões de esquecimento e de entrada, e o estado oculto  $\mathbf{h}_t$  apenas pelo portão de saída.

### 2.4.4 Redes Neurais Convolucionais LSTM

Baseado nos avanços das redes recorrentes, em especial das LSTMs, Shi et al. (2015) propõem uma nova rede convolucional LSTM (ConvLSTM) para previsões de chuva. Outras redes associando camadas convolucionais com LSTMs já foram usadas (SAINATH et al., 2015; DONAHUE et al., 2017), a diferença aqui está na estrutura interna da ConvLSTM que integra as características de ambas camadas (CNN e LSTM), tornando mais simples a construção da rede e permitindo capturar as correlações espaço-temporais dos dados.

Figura 12 – Modelo de rede ConvLSTM.



Fonte: (SHI et al., 2015).

As equações que modelam a rede ConvLSTM são similares aquelas da Equação (2.29), com uma mudança perceptível, a multiplicação matricial foi substituída pelo operador da convolução  $\star$ , e agora,  $\mathbf{X}_t$ ,  $\mathbf{H}_{t-1}$ ,  $\mathbf{C}_t$  e  $\mathbf{H}_t$  são matrizes, não mais vetores.

O resumo das equações do módulo ConvLSTM encontra-se a seguir:

$$\begin{aligned}
 \mathbf{i}_t &= \sigma(\mathbf{W}_i \star \mathbf{X}_t + \mathbf{R}_i \star \mathbf{H}_{t-1} + \mathbf{b}_i) \\
 \mathbf{f}_t &= \sigma(\mathbf{W}_f \star \mathbf{X}_t + \mathbf{R}_f \star \mathbf{H}_{t-1} + \mathbf{b}_f) \\
 \mathbf{o}_t &= \sigma(\mathbf{W}_o \star \mathbf{X}_t + \mathbf{R}_o \star \mathbf{H}_{t-1} + \mathbf{b}_o) \\
 \mathbf{g}_t &= \tanh(\mathbf{W}_g \star \mathbf{X}_t + \mathbf{R}_g \star \mathbf{H}_{t-1} + \mathbf{b}_g) \\
 \mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
 \mathbf{H}_t &= \mathbf{o}_t \odot \tanh(\mathbf{C}_t)
 \end{aligned} \tag{2.30}$$

A inclusão dessa rede como uma alternativa a LSTM permite uma comparação mais completa do problema de propagação, avaliando não apenas a mudança temporal, mas incluindo também as correlações espaciais do pulso na fibra.



### 3 METODOLOGIA E ETAPAS DE DESENVOLVIMENTO

Segundo [Prodanov e Freitas \(2013\)](#), o trabalho pode ser classificado como segue. Quanto a sua natureza, trata-se de um trabalho de pesquisa aplicada, visando a solução de um problema específico na área de fotônica.

Tem como objetivo uma pesquisa exploratória através de um procedimento técnico de pesquisa experimental, buscando melhorias no desempenho alcançado com modelos numéricos, empregando no lugar deste, modelos neurais.

Para isso, a abordagem é inteiramente quantitativa, e a comparação das melhorias é feita através de métricas quantificáveis como erro quadrático e tempo de cálculo dos modelos.

#### 3.1 Modelagem da Fibra

O algoritmo SSF será desenvolvido no [MATLAB \(2021\)](#) em razão de sua interface simples e intuitiva, e pelo acesso na literatura de códigos numéricos já implementados para o *software* [\(AGRAWAL, 2007\)](#). Os parâmetros da simulação são adequados ao problema de propagação em uma fibra de 13 metros, altamente não linear. Seus coeficientes são  $\gamma = 18,4 \times 10^{-3} \text{ W}^{-1}\text{m}^{-1}$ ,  $\beta_2 = -5,23 \times 10^{-27} \text{ s}^2\text{m}^{-1}$  (regime de dispersão anômalo) e  $\beta_3 = 4,27 \times 10^{-41} \text{ s}^3\text{m}^{-1}$ , similares ao usado por [Salmela et al. \(2021\)](#) para realizar uma compressão de soliton de ordem superior. Esses parâmetros estão na Tabela 2. O pulso tem um perfil de secante hiperbólico e com potência,  $P_0$ , e largura de pulso,  $T_0$ , variáveis.

Tabela 2 – Parâmetros da fibra.

| Parâmetro            | Valor                   | Unidade                      |
|----------------------|-------------------------|------------------------------|
| $L$                  | 13                      | m                            |
| $\alpha_{\text{dB}}$ | 0,05                    | dB/km                        |
| $\gamma$             | $18,4 \times 10^{-3}$   | $\text{W}^{-1}\text{m}^{-1}$ |
| $\beta_2$            | $-5,23 \times 10^{-27}$ | $\text{s}^2\text{m}^{-1}$    |
| $\beta_3$            | $4,27 \times 10^{-41}$  | $\text{s}^3\text{m}^{-1}$    |

Fonte: Próprio autor.

Trata-se de um caso extremo e particular, com uma dinâmica não linear complexa, mas que pode ser encontrado no contexto de geração de lasers pulsados, por exemplo. A escolha desse caso extremo está relacionado ao fato de serem os mais difíceis de modelar e que,

portanto, tornam o modelo numérico mais complexo, onde a não linearidade tem maior efeito.

Repetindo aqui a Equação (2.2) da propagação de um pulso ótico ultra curto na fibra:

$$i \frac{\partial A}{\partial z} = \frac{\beta_2}{2} \frac{\partial^2 A}{\partial T^2} + i \frac{\beta_3}{6} \frac{\partial^3 A}{\partial T^3} - i \frac{\alpha}{2} A - \gamma |A|^2 A$$

Como vamos tratar da propagação do pulso com amplitude normalizada  $U(z, \tau)$ , temos

$$A(z, \tau) = \sqrt{P_0} U(z, \tau) \quad (3.1)$$

então

$$i \frac{\partial U}{\partial z} = \frac{\hat{\beta}_2}{2} \frac{\partial^2 U}{\partial \tau^2} + i \frac{\hat{\beta}_3}{6} \frac{\partial^3 U}{\partial \tau^3} - i \frac{\alpha}{2} U - \hat{\gamma} |U|^2 U \quad (3.2)$$

onde  $\tau = T/T_0$ ,

$$\hat{\beta}_2 = \frac{\text{sgn}(\beta_2)}{L_D}, \quad \hat{\beta}_3 = \frac{\text{sgn}(\beta_3)}{L'_D}, \quad \hat{\gamma} = \frac{1}{L_{NL}} \quad (3.3)$$

e  $s = \text{sgn}(\beta_2) = \pm 1$  dependendo se a dispersão é normal ( $s = 1$ ) ou anômala ( $s = -1$ ), e  $\hat{\beta}_2$ ,  $\hat{\beta}_3$  e  $\hat{\gamma}$  são os termos parametrizados pelos comprimentos abaixo

$$L_D = \frac{T_0^2}{|\beta_2|}, \quad L'_D = \frac{T_0^3}{|\beta_3|}, \quad L_{NL} = \frac{1}{\gamma P_0}, \quad (3.4)$$

Aqui os termos da Equação (3.4), comprimento da dispersão,  $L_D$ , comprimento da dispersão associada ao TOD,  $L'_D$ , e o comprimento não linear,  $L_{NL}$  nada mais são que escalas de comprimento. Sua relação permite estimar se o regime é dominado principalmente pela dispersão ( $L_D/L_{NL} \ll 1$ ) ou pelas não linearidades ( $L_D/L_{NL} \gg 1$ ).

Por exemplo, a relação  $L_D/L_{NL}$  da propagação vista na Figura 13 de um soliton com  $P_0 = 26,3$  W e  $T_{\text{FWHM}} = 1,1$  ps pode ser calculado como a seguir

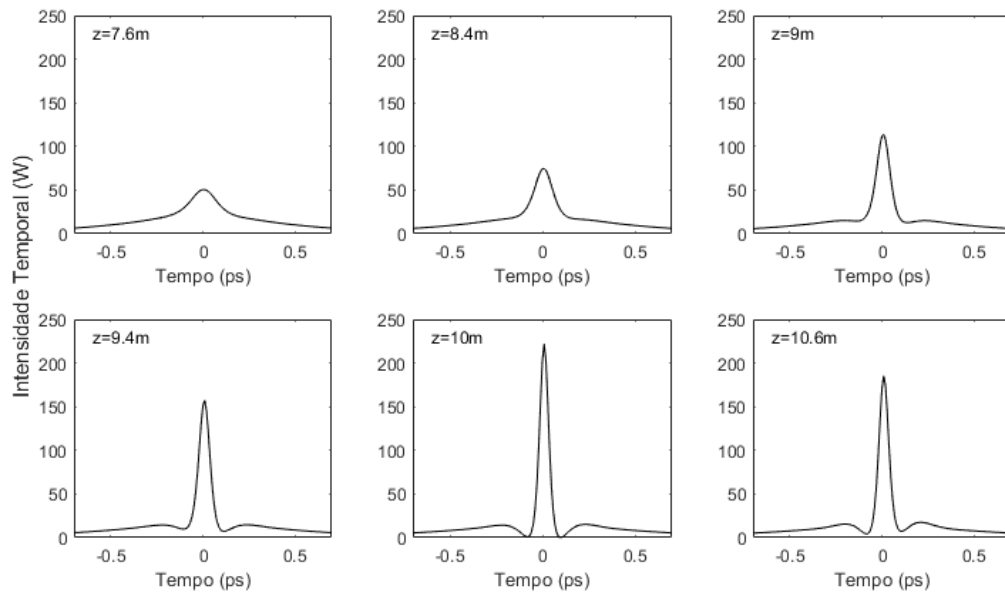
$$\frac{L_D}{L_{NL}} = \frac{\gamma P_0 T_0^2}{|\beta_2|} = 36,02 \gg 1 \quad (3.5)$$

ou ainda, a relação  $L_D/L_{NL}$  da propagação da Figura 14 de um soliton com  $P_0 = 34,2$  W e  $T_{\text{FWHM}} = 0,77$  ps pode ser dado pela Equação (3.6) seguinte

$$\frac{L_D}{L_{NL}} = \frac{\gamma P_0 T_0^2}{|\beta_2|} = 22,95 \gg 1 \quad (3.6)$$

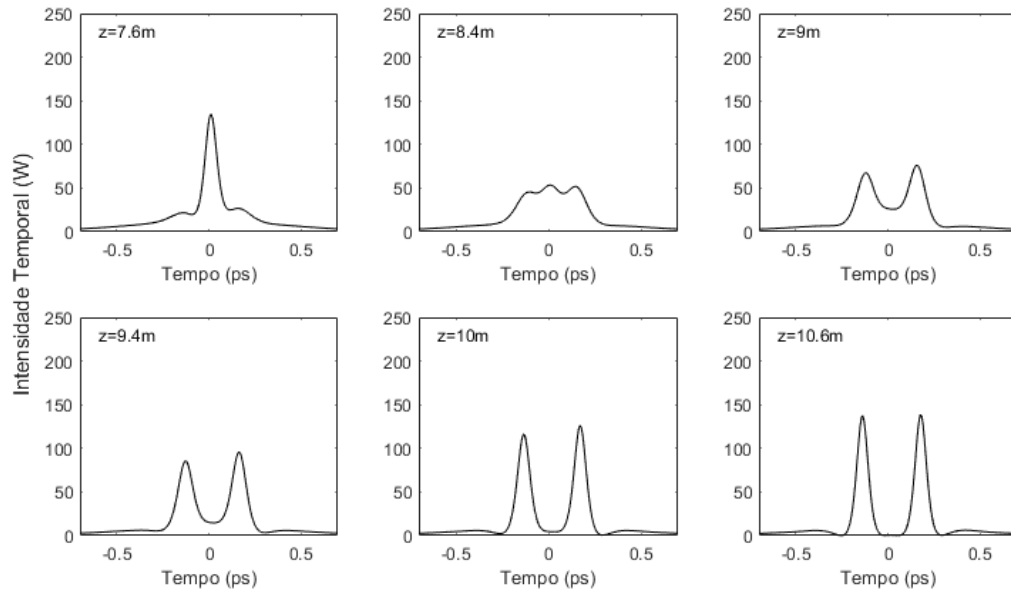
Comparando a dinâmica das Figuras 13 e 14 produzidas com o algoritmo SSF, é possível perceber a diferença radical na propagação de cada pulso. Portanto, trata-se de um caso extremo e particular, com um regime predominantemente não linear. A rede neural vai ser testada para esse caso extremo, e pode ser reproduzida e adaptada futuramente para o contexto de telecomunicações (tipicamente um regime dispersivo).

Figura 13 – Intensidade temporal para diferentes distâncias de um soliton de ordem superior com  $P_0 = 26,3 \text{ W}$  e  $T_{FWHM} = 1,1 \text{ ps}$ .



Fonte: Adaptado de [Salmela et al. \(2021\)](#).

Figura 14 – Intensidade temporal para diferentes distâncias de um soliton de ordem superior com  $P_0 = 34,19 \text{ W}$  e  $T_{FWHM} = 0,77 \text{ ps}$ .



Fonte: Próprio autor.

### 3.2 Algoritmo *Split-Step*

A evolução do pulso é sensível ao meio de propagação e as suas condições iniciais. A potência de pico na entrada da fibra varia no intervalo de 18,41 a 34,19 W, e a duração  $T_{\text{FWHM}}$  varia no intervalo de 0,77 a 1,43 ps (mapeando  $T_0$  no intervalo de 0,44 a 0,81 ps), o que caracteriza um soliton de ordem  $N$  variando de 3,5 a 8,9 (SALMELA et al., 2021).

Com um conjunto de 3.000 ( $40 \text{ powers} \times 75 \text{ widths}$ ) condições iniciais distintas, o objetivo é gerar uma base de dados com o perfil temporal de cada passo dado dentro do algoritmo SSF. Por exemplo, escolhendo um número de 100 passos no interior da fibra ( $\Delta z = 0,13 \text{ m}$ ) e o pulso amostrado com 256 pontos, o resultado é uma matriz tridimensional com  $7,68 \times 10^7$  elementos ( $100 \times 256 \times 3000$ ). Um conjunto dessa magnitude favorece o aprendizado da rede neural, mas é custoso computacionalmente, e torna o processo mais lento. Outra escolha possível é amostrar o pulso com 128 pontos, nesse caso o resultado é de  $3,84 \times 10^7$  elementos ( $100 \times 128 \times 3000$ ). Usaremos essa última.

Com os valores da fibra apresentados na Tabela 2, parametrizamos eles de acordo com a potência e comprimento do pulso, e seguimos os cálculos dos hamiltonianos como descrito na amostra de código abaixo. Esse código executa aquilo já explicado no Algoritmo 1 da Seção 2.3. O resultado é a matriz de propagação, onde cada linha representa o pulso em um passo  $\Delta z$  da propagação.

Listing 3.1 – Código *Split-Step*.

```

1 % (w-w0) -> i(d/dt)
2 D = 1i*beta2*omega*.^2/factorial(2);
3 D = 1i*beta3*omega*.^2/factorial(3) + D;
4 D = D - alpha/2;
5
6 D = exp(D*deltaz); % Hamiltoniano D
7 NL = 1i*gamma*deltaz; % Hamiltoniano N
8 U = sech(tau); % Pulso
9
10 function A = fiber(U,M,D,GAMMA,OMEGA)
11 NL = abs(U).^2;
12 A = U.*exp(NL.*GAMMA/2);
13
14 for n=1:M
15 % Propaga na fibra:
16 U = fft(D.*ifft(A));

```

```

17
18 NL = abs(U).^2;
19 A = U.*exp(NL.*GAMMA);
20 end
21
22 NL = abs(U).^2;
23 A = A.*exp(-NL.*GAMMA/2); % Final field
24 end
25
26 U = fiber(U,M,D,NL,omega)
27 A = sqrt(P0)*U;

```

O algoritmo completo pode ser encontrado nos Anexos [A](#), [B](#) e [C](#), ou em <https://github.com/gustavo-r-martins/TCC>.

Tabela 3 – Dados gerados.

|           | <b>0</b> | <b>1</b> | <b>2</b> | <b>3</b> | ... | <b>124</b> | <b>125</b> | <b>126</b> | <b>127</b> |
|-----------|----------|----------|----------|----------|-----|------------|------------|------------|------------|
| <b>0</b>  | 2,7585   | 2,8886   | 3,0242   | 3,1655   | ... | 3,3127     | 3,1655     | 3,0241     | 2,8885     |
| <b>1</b>  | 2,9230   | 2,9532   | 3,0002   | 3,0990   | ... | 3,2821     | 3,0930     | 2,9847     | 2,9488     |
| <b>2</b>  | 2,9937   | 3,0184   | 3,0487   | 3,1204   | ... | 3,2186     | 3,1061     | 3,0426     | 3,0115     |
| <b>3</b>  | 3,0466   | 3,0661   | 3,0924   | 3,1524   | ... | 3,2244     | 3,1360     | 3,0864     | 3,0594     |
| <b>4</b>  | 3,0912   | 3,1079   | 3,1306   | 3,1830   | ... | 3,2416     | 3,1686     | 3,1256     | 3,1000     |
| ...       | ...      | ...      | ...      | ...      | ... | ...        | ...        | ...        | ...        |
| <b>95</b> | 2,3747   | 2,3804   | 2,3767   | 2,3763   | ... | 2,4369     | 2,4093     | 2,3791     | 2,3687     |
| <b>96</b> | 2,3688   | 2,3700   | 2,3581   | 2,3657   | ... | 2,4208     | 2,3874     | 2,3693     | 2,3647     |
| <b>97</b> | 2,3473   | 2,3450   | 2,3374   | 2,3417   | ... | 2,4220     | 2,3968     | 2,3814     | 2,3634     |
| <b>98</b> | 2,3357   | 2,3312   | 2,3227   | 2,3254   | ... | 2,4392     | 2,4165     | 2,3837     | 2,3532     |
| <b>99</b> | 2,3391   | 2,3192   | 2,3054   | 2,3139   | ... | 2,4229     | 2,4185     | 2,4037     | 2,3729     |

100 linhas × 128 colunas

Fonte: Próprio autor.

Esses dados mostrados na Tabela [3](#) são de uma condição inicial apenas. O mesmo acontece para 3.000 condições iniciais ao todo.

### 3.3 Tratamento dos Dados

Com os dados gerados, devemos agora normalizá-los, transformar a série temporal em um problema de regressão e por fim adequar a estrutura dos dados de acordo com a entrada aceita pela rede neural.

### 3.3.1 Normalização

A normalização é um passo fundamental (ou mesmo compulsório) para o treinamento de uma rede neural, pois acelera seu aprendizado. A normalização, descrita pela Equação (3.7) a seguir, mapeia os dados gerados em um intervalo entre 0 (mínimo) e 1 (máximo).

$$\mathbf{X}_{\text{std}} = \frac{\mathbf{X} - X_{\min}}{X_{\max} - X_{\min}} \quad (3.7)$$

Aplicando a normalização na base de dados (na base inteira, não apenas naqueles mostrados na Tabela 3), obtemos os valores mapeados na Tabela 4 abaixo.

Tabela 4 – Dados normalizados.

|           | <b>0</b> | <b>1</b> | <b>2</b> | <b>3</b> | ... | <b>124</b> | <b>125</b> | <b>126</b> | <b>127</b> |
|-----------|----------|----------|----------|----------|-----|------------|------------|------------|------------|
| <b>0</b>  | 0,0071   | 0,0075   | 0,0078   | 0,0082   | ... | 0,0086     | 0,0082     | 0,0078     | 0,0075     |
| <b>1</b>  | 0,0076   | 0,0076   | 0,0078   | 0,0080   | ... | 0,0085     | 0,0080     | 0,0077     | 0,0076     |
| <b>2</b>  | 0,0078   | 0,0078   | 0,0079   | 0,0081   | ... | 0,0083     | 0,0080     | 0,0079     | 0,0078     |
| <b>3</b>  | 0,0079   | 0,0079   | 0,0080   | 0,0082   | ... | 0,0084     | 0,0081     | 0,0080     | 0,0079     |
| <b>4</b>  | 0,0080   | 0,0080   | 0,0081   | 0,0082   | ... | 0,0084     | 0,0082     | 0,0081     | 0,0080     |
| ...       | ...      | ...      | ...      | ...      | ... | ...        | ...        | ...        | ...        |
| <b>95</b> | 0,0061   | 0,0062   | 0,0061   | 0,0061   | ... | 0,0063     | 0,0062     | 0,0062     | 0,0061     |
| <b>96</b> | 0,0061   | 0,0061   | 0,0061   | 0,0061   | ... | 0,0063     | 0,0062     | 0,0061     | 0,0061     |
| <b>97</b> | 0,0061   | 0,0061   | 0,0060   | 0,0061   | ... | 0,0063     | 0,0062     | 0,0062     | 0,0061     |
| <b>98</b> | 0,0060   | 0,0060   | 0,0060   | 0,0060   | ... | 0,0063     | 0,0062     | 0,0062     | 0,0061     |
| <b>99</b> | 0,0060   | 0,0060   | 0,0060   | 0,0060   | ... | 0,0063     | 0,0063     | 0,0062     | 0,0061     |

100 linhas × 128 colunas

Fonte: Próprio autor.

### 3.3.2 Série Temporal × Regressão

O passo seguinte é transformar o problema atual de uma série temporal, onde cada atributo (colunas) evolui no tempo (linhas) de maneira sequencial, em um problema de regressão, dado uma sequência de observações, buscar inferir seu próximo valor.

Na Tabela 5 fica claro o que buscamos fazer, com um horizonte igual a 3, usamos 3 observações (no *box* vermelho) para predizer uma quarta observação (no *box* preto). O problema de regressão aumenta a quantidade total de elementos da base de dados de  $100 \times 128$  nesse caso em  $4 \times (100 - 3) \times 128$ , quase 4 vezes maior. A escolha do horizonte afeta diretamente na base de dados e no treinamento.

Tabela 5 – Visualizando problema de regressão nos dados.

|     | 0      | 1      | 2      | 3      | ... | 124    | 125    | 126    | 127    |
|-----|--------|--------|--------|--------|-----|--------|--------|--------|--------|
| 0   | 0,0071 | 0,0075 | 0,0078 | 0,0082 | ... | 0,0086 | 0,0082 | 0,0078 | 0,0075 |
| 1   | 0,0076 | 0,0076 | 0,0078 | 0,0080 | ... | 0,0085 | 0,0080 | 0,0077 | 0,0076 |
| 2   | 0,0078 | 0,0078 | 0,0079 | 0,0081 | ... | 0,0083 | 0,0080 | 0,0079 | 0,0078 |
| 3   | 0,0079 | 0,0079 | 0,0080 | 0,0082 | ... | 0,0084 | 0,0081 | 0,0080 | 0,0079 |
| 4   | 0,0080 | 0,0080 | 0,0081 | 0,0082 | ... | 0,0084 | 0,0082 | 0,0081 | 0,0080 |
| ... | ...    | ...    | ...    | ...    | ... | ...    | ...    | ...    | ...    |
| 95  | 0,0061 | 0,0062 | 0,0061 | 0,0061 | ... | 0,0063 | 0,0062 | 0,0062 | 0,0061 |
| 96  | 0,0061 | 0,0061 | 0,0061 | 0,0061 | ... | 0,0063 | 0,0062 | 0,0061 | 0,0061 |
| 97  | 0,0061 | 0,0061 | 0,0060 | 0,0061 | ... | 0,0063 | 0,0062 | 0,0062 | 0,0061 |
| 98  | 0,0060 | 0,0060 | 0,0060 | 0,0060 | ... | 0,0063 | 0,0062 | 0,0062 | 0,0061 |
| 99  | 0,0060 | 0,0060 | 0,0060 | 0,0060 | ... | 0,0063 | 0,0063 | 0,0062 | 0,0061 |

100 linhas × 128 colunas

Fonte: Próprio autor.

### 3.3.3 Dimensionamento dos dados

O LSTM espera uma entrada com três dimensões:  $[batch, timesteps, feature]$  (BROWNLEE, 2017a; CHOLLET et al., 2015a). Mesmo para entradas 2D, isto é, com apenas um atributo sendo usado para realizar a predição, a amostra deve ter formato 3D,  $[batch, timesteps, 1]$ .

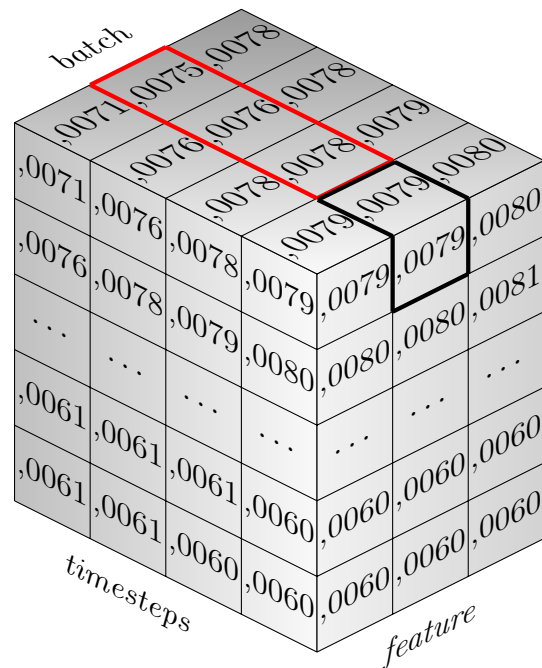
Cada termo pode ser descrito como segue:

- **batch**: A quantidade de dados de entrada. Tipicamente as linhas de um *dataset*.
- **timesteps**: Variação temporal de uma variável (atributo) na amostra observada.
- **feature**: Diferentes variáveis observadas em uma amostra.

Adequando o *dataset* da Tabela 5, obtemos a matriz da Figura 15. É possível reconhecer os dados das primeiras 3 colunas usadas para exemplificar. Cada linha da matriz define uma amostra, cada coluna é uma variação temporal, ou *timestep* dos atributos, e no terceiro eixo, ortogonal aos outros dois, estão as *features* das amostras. Essa matriz vai servir como banco de dados para a rede construída no *Colaboratory* (GOOGLE, 2021).

Já a rede ConvLSTM espera uma entrada em cinco dimensões:  $[samples, time, rows, cols, channels]$  (CHOLLET et al., 2015b). Para o nosso caso, temos  $[samples, 1, rows, cols, 1]$ .

Figura 15 – Matriz 3D dos Dados.



Fonte: Próprio autor.

## 3.4 Rede Neural

Esta seção descreve a construção da rede, a etapa de treinamento, predição, avaliação dos resultados através das métricas de desempenho, e por fim uma comparação das redes testadas e seus desempenhos.

### 3.4.1 Construção

A construção da rede neural é empírica. Dada a natureza do problema de evolução temporal, faremos o uso da *long short-term memory* nas camadas da rede. Outros módulos também serão usados como *dropout* e *dense* para compor a estrutura do modelo neural. A arquitetura completa e seus parâmetros (*batch size*, número de épocas, método de otimização, entre outros) serão testados durante a execução do trabalho.

- **Otimizador:** Algoritmo implementado para acelerar o processo de treinamento.
- **Função de Ativação:** Função de ativação usada na camada de saída da rede.
- **Epochs:** A quantidade de vezes que os dados são completamente treinados.
- **Batch Size:** Tamanho do lote passado por vez, visto que o *dataset* é muito grande.



Tabela 6 – Parâmetros de rede e treinamento.

| Parâmetro          |          |
|--------------------|----------|
| Otimizador         | RMSProp* |
| Função de ativação | 'tanh'   |
| <i>Epochs</i>      | 60       |
| <i>Batch Size</i>  | 128      |

\* RMSProp - *Root Mean Square Propagation*.

Fonte: Próprio autor.

A Tabela 6 mostra os parâmetros de rede e seus valores. A escolha do número de épocas (*epochs*) e tamanho do lote (*batches*) a princípio, para a comparação das redes a serem testadas, foi fixada em 60 e 128, respectivamente, para que seja possível realizar uma variedade de testes de modelos de redes neurais.

A seguir temos as arquiteturas das redes LSTM e ConvLSTM construídas:

Listing 3.2 – Código com a construção da rede LSTM.

```

1 def build_model(inputs, output_size, neurons, actv_func=actv_func,
2 dropout=dropout, loss=loss, optimizer=optimizer):
3 model = Sequential()
4 model.add(LSTM(256, return_sequences=0, activation=actv_func,
5 input_shape=(inputs.shape[1], inputs.shape[2])))
6 model.add(Dropout(0.25))
7 model.add(Dense(units=output_size))
8 model.add(Dense(units=output_size))
9 model.add(Activation(actv_func))
10 model.compile(loss=loss, optimizer=optimizer, metrics=['mae'])
11 model.summary()
12 return model

```

Tabela 7 – Arquitetura da rede LSTM.

| Layer                        | Output Shape | # Param |
|------------------------------|--------------|---------|
| LSTM                         | (None, 256)  | 394.240 |
| Dropout                      | (None, 256)  | 0       |
| Dense (1)                    | (None, 128)  | 32.896  |
| Dense (2)                    | (None, 128)  | 16.512  |
| Activation                   | (None, 128)  | 0       |
| <i>Total Params: 443.648</i> |              |         |

Fonte: Próprio autor.

Na arquitetura da rede LSTM apresentada no código acima e na Tabela 7 instanciamos o

modelo *Sequential()*, dessa forma é possível passar uma lista de camadas em sequência. A camada LSTM é instanciada primeiro e por isso também recebe como parâmetro o formato dos dados de entrada, *input\_shape*. Essa camada tem como função de ativação a “tanh”. Após cada camada LSTM, é instanciada a camada *Dropout*. Essa técnica visa desativar parte (25%) dos neurônios da LSTM de forma aleatória durante a etapa de treinamento, com isto, previne-se *overfitting* da rede. Por fim, temos duas camadas densas (ativação linear) e a camada de saída.

E abaixo tem-se a rede ConvLSTM:

Listing 3.3 – Código com a construção da rede ConvLSTM.

```

1 def build_model(inputs , output_size , neurons , actv_func=actv_func ,
2 dropout=dropout , loss=loss , optimizer=optimizer):
3 model = Sequential ()
4 model.add(ConvLSTM2D(filters =12, kernel_size =(3,3) ,
5 padding='same' , input_shape=inputs .shape [1:]))
6 model.add(Flatten ())
7 model.add(Dropout (0.25))
8 model.add(Dense (units=output_size))
9 model.add(Dense (units=output_size))
10 model.add(Activation (actv_func))
11 model.compile(loss=loss , optimizer=optimizer , metrics=['mae'])
12 model.summary ()
13 return model

```

Tabela 8 – Arquitetura da rede ConvLSTM2D.

| <i>Layer</i>                   | <i>Output Shape</i> | # Param |
|--------------------------------|---------------------|---------|
| ConvLSTM2D                     | (None, 5, 128, 12)  | 5.664   |
| <i>Flatten</i>                 | (None, 7680)        | 0       |
| <i>Dropout</i>                 | (None, 7680)        | 0       |
| <i>Dense</i> (1)               | (None, 128)         | 983.168 |
| <i>Dense</i> (2)               | (None, 128)         | 16.512  |
| <i>Activation</i>              | (None, 128)         | 0       |
| <i>Total Params: 1.005.344</i> |                     |         |

Fonte: Próprio autor.

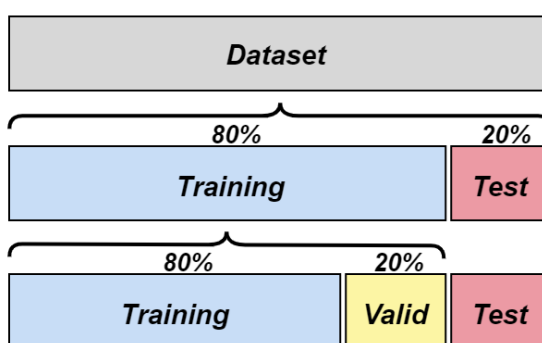
A arquitetura da rede ConvLSTM é similar ao LSTM apresentado antes. Instanciamos inicialmente o modelo *Sequential()*. A camada ConvLSTM2D recebe os tensores de entrada com formato 5D. Os parâmetros dessa camada são, principalmente, *filters*, *kernel\_size* e *padding*. *Filters* é número de filtros de saída na convolução, o tamanho do *kernel* define o tamanho do filtro da convolução, e o *padding = 'same'* define a “imagem” de saída com o

mesmo tamanho da entrada. Essa camada tem como função de ativação a “tanh”. Após cada camada ConvLSTM, é instanciada a camada *Flatten*, que “vetoriza” a matriz de saída. As camadas posteriores já foram explicadas.

### 3.4.2 Treinamento

Os dados são divididos em três *datasets*: 80% dados de treino e validação, e 20% dados de teste. A divisão é mostrada na Figura 16. A porcentagem exibe valores típicos.

Figura 16 – Divisões do *dataset*.



Fonte: Próprio autor.

A comparação, durante o treinamento, do *fit* dos dados de treino e validação permite avaliar se há um subajuste (*underfitting*) das observações, ou ainda se o modelo está “copiando” os dados de entrada, ou seja, um sobreajuste (*overfitting*). Os dados de teste não são usados durante o treinamento, de modo que é possível verificar a capacidade da rede neural fazer generalizações, avaliando seu aprendizado em cima desses dados “novos”.

Para realizar essa comparação do *fit* da rede treinada, podemos analisar graficamente seu *log* (registro) de treinamento. No Keras, isso é feito acessando a variável *history* do método *fit()*, que armazena a métrica de custo *loss* ao final de cada época. Uma boa performance acontece quando as curvas de treino e validação decrescem e se aproximam (BROWNLEE, 2017b).

### 3.4.3 Métricas

Com a rede treinada, os dados de teste (não usados durante o treinamento) são passados para a rede, que deve fazer a predição das respostas. É com essas predições que será calculado o desempenho do modelo.

A seguir temos um conjunto de métricas usadas para avaliação dos modelos:

As duas equações mais recorrentes na literatura são o erro quadrático médio (MSE) e sua raiz (RMSE), descrito pela Equação (3.8) e Equação (3.9) abaixo

$$\text{MSE} = \frac{1}{N} \sum_{m=1}^N (x_m - \hat{x}_m)^2 \quad (3.8)$$

$$\text{RMSE} = \sqrt{\text{MSE}} \quad (3.9)$$

onde  $x_m$  é o valor obtido pela SSF e  $\hat{x}_m$  o valor obtido pela rede para cada previsão  $m$ .

Uma outra métrica, similar ao erro quadrático, é o erro absoluto médio. Descrito pela Equação (3.10), temos

$$\text{MAE} = \frac{1}{N} \sum_{m=1}^N |x_m - \hat{x}_m| \quad (3.10)$$

Para os índices acima, quanto menor, melhor o ajuste. De outra forma, para o coeficiente de determinação  $R^2$  descrito na Equação (3.11) a seguir, que mede a redução da variabilidade total da saída,  $0 \leq R^2 \leq 1$ , queremos que  $R^2$  se aproxime de 1,

$$R^2 = 1 - \frac{\sum_{m=1}^N (x_m - \hat{x}_m)^2}{\sum_{m=1}^N (x_m - \bar{x})^2} \quad (3.11)$$

sendo que  $\bar{x}$  denota a média dos valores de  $x_m$ .

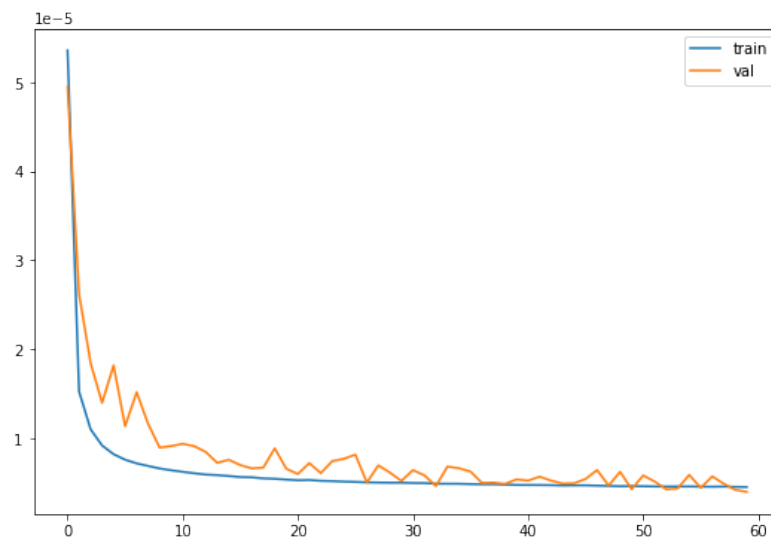
## 4 RESULTADOS

Neste capítulo são discutidos os resultados obtidos com a rede LSTM e ConvLSTM, seu comportamento durante a etapa de treinamento, as arquiteturas avaliadas para o problema, a escolha do melhor modelo e por fim, sua comparação através das métricas de desempenho descritas anteriormente. Ainda, as redes foram submetidas a novos dados, fora do *range* para o qual foram construídas, de modo a avaliar sua adaptação nesta nova situação.

### 4.1 Treinamento e Redes Construídas

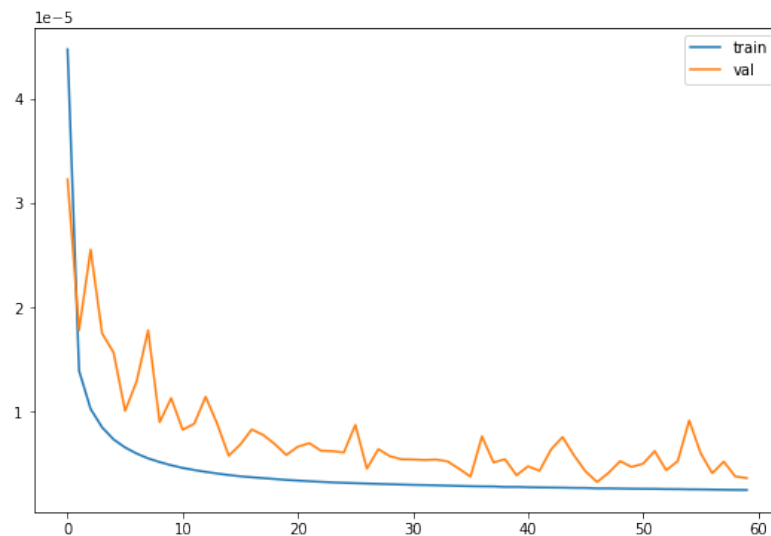
O gráfico do *log* de treinamento dos modelos LSTM e ConvLSTM podem ser vistos na Figura 17 e Figura 18, respectivamente. Nos dois casos as curvas de treino e validação iniciam com valores de *loss* elevados – a métrica usada foi a MSE, que caem rapidamente logo após as primeiras épocas, ambas se aproximando ao longo do processo de treinamento. Nas figuras, ocorrem perturbações (afastamento das curvas devido ao *overfitting*) no conjunto de validação ao longo das épocas, mais perceptível na Figura 18, mas a curva se mantém próxima do conjunto de treinamento.

Figura 17 – *Loss* durante o treinamento da rede LSTM.



Fonte: Próprio autor.

Figura 18 – Loss durante o treinamento da rede ConvLSTM.



Fonte: Próprio autor.

Foram construídas, na etapa de treinamento, as redes exibidas na Tabela 9 variando sua estrutura e parâmetros indicados na tabela. As métricas dispostas na tabela foram comparadas a fim de identificar o valor ótimo para o problema, a melhor arquitetura é descrita na seção 4.2 a seguir.

Os códigos usados para as redes neurais estão disponíveis no Anexo D e no [link <https://drive.google.com/drive/folders/11fhEmXPAAo6izv\\_PJohSwBxEwZGN8LJ2?usp=sharing>](https://drive.google.com/drive/folders/11fhEmXPAAo6izv_PJohSwBxEwZGN8LJ2?usp=sharing).

Tabela 9 – Comparação das redes construídas.

| Rede Neural | $ts^*$ | RMSE     | MSE      | MAE      | R <sup>2</sup> |
|-------------|--------|----------|----------|----------|----------------|
| LSTM-1DNN   | 1      | 0,39685% | 0,00274% | 0,20612% | 97,16265%      |
|             | 3      | 0,36357% | 0,00243% | 0,20734% | 97,54471%      |
|             | 5      | 0,37247% | 0,00245% | 0,19835% | 97,77380%      |
|             | 7      | 0,37822% | 0,00266% | 0,20702% | 97,34842%      |
|             | 10     | 0,40549% | 0,00291% | 0,20932% | 97,08891%      |
| LSTM-2DNN   | 1      | 0,37984% | 0,00259% | 0,20404% | 97,66358%      |
|             | 3      | 0,38025% | 0,00279% | 0,21674% | 97,28087%      |
|             | 5      | 0,40194% | 0,00292% | 0,21624% | 97,26246%      |
|             | 7      | 0,42929% | 0,00334% | 0,25596% | 96,61900%      |
|             | 10     | 0,40210% | 0,00296% | 0,20761% | 97,35190%      |
| LSTM-3DNN   | 1      | 0,41614% | 0,00297% | 0,21097% | 96,76408%      |
|             | 3      | 0,44371% | 0,00317% | 0,27293% | 96,05296%      |
|             | 5      | 0,39985% | 0,00290% | 0,21078% | 97,06101%      |
|             | 7      | 0,46294% | 0,00354% | 0,26165% | 96,52167%      |
|             | 10     | 0,43972% | 0,00384% | 0,23339% | 97,62067%      |

|                                          |   |          |          |          |           |
|------------------------------------------|---|----------|----------|----------|-----------|
| <b>ConvLSTM1D</b><br><i>Filters = 12</i> | 1 | 0,74491% | 0,01043% | 0,34748% | 91,50260% |
|                                          | 3 | 0,60102% | 0,00893% | 0,31951% | 96,47177% |
|                                          | 5 | 0,50187% | 0,00527% | 0,28779% | 96,87501% |
| <b>ConvLSTM1D</b><br><i>Filters = 16</i> | 1 | 0,51136% | 0,00426% | 0,32112% | 95,21182% |
|                                          | 3 | 0,50833% | 0,00386% | 0,28783% | 95,76006% |
|                                          | 5 | -        | -        | -        | -         |
| <b>ConvLSTM1D</b><br><i>Filters = 20</i> | 1 | 0,48925% | 0,00458% | 0,27183% | 96,55981% |
|                                          | 3 | 0,46801% | 0,00422% | 0,25676% | 97,21294% |
|                                          | 5 | -        | -        | -        | -         |
| <b>ConvLSTM2D</b><br><i>Filters = 12</i> | 1 | 0,70189% | 0,00825% | 0,37831% | 90,56515% |
|                                          | 3 | 0,54661% | 0,00461% | 0,28180% | 94,15953% |
|                                          | 5 | 0,38661% | 0,00239% | 0,20754% | 97,09628% |
| <b>ConvLSTM2D</b><br><i>Filters = 16</i> | 1 | 0,69117% | 0,00943% | 0,37054% | 93,66823% |
|                                          | 3 | 0,44213% | 0,00339% | 0,23476% | 96,74477% |
|                                          | 5 | 0,36951% | 0,00240% | 0,21502% | 97,75118% |
| <b>ConvLSTM2D</b><br><i>Filters = 20</i> | 1 | 0,65416% | 0,01009% | 0,33308% | 95,12155% |
|                                          | 3 | 0,44925% | 0,00329% | 0,26440% | 96,54282% |
|                                          | 5 | 0,40264% | 0,00238% | 0,20361% | 96,17790% |

\* ts: *timestep*

Fonte: Próprio autor.

## 4.2 Resultado da Melhor Arquitetura

Foram feitos novos testes prolongando o tempo de treinamento ( $epochs = 240$ ) e variando o número de neurônios artificiais das camadas LSTM. Diferentemente do melhor modelo apresentado na Tabela 9, a arquitetura da rede LSTM da Tabela 10 foi feita para o  $timestep = 1$  conforme os novos testes executados. Em comparação a arquitetura da Tabela 7, eliminou-se uma camada densa e o número de neurônios da camada LSTM agora é 768.

Tabela 10 – Arquitetura final da rede LSTM.

| <i>Layer</i>                       | <i>Output Shape</i> | <i># Param</i> |
|------------------------------------|---------------------|----------------|
| LSTM                               | (None, 768)         | 2.755.584      |
| Dropout                            | (None, 768)         | 0              |
| Dense                              | (None, 128)         | 98.432         |
| Activation                         | (None, 128)         | 0              |
| <i>Total Params: 2.854.016</i>     |                     |                |
| <i>Trainable Params: 2.854.016</i> |                     |                |
| <i>Non-Trainable Params: 0</i>     |                     |                |

Fonte: Próprio autor.

A ConvLSTM1D exibiu os piores resultados comparativamente, e por isso não foram testadas novas configurações. Para a arquitetura final da rede ConvLSTM2D, foram

utilizados os parâmetros  $filters = 16$  e  $kernel\_size = (5, 5)$ . Também eliminou-se a segunda camada densa vista na Tabela 8 e que não segue presente na arquitetura da rede ConvLSTM2D da Tabela 11. Nesse caso, o número de épocas foi 80.

Tabela 11 – Arquitetura final da rede ConvLSTM2D.

| <i>Layer</i>                       | <i>Output Shape</i> | <i># Param</i> |
|------------------------------------|---------------------|----------------|
| ConvLSTM2D                         | (None, 5, 128, 16)  | 27.264         |
| Flatten                            | (None, 10240)       | 0              |
| Dropout                            | (None, 10240)       | 0              |
| Dense                              | (None, 128)         | 1.310.848      |
| Activation                         | (None, 128)         | 0              |
| <i>Total Params: 1.338.112</i>     |                     |                |
| <i>Trainable Params: 1.338.112</i> |                     |                |
| <i>Non-Trainable Params: 0</i>     |                     |                |

Fonte: Próprio autor.

Com essas arquiteturas, conseguimos os resultados apresentados na Tabela 12.

Tabela 12 – Comparação final das redes construídas.

| <b>Rede Neural</b> | <b>Tempo</b> | <b>RMSE</b> | <b>MSE</b> | <b>MAE</b> | <b>R<sup>2</sup></b> |
|--------------------|--------------|-------------|------------|------------|----------------------|
| <b>LSTM</b>        | 5,5 s        | 0,29448%    | 0,00149%   | 0,16608%   | 98,53148%            |
| <b>ConvLSTM</b>    | 9 s          | 0,25158%    | 0,00104%   | 0,14754%   | 98,93181%            |

\* Tempo do algoritmo SSF gerar toda a base de dados: 16 s.

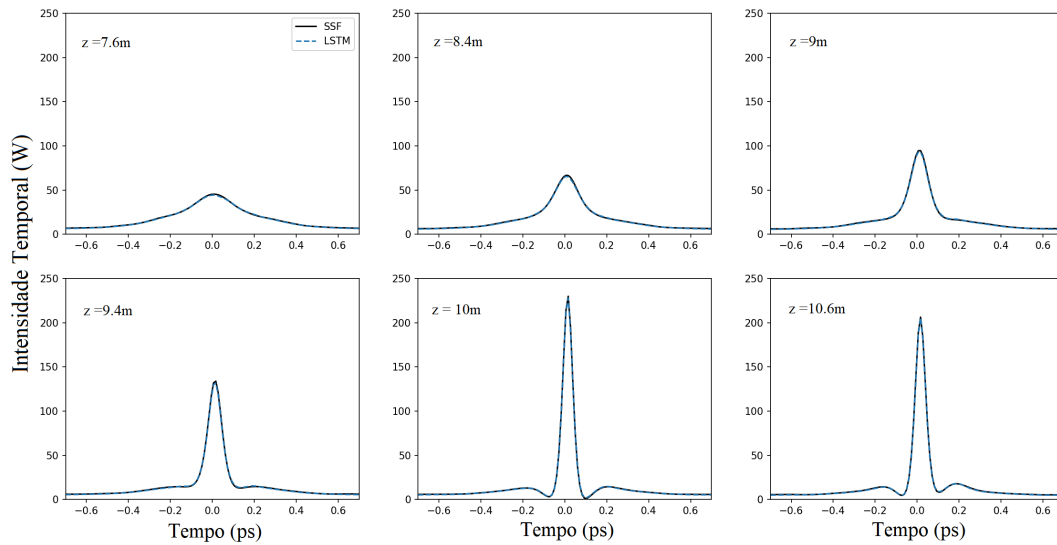
Fonte: Próprio autor.

Pela tabela, vemos que tanto a rede LSTM como a ConvLSTM exibem resultados ótimos, com  $RMSE < 0,3\%$  e  $R^2 > 98\%$  nos dois casos. Com essas métricas, o erro não é perceptível visualmente, Figuras 19 e 20. Nesse caso, comparando as redes pelo tempo para a predição de toda a base de dados, exibida na segunda coluna da Tabela 12, a rede LSTM é mais adequada por ser mais rápida, levando em média 5,5 s para predizer a base de testes; a rede ConvLSTM leva em média 9 s. Vale ressaltar que ambas apresentaram um tempo melhor que o algoritmo SSF, de 16 segundos. Também, avaliando as redes pelas métricas de avaliação dos erros, a rede ConvLSTM obteve os melhores resultados, com  $RMSE = 0,25\%$ ,  $MAE = 0,15\%$  e coeficiente  $R^2 = 99\%$ .

Como pode ser notado pelas figuras, as redes neurais propostas conseguem prever corretamente a propagação óptica na fibra.

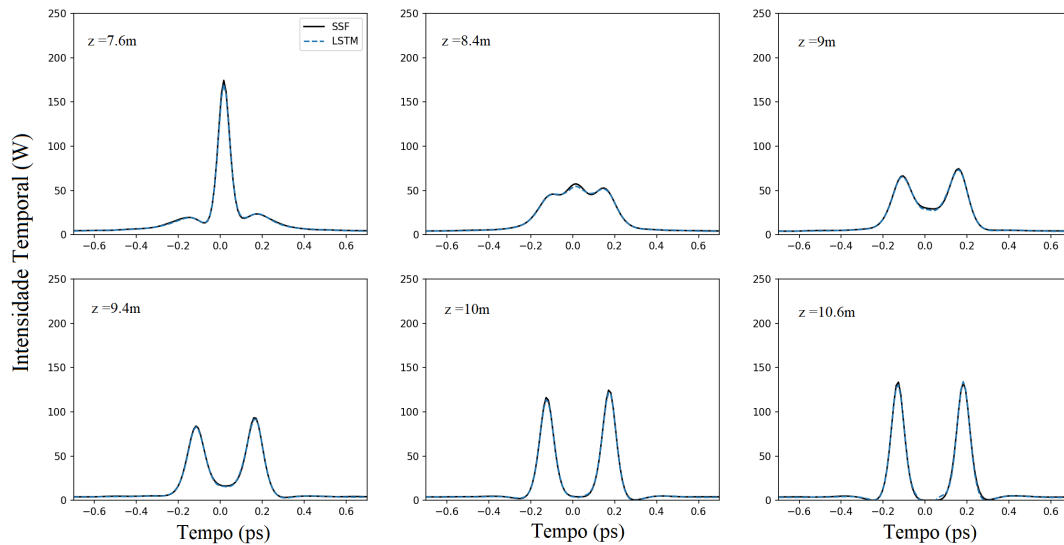


Figura 19 – Soliton com  $P_0 = 26,3$  W e  $T_{FWHM} = 1,1$  ps. SSF  $\times$  LSTM.



Fonte: Próprio autor.

Figura 20 – Soliton com  $P_0 = 34,19$  W e  $T_{FWHM} = 0,77$  ps. SSF  $\times$  LSTM.



Fonte: Próprio autor.

### 4.3 Novos Testes

Uma vantagem em usar redes neurais é sua capacidade de generalizar conhecimento, é isso que exploramos nessa seção. Para isso, foi gerado novos pulsos óticos de condição inicial distinta do intervalo  $P_0$  variando de 18,41 a 34,19 W e  $T_{FWHM}$  de 0,77 a 1,43 ps.

Agora, submetemos a rede sem efetuar novo treinamento a um conjunto de dados de pulsos gerados em um intervalo no qual a potência de pico  $P_0$  varia de 35 a 40 W e a largura de pulso  $T_{FWHM}$  varia de 0,77 a 2 ps. O novo conjunto gerado possui 700 (20 *powers*  $\times$  35 *widths*) condições iniciais distintas.

Na Tabela 13 a seguir, temos as métricas obtidas para as redes LSTM e ConvLSTM para esse novo *range*. O resultado é similar ao averiguado na Tabela 12, em que a rede LSTM efetua as previsões mais rápido que a rede ConvLSTM, levando em média 8,2 s e mínima de 6 s para predizer toda a base de dados novos.

Tabela 13 – Comparação das redes para novos dados.

| Rede Neural | Tempo | RMSE     | MSE      | MAE      | R <sup>2</sup> |
|-------------|-------|----------|----------|----------|----------------|
| LSTM        | 8,2 s | 0,48943% | 0,00444% | 0,23966% | 94,86023%      |
| ConvLSTM    | 10 s  | 0,41360% | 0,00281% | 0,21594% | 95,78696%      |

Fonte: Próprio autor.

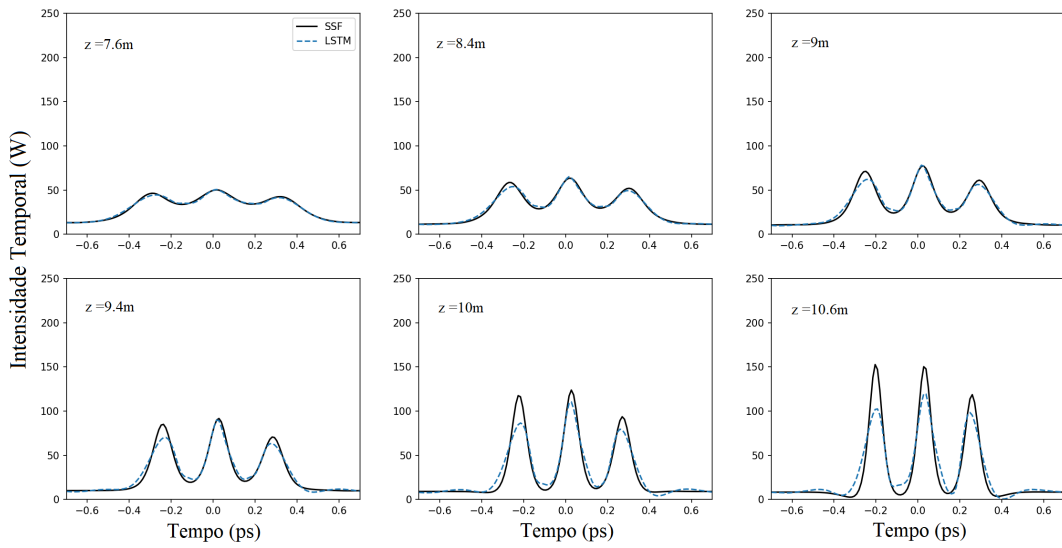
As métricas de erro também foram afetadas. A rede LSTM por exemplo, apresenta  $RMSE = 0,48943\%$  e  $R^2 = 94,86\%$ , enquanto a rede ConvLSTM resultou em um erro  $RMSE = 0,41360\%$  e coeficiente  $R^2 = 95,78696\%$ , essa rede é, ainda, mais robusta quando comparada a LSTM. As redes, desse modo, aumentaram o erro RMSE em mais de 60%, as métricas MSE e MAE também cresceram.

Apesar dos valores apresentados na Tabela 13 serem relativamente pequenos, o aumento percentual pode ser verificado nas Figuras 21 e 22, a primeira gerada a partir de um pulso de condição inicial  $P_0 = 40$  W e  $T_{FWHM} = 1,46$  ps ( $L_D/L_{NL} = 96,51$ ) e a segunda gerada a partir de um pulso com  $P_0 = 38,68$  W e  $T_{FWHM} = 1,96$  ps ( $L_D/L_{NL} = 168,19$ ).

Nas figuras, vemos que o *fit* não é adequado nos picos e vales do pulso propagado a medida que a potência de pico aumenta. A rede LSTM acompanha o método SSF até 9 metros de fibra, mas nos pontos 9,4 m, 10 m e 10,6 m o modelo neural não segue totalmente o numérico.

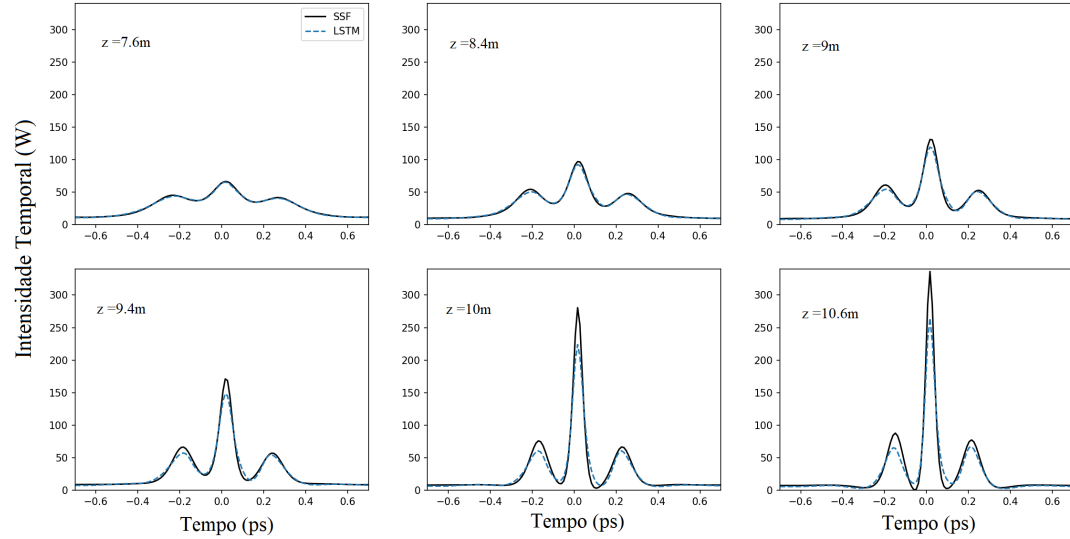
Mesmo assim, conseguimos uma boa aproximação, e o erro  $MSE = 4,44 \times 10^{-5}$  do LSTM nesse caso supera todas as arquiteturas descritas por Gautam, Choudhary e Lall (2021), que não avalia a rede ConvLSTM, com  $MSE = 2,81 \times 10^{-5}$ , ainda melhor que a LSTM. Não comparamos o resultado obtido com Salmela et al. (2021) por não ser usado no artigo uma métrica igual as tratadas aqui; o RMSE normalizado conseguido lá foi  $R = 0,097$ .

Figura 21 – Soliton com  $P_0 = 40,0$  W e  $T_{FWHM} = 1,46$  ps. SSF  $\times$  LSTM.



Fonte: Próprio autor.

Figura 22 – Soliton com  $P_0 = 38,68$  W e  $T_{FWHM} = 1,96$  ps. SSF  $\times$  LSTM.



Fonte: Próprio autor.

## 5 CONCLUSÕES E PROJETOS FUTUROS

### 5.1 Conclusões

O objetivo desse trabalho foi propor um modelo neural para o problema de propagação não linear de pulsos óticos, com desempenho similar ao algoritmo estado-da-arte *split-step* Fourier, a fim de obter um modelo de menor custo computacional e adaptável.

Durante a etapa de treinamento, foi avaliado um conjunto de diferentes redes neurais, variando parâmetros como o número de nós e de camadas da rede, o número de passos usados na predição (chamado aqui *timesteps*) e também o tipo de camada (LSTM ou ConvLSTM). A métrica principal para comparar essas redes foi a raiz do erro quadrático médio, RMSE, mas também foram calculadas a MSE, MAE e o coeficiente  $R^2$ .

Dada as características de cada rede, foram selecionadas as duas melhores avaliadas, uma LSTM e outra *convolutional* LSTM. Nos dois casos os resultados foram excelentes, com  $RMSE < 0,3\%$ . A partir das métricas de avaliação de erro, a rede ConvLSTM2D usada no trabalho foi mais robusta, com  $RMSE = 0,25\%$  e  $R^2 = 99\%$ . Comparando as redes pelo tempo de predição, a rede LSTM foi mais eficiente, com tempo médio de predição igual a 5,5 segundos para o *dataset* reservado aos testes. A escolha do modelo dependerá da aplicação, se o erro menor é essencial (e.g. modelo numérico de um laser) ou ainda aplicações nas quais o tempo deve ser mínimo.

Também foi mostrado que as mesmas redes podem ser estendidas para uma nova aplicação, sem necessidade de retreinamento. Nesse caso, a rede LSTM obteve erro  $RMSE = 0,48943\%$  e a ConvLSTM um erro  $RMSE = 0,41360\%$ , um aumento de 60% em relação ao erro obtido com o conjunto de testes. A predição funciona adequadamente para a propagação enquanto a variação da potência ótica se mantém pequena.

Dessa forma, este trabalho explora e constrói modelos neurais viáveis para o problema complexo de propagação ótica, tendo como resultado duas redes de características distintas e próprias para cada aplicação.

## 5.2 Projetos futuros

Deste trabalho, pode ser derivado os seguintes problemas de pesquisa:

1. **Telecomunicações:** O problema tratado no corpo do texto foi o de uma fibra de somente 13 metros com propagação predominantemente não linear. Podemos adaptar a rede construída para o problema de telecomunicações, onde ocorre a transmissão de pulsos óticos por milhares de quilômetros, e a propagação é dispersiva, ou linear. Ao utilizar uma rede neural para tratar o problema de propagação, é possível usar técnicas como o *transfer learning*, que transfere o conhecimento das redes treinadas a um novo problema, relacionado ao anterior, dessa forma estendendo seu uso.
2. **Lasers:** O caso apresentado neste trabalho foi de uma propagação de um pulso ótico, mas não abordamos a geração desse pulso, o que é geralmente feita empregando lasers. Na construção de um laser, ocorre a amplificação e absorção do feixe de luz repetidas vezes em um anel formado por fibra ótica. Nele, os efeitos não lineares e lineares se combinam de maneira complexa. Por isso, a aplicação de redes neurais pode contribuir nos estudos dessa área.
3. **Evolução espectral de Super-contínuos:** Presente no artigo que inspirou este trabalho, a aplicação de *machine learning* se estende também ao espectro e pode ser encontrado um caso da evolução de super-contínuo, ou seja, um pulso ótico que, quando propagado, tem sua banda espectral espalhada para uma larga faixa do comprimento de onda.
4. **Dinâmicas não lineares espaço-temporais em fibras multi-modos:** A partir da rede desenvolvida por [Salmela et al. \(2021\)](#) para pulsos em fibra monomodo, [Teġin et al. \(2021\)](#) propõem o reuso dessa estrutura para predições de dinâmicas não lineares multimodais.

## REFERÊNCIAS

- AGRAWAL, G. P. **Nonlinear Fiber Optics**. 4. ed. [S.l.]: Academic Press, 2007. Citado 3 vezes nas páginas [17](#), [18](#) e [32](#).
- AGRAWAL, G. P. **Fiber-Optic Communication Systems**. 5. ed. [S.l.]: John Wiley & Sons, 2021. Citado na página [14](#).
- ALESHAMS, M.; ZARIFKAR, A.; SHEIKHI, M. Split-step fourier transform method in modeling of pulse propagation in dispersive nonlinear optical fibers. In: **Proceedings of CAOL 2005. Second International Conference on Advanced Optoelectronics and Lasers, 2005**. [S.l.: s.n.], 2005. v. 2, p. 124–126 vol. 2. Citado na página [14](#).
- BROWNLEE, J. *Deep Learning for Time Series Forecasting. Machine Learning Mastering*, 2017. Disponível em: <https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/>. Acesso em: 27 nov. 2021. Citado na página [38](#).
- BROWNLEE, J. *How to Diagnose Overfitting and Underfitting of LSTM Models. Machine Learning Mastering*, 2017. Disponível em: <https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/>. Acesso em: 12 fev. 2022. Citado na página [42](#).
- CASTELLANI, C. E. S. **All-fibre wavelength versatile short pulsed laser sources**. Tese (Doutorado) — Imperial College London, United Kingdom, 10 2013. Citado 2 vezes nas páginas [14](#) e [19](#).
- CERTÓRIO, C. H. da S. **Simulador MATLAB para Sistemas Ópticos WDM Amplificados**. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, 2009. Citado na página [17](#).
- CHOLLET, F. et al. **Keras**. 2015. Disponível em: [https://keras.io/api/layers/recurrent\\_layers/lstm/](https://keras.io/api/layers/recurrent_layers/lstm/). Acesso em: 15 dez. 2021. Citado na página [38](#).
- CHOLLET, F. et al. **Keras**. 2015. Disponível em: [https://keras.io/api/layers/recurrent\\_layers/conv\\_lstm2d/](https://keras.io/api/layers/recurrent_layers/conv_lstm2d/). Acesso em: 27 dez. 2021. Citado na página [38](#).
- DEITERDING, R.; GLOWINSKI, R.; OLIVER, H.; POOLE, S. A reliable split-step fourier method for the propagation equation of ultra-fast pulses in single-mode optical fibers. **Journal of Lightwave Technology**, v. 31, p. 2008–2017, 06 2013. Citado na página [25](#).
- DONAHUE, J.; HENDRICKS, L. A.; ROHRBACH, M.; VENUGOPALAN, S.; GUADARRAMA, S.; SAENKO, K.; DARRELL, T. Long-term recurrent convolutional networks for visual recognition and description. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 4, p. 677–691, 2017. Citado na página [31](#).
- FRIGO, M.; JOHNSON, S. G. The design and implementation of FFTW3. **Proceedings of the IEEE**, v. 93, n. 2, p. 216–231, 2005. Special issue on “Program Generation, Optimization, and Platform Adaptation”. Citado na página [24](#).

GAUTAM, N.; CHOUDHARY, A.; LALL, B. Neural networks for predicting optical pulse propagation through highly nonlinear fibers. In: **2021 National Conference on Communications (NCC)**. [S.l.: s.n.], 2021. p. 1–6. Citado na página [49](#).

GENTY, G.; SALMELA, L.; DUDLEY, J. M.; BRUNNER, D.; KOKHANOVSKIY, A.; KOBTSEV, S.; TURITSYN, S. K. Machine learning and applications in ultrafast photonics. **Nature Photonics**, v. 15, p. 91–101, 02 2021. Citado na página [15](#).

GOOGLE. **Google Colaboratory**. Mountain View, 2021. Disponível em: <https://colab.research.google.com/notebooks/intro.ipynb>. Acesso em: 13 set. 2021. Citado na página [38](#).

HECHT, J. **City of Light: The Story of Fiber Optics**. Oxford: Oxford University Press, 1999. (Sloan Technology). Citado na página [14](#).

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 11 1997. Citado 2 vezes nas páginas [15](#) e [29](#).

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, v. 349, n. 6245, p. 255–260, 2015. Citado na página [15](#).

KAPRON, F. P.; KECK, D. B.; MAURER, R. D. Radiation Losses in Glass Optical Waveguides. **Applied Physics Letters**, v. 17, n. 10, p. 423, 8 1970. Citado na página [19](#).

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological Cybernetics**, v. 43, p. 59–69, 1982. Citado na página [27](#).

MATHWORKS. **White Paper: Deep Learning for Signal Processing with MATLAB**. [S.l.], 2019. Citado na página [26](#).

MATLAB. **9.11.0.1769968 (R2021b)**. Natick, Massachusetts: The MathWorks Inc., 2021. Disponível em: <https://www.mathworks.com/>. Acesso em: 21 nov. 2021. Citado na página [32](#).

NÄRHI, M.; SALMELA, L.; TOIVONEN, J.; BILLET, C.; DUDLEY, J. M.; GENTY, G. Machine learning analysis of extreme events in optical fibre modulation instability. **Nature Communications**, v. 9, n. 1, p. 4923, Nov 2018. Citado na página [15](#).

OLIARI, V.; GOOSSENS, S.; HÄGER, C.; LIGA, G.; BÜTLER, R. M.; HOUT, M. van den; HEIDE, S. van der; PFISTER, H. D.; OKONKWO, C.; ALVARADO, A. Revisiting efficient multi-step nonlinearity compensation with machine learning: An experimental demonstration. **Journal of Lightwave Technology**, v. 38, n. 12, p. 3114–3124, 2020. Citado na página [14](#).

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. Novo Hamburgo: Feevale, 2013. Citado na página [32](#).

ROJAS, R. **Neural Networks: A Systematic Introduction**. 1. ed. Berlim: Springer, 1996. Citado 3 vezes nas páginas [26](#), [27](#) e [28](#).

SAINATH, T. N.; VINYALS, O.; SENIOR, A. W.; SAK, H. Convolutional, long short-term memory, fully connected deep neural networks. **2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**, p. 4580–4584, 2015. Citado na página [31](#).

SALMELA, L.; TSIPINAKIS, N.; FOI, A.; BILLET, C.; DUDLEY, J. M.; GENTY, G. Predicting ultrafast nonlinear dynamics in fibre optics with a recurrent neural network. **Nature Machine Learning**, v. 3, p. 344–354, 02 2021. ArXiv preprint arXiv:2004.14126. Citado 6 vezes nas páginas [15](#), [32](#), [34](#), [35](#), [49](#) e [52](#).

SCHLOSS, J. **The Arcane Algorithm Archive**. 2018. Disponível em: [<https://www.algorithm-archive.org/>](https://www.algorithm-archive.org/). Acesso em: 13 set. 2021. Citado na página [24](#).

SHI, X.; CHEN, Z.; WANG, H.; YEUNG, D. Y.; WONG, W. K.; WOO, W. C. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: **Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1**. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 802–810. Citado na página [31](#).

TEĞİN, U.; DINÇ, N. U.; MOSER, C.; PSALTIS, D. Reusability report: Predicting spatiotemporal nonlinear dynamics in multimode fibre optics with a recurrent neural network. **Nature Machine Intelligence**, v. 3, p. 387–391, 05 2021. Citado na página [52](#).

VARGAS, G. M. **Predição do Preço de Ações Utilizando Redes Neurais e Análise de Sentimento**. 2021. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação), UFES (Universidade Federal do Espírito Santo), Espírito Santo, Brasil. Citado 2 vezes nas páginas [27](#) e [29](#).

WEISS, G. H.; MARADUDIN, A. A. The baker-hausdorff formula and a problem in crystal physics. **Journal of Mathematical Physics**, v. 3, p. 771, 02 1962. Citado na página [24](#).



# Apêndices

## APÊNDICE A – EQUAÇÕES DE MAXWELL

Para todos os fenômenos eletromagnéticos, inclusive fenômenos óticos, nos apoiamos nas equações de Maxwell para o desenvolvimento matemático necessário. São quatro as equações,

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \nabla \cdot \mathbf{D} = \rho_f \qquad (\text{A.1})$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \qquad \nabla \cdot \mathbf{B} = 0 \qquad (\text{A.2})$$

onde  $\mathbf{E}$  é o vetor campo elétrico e  $\mathbf{H}$  o vetor campo magnético. E também  $\mathbf{D}$  e  $\mathbf{B}$  são os vetores densidade de fluxo elétrico e magnético correspondentes. Ainda,  $\mathbf{J}$  é o vetor densidade de corrente e  $\rho_f$  a densidade de carga elétrica. Como sabemos, numa fibra ótica, sendo um meio dielétrico e por isso, sem cargas livres. Nesse caso  $\mathbf{J} = \mathbf{0}$  e  $\rho_f = 0$ .

Os vetores densidade de fluxo se relacionam com seus respectivos campos através das Equações [A.3](#) e [A.4](#),

$$\mathbf{D} = \varepsilon_0 \mathbf{E} + \mathbf{P} \qquad (\text{A.3})$$

$$\mathbf{B} = \mu_0 \mathbf{H} + \mathbf{M} \qquad (\text{A.4})$$

onde  $\varepsilon_0$  é a permissividade elétrica do vácuo e  $\mu_0$  a permeabilidade magnética do vácuo, e  $\mathbf{P}$  e  $\mathbf{M}$  são as polarizações elétrica e magnética. Na fibra ótica,  $\mathbf{M} = \mathbf{0}$ , então  $\mathbf{B} = \mu_0 \mathbf{H}$ .

A partir daí, buscamos obter a equação de onda que descreve a propagação de luz na fibra

$$\begin{aligned} \nabla \times (\nabla \times \mathbf{E}) &= \nabla \times \left( -\frac{\partial \mathbf{B}}{\partial t} \right) \\ &= -\mu_0 \frac{\partial (\nabla \times \mathbf{H})}{\partial t} \\ &= -\mu_0 \frac{\partial^2 \mathbf{D}}{\partial t^2} \\ &= -\frac{1}{c^2} \frac{\partial^2 \mathbf{E}}{\partial t^2} - \mu_0 \frac{\partial^2 \mathbf{P}}{\partial t^2} \end{aligned}$$

e foi usado a relação  $\mu_0 \varepsilon_0 = 1/c^2$ . No caso de interesse nos efeitos não lineares, a polarização  $\mathbf{P}$  inclui uma parcela linear  $\mathbf{P}_L$  e outra não linear  $\mathbf{P}_{NL}$ ,

$$\mathbf{P}(\mathbf{r}, t) = \mathbf{P}_L(\mathbf{r}, t) + \mathbf{P}_{NL}(\mathbf{r}, t). \qquad (\text{A.5})$$

E aplicando a identidade vetorial

$$\nabla \times \nabla \times \mathbf{E} \equiv \nabla (\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E} = -\nabla^2 \mathbf{E} \qquad (\text{A.6})$$

conseguimos a equação de onda,

$$\nabla^2 \mathbf{E} - \frac{1}{c^2} \frac{\partial^2 \mathbf{E}}{\partial t^2} = \mu_0 \frac{\partial^2 \mathbf{P}_L}{\partial t^2} + \mu_0 \frac{\partial^2 \mathbf{P}_{NL}}{\partial t^2}. \quad (\text{A.7})$$

Por fim, simplificamos a Equação (A.7) realizando a transformação de Fourier de  $\mathbf{E}(\mathbf{r}, t)$ , e relacionando os demais termos com esse vetor. Isso resume a equação de Helmholtz abaixo,

$$\nabla^2 \tilde{\mathbf{E}} + \varepsilon(\omega) k_0^2 \tilde{\mathbf{E}} = 0 \quad (\text{A.8})$$

onde  $k_0 = \omega/c$ ,  $\tilde{\mathbf{E}}(\mathbf{r}, \omega)$  é a transformada de  $\mathbf{E}(\mathbf{r}, t)$  e a constante dielétrica é dada por  $\varepsilon(\omega) = 1 + \chi^{(1)} + \varepsilon_{NL}$ , com as contribuições lineares e não lineares da constante dielétrica, relacionados as respectivas polarizações.

Nas equações acima foram usadas muitas simplificações, cobrindo de maneira introdutória o assunto, recomenda-se a leitura de outras fontes para se aprofundar no assunto.

## APÊNDICE B – EQUAÇÃO DE PROPAGAÇÃO NÃO LINEAR

Partindo da equação de Helmholtz, vamos fazer o desenvolvimento matemático da Equação de propagação (2.2),

$$\nabla^2 \tilde{\mathbf{E}} + \varepsilon(\omega) k_0^2 \tilde{\mathbf{E}} = 0. \quad (\text{B.1})$$

Assumindo uma solução do tipo,

$$\tilde{\mathbf{E}}(\mathbf{r}, \omega - \omega_0) = F(x, y) \tilde{A}(z, \omega - \omega_0) \exp(i\beta_0 z), \quad (\text{B.2})$$

onde  $\tilde{A}(\omega)$  é uma função que varia lentamente em  $z$ ,  $F(x, y)$  a distribuição modal, e  $\beta_0$  o número de onda. Substituindo (B.2) em (B.1), e sabendo que o operador laplaciano  $\nabla^2$  é

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}, \quad (\text{B.3})$$

por meio do método de separação de variáveis, temos

$$\frac{\partial^2 [F \tilde{A} \exp(i\beta_0 z)]}{\partial x^2} + \frac{\partial^2 [F \tilde{A} \exp(i\beta_0 z)]}{\partial y^2} + \frac{\partial^2 [F \tilde{A} \exp(i\beta_0 z)]}{\partial z^2} + \varepsilon(\omega) k_0^2 F \tilde{A} \exp(i\beta_0 z) = 0$$

$$\tilde{A} \exp(i\beta_0 z) \left( \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \varepsilon(\omega) k_0^2 F \right) + F \frac{\partial^2 [\tilde{A} \exp(i\beta_0 z)]}{\partial z^2} = 0$$

$$\tilde{A} \exp(i\beta_0 z) \left( \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \varepsilon(\omega) k_0^2 F \right) + F \exp(i\beta_0 z) \left( \frac{\partial^2 \tilde{A}}{\partial z^2} + 2i\beta_0 \frac{\partial \tilde{A}}{\partial z} - \beta_0^2 \tilde{A} \right) = 0$$

$$\tilde{A} \left( \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \varepsilon(\omega) k_0^2 F \right) + F \left( 2i\beta_0 \frac{\partial \tilde{A}}{\partial z} - \beta_0^2 \tilde{A} \right) = 0$$

$$\tilde{A} \left( \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \varepsilon(\omega) k_0^2 F \right) - \tilde{A} (F \tilde{\beta}^2) + F \left( 2i\beta_0 \frac{\partial \tilde{A}}{\partial z} - \beta_0^2 \tilde{A} \right) + F (\tilde{A} \tilde{\beta}^2) = 0$$

$$\tilde{A} \left( \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + [\varepsilon(\omega) k_0^2 - \tilde{\beta}^2] F \right) + F \left( 2i\beta_0 \frac{\partial \tilde{A}}{\partial z} + (\tilde{\beta}^2 - \beta_0^2) \tilde{A} \right) = 0$$

A segunda derivada  $\partial^2 \tilde{A} / \partial z^2$  foi eliminada visto que assumimos  $\tilde{A}(z, \omega)$  variando lentamente em função de  $z$ . Portanto,

$$\frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + [\varepsilon(\omega) k_0^2 - \tilde{\beta}^2] F = 0 \quad (\text{B.4})$$

$$2i\beta_0 \frac{\partial \tilde{A}}{\partial z} + (\tilde{\beta}^2 - \beta_0^2) \tilde{A} = 0 \quad (\text{B.5})$$

em que  $F(x, y)$  pode ser aproximada por  $F \approx \exp[-(x^2 + y^2)/w^2]$  e  $A(z, \omega)$  é obtida da Equação (B.5).

A seguir, fazemos algumas considerações

$$\begin{aligned}\tilde{\beta}(\omega) &\approx \beta(\omega) + \Delta\beta_0, \\ \tilde{\beta}^2 - \beta_0^2 &\approx 2\beta_0(\tilde{\beta} - \beta_0),\end{aligned}$$

que, substituindo na Equação (B.5), obtemos

$$\frac{\partial \tilde{A}}{\partial z} = i[\beta(\omega) + \Delta\beta_0 - \beta_0]\tilde{A}. \quad (\text{B.6})$$

E  $\beta(\omega)$  é normalmente dado em uma série de Taylor,

$$\beta(\omega) = \beta_0 + (\omega - \omega_0)\beta_1 + \frac{1}{2}(\omega - \omega_0)^2\beta_2 + \frac{1}{6}(\omega - \omega_0)^3\beta_3 + \dots, \quad (\text{B.7})$$

onde

$$\beta_m = \left( \frac{d^m \beta}{d\omega^m} \right)_{\omega=\omega_0} \quad (m = 1, 2, \dots). \quad (\text{B.8})$$

É suficiente para o caso de propagação tratado aqui valores de  $m \leq 3$ , termos de maior ordem podem ser ignorados, sendo assim

$$\frac{\partial \tilde{A}}{\partial z} = i[(\omega - \omega_0)\beta_1 + \frac{1}{2}(\omega - \omega_0)^2\beta_2 + \frac{1}{6}(\omega - \omega_0)^3\beta_3 + \Delta\beta_0]\tilde{A} \quad (\text{B.9})$$

E portanto, finalmente fazendo a transformação inversa de Fourier na Equação (B.9), usando  $(\omega - \omega_0) \leftrightarrow i\partial/\partial t$ ,

$$\frac{\partial A}{\partial z} + \beta_1 \frac{\partial A}{\partial t} + i\frac{\beta_2}{2} \frac{\partial^2 A}{\partial t^2} - \frac{\beta_3}{6} \frac{\partial^3 A}{\partial t^3} = i\Delta\beta_0 A, \quad (\text{B.10})$$

com o termo  $\Delta\beta_0$  incluindo os efeitos de atenuação e não linearidade, sendo substituído por  $\alpha$  e  $\gamma$ , respectivamente,

$$\frac{\partial A}{\partial z} + \beta_1 \frac{\partial A}{\partial t} + i\frac{\beta_2}{2} \frac{\partial^2 A}{\partial t^2} - \frac{\beta_3}{6} \frac{\partial^3 A}{\partial t^3} + \frac{\alpha}{2} A = i\gamma(\omega_0)|A|^2 A \quad (\text{B.11})$$

Por fim, fazendo a substituição pelo frame de referência  $T = t - z/v_g \equiv t - \beta_1 z$ . Pela regra da cadeia, temos

$$\begin{aligned}\frac{\partial A}{\partial t} &= \frac{\partial A}{\partial T} \frac{\partial T}{\partial t} + \frac{\partial A}{\partial z} \frac{\partial z}{\partial t} = \frac{\partial A}{\partial T} \\ \frac{\partial A}{\partial z} &= \frac{\partial A}{\partial z} \frac{\partial z}{\partial z} + \frac{\partial A}{\partial T} \frac{\partial T}{\partial z} = \frac{\partial A}{\partial z} - \beta_1 \frac{\partial A}{\partial T}\end{aligned}$$

E finalmente, obtemos a equação de propagação não linear

$$\frac{\partial A}{\partial z} + i\frac{\beta_2}{2} \frac{\partial^2 A}{\partial T^2} - \frac{\beta_3}{6} \frac{\partial^3 A}{\partial T^3} + \frac{\alpha}{2} A = i\gamma|A|^2 A. \quad (\text{B.12})$$

## APÊNDICE C – DECIBEL

O uso da unidade de medida relativa adimensional decibel (dB) na engenharia facilita as comparações de valores com grande *range*, como potência e frequência. Outra aplicação comum são as perdas de transmissão em telecomunicações. Uma razão  $R$  pode ser escrita em decibéis através da Equação [C.1](#)

$$R_{\text{dB}} = 10 \log_{10} R \quad (\text{C.1})$$

Nesse caso, com o logaritmo de base dez, conseguimos comparar diferentes ordens de grandezas de maneira simples,

Tabela 14 – Comparação de valores em dB.

| $R_{\text{dB}}$ | Razão $R$     |
|-----------------|---------------|
| 30              | = 1.000       |
| 20              | = 100         |
| 10              | = 10          |
| 6               | $\approx 4$   |
| 3               | $\approx 2$   |
| 0               | = 1           |
| -3              | $\approx 1/2$ |
| -6              | $\approx 1/4$ |
| -10             | = 1/10        |
| -20             | = 1/100       |
| -30             | = 1/1.000     |

Fonte: Próprio autor.

Vale lembrar ainda que  $R > 0$  por se tratar de uma função logarítmica.

# Anexos

## ANEXO A – CÓDIGO PRINCIPAL

```

% Escrito por Gustavo R Martins
% Parte do Trabalho de Conclusão de Curso - TCC
clear; clc;

% ===== PARÂMETROS DA FIBRA =====
% =====

L = 13; % Comprimento da fibra (m);
beta2 = -5.23e-27; % Dispersão GVD (s2/m): + normal, - anômalo
beta3 = 4.27e-41; % Dispersão TOD (S3/m): + right, - left;
gamma = 18.4e-3; % Auto-modulação (W/m)
alpha_dB = 0.05; % Atenuação da fibra (~0.2 dB/km)
alpha = alpha_dB/4.343;
S = 0; % self-steepening
tauR = 0; % Espalhamento Raman

% ===== PARÂMETROS DO PULSO =====
% =====

numPower = 40;
numTFWHM = 75;
PP = linspace(18.41,34.19,numPower); % Potência de entrada
TT = linspace(0.77e-12,1.43e-12,numTFWHM); % FWHM

lambda0 = 1550e-9; % Comprimento-de-onda central
mshape = 0; % m = 0 para sech, m > 0 para Gaussian;
chirp0 = 0; % Chirp (C = 0, default)

c = physconst('LightSpeed');
w0 = c/lambda0;

if mshape==0
 T0 = TT/(2*log(1+sqrt(2)));
else
 T0 = TFWHM/(2*sqrt(log(2)));

```



```

end

n = 0;
BETA2 = zeros(1,numPower*numTFWHM);
BETA3 = zeros(1,numPower*numTFWHM);
GAMMA = zeros(1,numPower*numTFWHM);
for p = 1:numPower
 for t = 1:numTFWHM
 n = n + 1;
 P0 = PP(p); T0 = T0(t);
 [BETA2(n),BETA3(n),GAMMA(n)] = param(beta2,beta3,gamma,P0,T0);
 end
end
end

% ===== PARÂMETROS DA SIMULAÇÃO =====
% =====

nt = 2^10; % N° pontos FFT
Tmax = 0.7e-12; % Window size
step_num = 1e3; % N° passos
deltaz = L/step_num; % step size in z

zsamp = 10;
tsamp = 4;

XXX = zeros(step_num/zsamp,nt/tsamp,numPower*numTFWHM);

n = 0;
tic
for p = 1:numPower
 for t = 1:numTFWHM

 n = n + 1;

 dtau = (2*Tmax/T0(t))/nt;
 fs = 1/dtau;

 %---tau and omega arrays
 tau = (-nt/2:nt/2-1)*dtau; % Grid temporal
 end
end

```

```

omega = (2*pi*fs/nt)*[(0:nt/2-1) (-nt/2:-1)]; % Grid espectral

% ===== PULSO =====
uu = sech(tau); % soliton

% ===== HAMILTONIANOS =====
D = 1i*BETA2(n)*omega.^2/factorial(2);
D = 1i*BETA3(n)*omega.^3/factorial(3) + D - alpha/2;

D = exp(D*deltaz);
NL = 1i*GAMMA(n)*deltaz;

% ===== SPLIT-STEP =====
U_final = fiber(uu,step_num,D,NL,omega);

% ===== PREPARANDO =====

U_final = PP(p)*abs(U_final).^2;

U_final = downsample(U_final,zsamp);
U_final = U_final.';

U_final = downsample(U_final,tsamp);
U_final = U_final.';

XXX(:, :, n) = U_final;
end
end
toc

```

## ANEXO B – PARAMETRIZAÇÃO

```

function [beta2,beta3,gamma] = param(beta2,beta3,gamma,P0,T0)
arguments
 beta2 = 0
 beta3 = 0
 gamma = 0
 P0 = 1
 T0 = 1
end

if (beta3 ~= 0)
 L_D = T0^3/abs(beta3);
 beta3 = sign(beta3)/L_D;
end

if (beta2 ~= 0)
 L_D = T0^2/abs(beta2);
 % N = sqrt(gamma*P0*T0^2/abs(beta2));
 beta2 = sign(beta2)/L_D;
end

if gamma ~= 0
 L_NL = 1/(gamma*P0);
 gamma = 1/L_NL;
end
end

```

## ANEXO C – FIBRA (SPLIT-STEP)

```

function A = fiber(U,M,D,GAMMA,OMEGA,S,R)
 arguments
 U; M; D;
 GAMMA; OMEGA;
 S = 0; R = 0;
 end

 % NL é o Hamiltoniano da Não linearidade
 NL = abs(U).^2 + ...
 1i*S*conj(U).*ifft(1i*(OMEGA).*fft(U))+...
 (1i*S - R)*ifft(1i*(OMEGA).*fft(abs(U).^2));

 PULSE_EVOL = zeros(M,length(U));

 A = U.*exp(NL.*GAMMA/2); % note h/2
 for n=1:M
 % Armazena evolução do pulso numa matriz:
 PULSE_EVOL(n,:) = A.*exp(NL.*-GAMMA/2);

 % Propagação da fibra:
 U = fft(D.*ifft(A));
 NL = abs(U).^2 + ...
 1i*S*conj(U).*ifft(1i*(OMEGA).*fft(U))+...
 (1i*S - R)*ifft(1i*(OMEGA).*fft(abs(U).^2));

 A = U.*exp(NL.*GAMMA);
 end
 end

 NL = abs(U).^2 + ...
 1i*S*conj(U).*ifft(1i*(OMEGA).*fft(U)) + ...
 (1i*S - R)*ifft(1i*(OMEGA).*fft(abs(U).^2));

 U = A.*exp(-NL.*GAMMA/2); % Final field

 A = PULSE_EVOL;
end

```

## ANEXO D – REDE NEURAL

```

% Escrito por Gustavo R Martins
% Parte do Trabalho de Conclusão de Curso - TCC
% Escrito em Python no Google Collab

% ===== INTEGRAÇÃO COM O DRIVE =====
% =====

!pip install xlrd
from google.colab import drive
drive.mount('/content/drive')

% ===== BIBLIOTECAS BÁSICAS =====
% =====

import scipy.io
import matplotlib.pyplot as plt
import gc
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import TimeSeriesSplit

% ===== CARREGANDO DADOS =====
% =====

datapath = '/content/drive/My Drive/Colab Notebooks/TCC/XXXStand.mat'
xxx = scipy.io.loadmat(datapath)
mat = xxx['XXX']

% ===== DEFININDO FUNÇÃO DE NORMALIZAÇÃO =====
% =====

def scale(X):
 X_std = (X - X.min()) / (X.max() - X.min())
 return X_std, (X.max() - X.min()), X.min()

```

```

% ===== TRANSFORMANDO EM PROBLEMA DE REGRESSÃO =====
% =====

ts = 3
cmd = 3000;
samples = 3000*(100-ts)
features = 128
timestep = ts + 1

dataset = np.zeros((samples*timestep,features))
cnt = 0
for m in range(0,cmd):
 for n in range(0,100-ts):
 dataset[cnt*timestep:(cnt+1)*timestep,:] = mat[n:n+timestep,:,m]
 cnt += 1

Normalizar
[dataset,R,Xmin] = scale(dataset)
data = np.reshape(dataset,(samples,timestep,features))

Show Dataset
columns = ['x(t-3)', 'x(t-2)', 'x(t-1)', 'x(t)']
pd.DataFrame(np.transpose(data[0,0:11,:]), columns=columns)

% ===== DIVIDINDO DATASET =====
% =====

trainSize = int(len(XData)*0.8) # Tamanho dos dados de treinamento

XTest = XData[trainSize:len(XData)] # Testes
YTest = YData[trainSize:len(XData)]

validSize = int(trainSize * 0.8) # Tamanho dos dados de treinamento

XTrain = XData[0:validSize] # Treinamento
YTrain = YData[0:validSize] # Treinamento
XValid = XData[validSize:trainSize] # Validação
YValid = YData[validSize:trainSize] # Validação

```

```
% ===== BIBLIOTECA PARA ML =====
% =====

import keras
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.callbacks import TensorBoard
from time import time
from keras.callbacks import ModelCheckpoint

% ===== PARÂMETRO DA REDE =====
% =====

neurons = 256 # number of hidden units in the LSTM layer
actv_func = 'tanh' # activation function for LSTM and Dense layer
loss = 'mse' # loss function for calculating the gradient
optimizer= 'RMSprop' # optimizer for applying gradient decent
dropout = 0.25 # dropout ratio used after each LSTM layer
batch_size = 128
epochs = 60
window_len = 60 # to be used as the look back window

% ===== CONSTRUINDO MODELO =====
% =====

def build_model(inputs, output_size, neurons, actv_func=actv_func,
 dropout=dropout, loss=loss, optimizer=optimizer):
 model = Sequential()
 model.add(LSTM(256, return_sequences=False, activation=actv_func,
 input_shape=(inputs.shape[1], inputs.shape[2])))
 model.add(Dropout(dropout))
 model.add(Dense(units=output_size))
 model.add(Activation(actv_func))
 model.compile(loss=loss, optimizer=optimizer, metrics=['mae'])
 model.summary()
 return model
```

```
% ===== ARQUITETURA =====
% =====

gc.collect()

random seed for reproducibility
np.random.seed(202)

initialise model architecture
rnn_model = build_model(XTrain,output_size=YTrain.shape[1],
 neurons=neurons)
tensorboard = TensorBoard(log_dir="log/{}".format(time()))

filepath="weights.best.hdf5"
metric = 'val_loss'
checkpoint = ModelCheckpoint(filepath, monitor=metric, verbose=1,
 save_best_only=True, mode='auto')
callbacks_list = [checkpoint]

% ===== TREINANDO =====
% =====

history = rnn_model.fit(XTrain,YTrain,epochs=epochs,batch_size=batch_size,
 verbose=1, validation_data=(XValid, YValid),
 shuffle=True, callbacks=[tensorboard,checkpoint])

% ===== LOSS DURANTE TREINO =====
% =====

#Plot Loss
pyplot.figure(figsize=(9,6))
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='val')
pyplot.legend()

% ===== PREDIÇÃO =====
% =====

encontradoTest = rnn_model.predict(XTest)
```



```
% ===== MÉTRICAS =====
% =====

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import r2_score, max_error

MSE = mean_squared_error(YTest, encontradoTest, squared=True)
RMSE = mean_squared_error(YTest, encontradoTest, squared=False)
R2 = r2_score(YTest, encontradoTest)
MAE = mean_absolute_error(YTest, encontradoTest)
MAXE = max_error(YTest.flatten(), encontradoTest.flatten())

print('RMSE = ', RMSE*100,'%')
print('MSE = ', MSE*100,'%')
print('MAE = ', MAE*100,'%')
print('R2 = ', R2*100,'%')
print('MAX = ', MAXE)

% ===== SALVANDO REDE CRIADA =====
% =====

savepath = "/content/drive/My Drive/Colab Notebooks/TCC/lstm-model.h5"
rnn_model.save(savepath)

% ===== CARREGANDO REDE CRIADA =====
% =====

savepath = "/content/drive/My Drive/Colab Notebooks/TCC/lstm-model.h5"
rnn_model.load_weights(savepath)
```