UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA

LUCAS BEZERRA DE ALMEIDA MÉLO

Caracterização de Caos Determinístico em Laser de Modos Travados em Fibra de Yb e Otimização de Fase Espectral para Absorção de Pulsos Banda Larga por Conjunto de Átomos Banda Estreita

Recife

2023

LUCAS BEZERRA DE ALMEIDA MÉLO

Caracterização de Caos Determinístico em Laser de Modos Travados em Fibra de Yb e Otimização de Fase Espectral para Absorção de Pulsos Banda Larga por Conjunto de Átomos Banda Estreita

Tese apresentada ao Programa de Pós Graduação em Física da Universidade Federal de Pernambuco como requisito parcial à obtenção do título de Doutor em Física. Área de concentração: Ótica

Orientador: Marcio Heraclyto Gonçalves de Miranda

Coorientador: Daniel Felinto Pires Barbosa

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

M528c Mélo, Lucas Bezerra de Almeida
Caracterização de caos determinístico em laser de modos travados em fibra de Yb e otimização de fase espectral para absorção de pulsos banda larga por conjunto de átomos banda estreita / Lucas Bezerra de Almeida Mélo. – 2023.
146 f.

Orientador: Marcio Heraclyto Gonçalves de Miranda.
Tese (Doutorado) – Universidade Federal de Pernambuco. CCEN, Física, Recife, 2023.
Inclui referências e apêndices.

1. Óptica. 2. Laser de modos travados. 3. Fibra ótica. I. Miranda, Marcio Heraclyto Gonçalves de (orientador). II. Título.

535.2 CDD (23. ed.) UFPE- CCEN 2023 - 19

LUCAS BEZERRA DE ALMEIDA MÉLO

CARACTERIZAÇÃO DE CAOS DETERMINÍSTICO EM LASER DE MODOS TRAVADOS EM FIBRA DE Yb E OTIMIZAÇÃO DE FASE ESPECTRAL PARA ABSORÇÃO DE PULSOS BANDA LARGA POR CONJUNTO DE ÁTOMOS BANDA ESTREITA

Tese apresentada ao Programa de Pós-Graduação em Física da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Física.

Aprovada em: 11/01/2023.

BANCA EXAMINADORA

_____
Prof. Marcio Heraclyto Gonçalves de Miranda
Orientador
Universidade Federal de Pernambuco

_____
Prof. Anderson Monteiro Amaral
Examinador Interno
Universidade Federal de Pernambuco

_____
Prof. José Wellington Rocha Tabosa
Examinador Interno
Universidade Federal de Pernambuco

_____
Prof. Lázaro Aurélio Padilha Junior
Examinador Externo
Universidade Estadual de Campinas

_____
Prof. Luis Gustavo Marcassa
Examinador Externo
Universidade de São Paulo

RESUMO

Apresentamos uma caracterização do caos determinístico no laser de modos travados em fibra dopada com Yb, identificando traços experimentais e demonstrando caos unimodal nesse sistema. Propomos um ajuste para o mapa de retorno caótico que é capaz de manifestar boa semelhança com os dados experimentais. Por fim, vemos que é possível reconhecer transições na rota para o caos utilizando a potência média do laser, revelando o caráter global do regime não linear. Após isso, abordamos outro problema no qual otimizamos a fase espectral de um pulso de laser ultra rápido para uma absorção sequencial de dois fótons em um meio atômico denso. Utilizamos um modulador espacial de luz para aplicar as mudanças de fase espectral e um algoritmo genético para auxiliar na determinação da alteração de fase espectral ótima. Ao longo do estudo quebramos o problema em várias etapas simples, permitindo o entendimento da otimização como uma superposição de efeitos e trazendo melhorias na absorção que superaram 2500%.

Palavras-chaves: laser de modos travados; fibra ótica; itérbio; caos determinístico; absorção ressonante de dois fótons; fase espectral; algoritmo genético; otimização.

ABSTRACT

We present a characterization for deterministic chaos in a mode locked Yb doped fiber laser, identifying experimental traces and demonstrating unimodal chaos in this system. We offer a curve to fit the chaotic return map that shows good resemblance with the experimental data. Lastly, we display how it is possible to identify transitions in the route to chaos using only the average laser power, revealing the global effects of the non-linear behavior. After that, we approach another problem in which we optimize the spectral phase of a ultrafast laser pulse for a two-photon sequential absorption in an dense atomic ensemble. We utilize a spatial light modulator to apply the spectral phase changes and a genetic algorithm as an auxiliary tool in determining the optimal spectral phase alterations. During the studies we break the problem in simple steps, allowing the understanding of the optimization as a superposition of effects and achieving improvements in absorption that surpass 2500%.

Keywords: mode mocked laser; optical fiber; ytterbium; deterministic chaos; resonant two-photon absorption; spectral phase; genetic algorithm; optimization.

LISTA DE ABREVIATURAS E SIGLAS

R^2	Coeficiente de Determinação
ADF	Absorção de Dois Fótons
AG	Algoritmo Genético
ASDF	Absorção Sequencial de Dois Fótons
DO	Densidade Ótica
LTF	Limitado por Transformada de Fourier
MEL	Modulador Espacial de Luz
MF	Máscara de Fase
NRFI	Não Ressonante com os Fótons Individuais
RF	Rádio-Frequência
RFI	Ressonante com os Fótons Individuais
Yb	Itérbio

SUMÁRIO

1 INTRODUÇÃO

Ao longo deste trabalho serão apresentadas duas linhas de pesquisa. Iniciamos uma caracterização da rota para o caos em um laser de modos travados em fibra de Yb, demonstrando caos unimodal para esse sistema. Em seguida, iremos trabalhar em uma simulação do problema de otimização de fase espectral para a absorção de um pulso em uma transição de dois fótons numa amostra densa. Essa mudança drástica no perfil da pesquisa foi necessária devido a pandemia do COVID-19, tendo em vista que os laboratórios estavam fechados durante o período de contenção da doença.

1.1 CARACTERIZAÇÃO DA ROTA PARA O CAOS NO LASER DE MODOS TRAVADOS EM FIBRA DE YB

Desde a primeira proposta de caos determinístico em 1963, por Lorenz (1), a compreensão deste problema, que se diferencia de simples ruído aleatório, avançou de diferentes formas (2). Apesar desse avanço contar com um bom número de demonstrações experimentais, explorar caos determinístico em sistemas físicos revela alguns desafios particulares devido as instabilidades naturais de qualquer experimento. Além disso, é comum que as dificuldades práticas impossibilitem uma abordagem tão sistemática quanto em um exploração teórica.

Dentre os sistemas utilizados para estudar o regime caótico, lasers estão continuamente presentes (3, 4, 5, 6, 7, 8). Ao longo da construção e caracterização do nosso laser de modos travados em fibra de Yb, percebemos uma grande variedade de regimes dinâmicos (9). De fato, a variedade vai muito além dos interesses deste trabalho e do que demonstramos. Para dar alguns exemplos, a literatura mostra a existência de *rogue waves* (10, 11), formação de pulsos *noise-like* (12, 13, 14), explosão de sólitons (15) e formação de moléculas solitônicas (16). Por um lado, nosso interesse na construção do laser era de operá-lo em regime estável. Por outro, ao longo da caracterização percebemos que as dinâmicas típicas da rota para o caos eram facilmente acessadas e que precisaríamos observá-las com mais atenção, a fim de evitá-las. Contudo, não encontramos na literatura análises experimentais sistemáticas do caos nesse sistema, demonstrando os traços de comportamento caótico nos instrumentos tipicamente utilizados na caracterização de um laser, para compararmos com os nossos dados. Por esse motivo, iniciamos a investigação que será apresentada aqui.

Visto que nosso sistema opera em modos travados, o pulso de femtossegundo gerado em nosso laser dificulta análises diretas do comportamento dinâmico. Porém, a literatura mostra como é possível entender a rota para o caos apenas com uma abordagem de recorrência (17, 2). Com isso, analisamos os dados experimentais observando a informação visível no osciloscópio, ou seja, a maneira como a energia de um pulso se relaciona com a energia do pulso seguinte. Isso nos permite observar caos unimodal nesse sistema, apesar de sua complexidade, obter informações suficientes para calcular o coeficiente de Lyapunov (uma maneira de quantificar o regime caótico) e encontrar um ajuste que nos dá uma boa correspondência com os dados.

Em seguida, partimos para correlacionar o mecanismo de travamentos de modos do laser, a saber, a rotação não linear de polarização, com a leitura da potência média. Mostramos que, embora a potência total do laser se comporte de forma linear, a transmissão da potência média não apenas reflete o comportamento da rotação não linear da polarização como também é capaz de marcar as transições de rota para o caos. Ter uma medida tão simples como ferramenta para identificar transições dinâmicas como esta pode facilitar bastante a manutenção do laser para aplicações nas quais se requer um laser estável.

Dedicamos os Capítulos 2 e 3 para trabalhar esses tópicos. No Capítulo 2 falamos sobre as ferramentas de caracterização de rota para o caos que serão utilizadas no Capítulo 3, onde apresentamos a caracterização dos dados experimentais. Apresentamos brevemente o sistema em questão e então abordamos os dados experimentais na mesma sequência do capítulo anterior. Em seguida, tratamos do problema de identificação de rota para o caos usando a curva de transmissão, discutindo o paralelo entre a rotação não linear da polarização e a medida de potência média.

1.2 OTIMIZAÇÃO DE FASE ESPECTRAL PARA ABSORÇÃO DE PULSO EM TRANSIÇÃO DE DOIS FÓTONS RESSONANTE EM UMA AMOSTRA ATÔMICA DENSA

A capacidade de gerar pares de fótons através de conversão paramétrica descendente já se mostrou útil de muitas formas ao longo dos últimos anos (18, 19, 20). Atualmente, o nível de complexidade das aplicações está limitada devido à dificuldade probabilística de gerar um número alto de pares de fótons simultaneamente. O desenvolvimento de memórias quânticas traz a esperança de solucionar esse problema, dando a possibilidade de armazenar os fótons gerados pela conversão paramétrica descendente até que o número de pares de fótons desejado seja alcançado.

Porém, o desenvolvimento de memórias quânticas tem seus próprios desafios. Os fótons gerados por conversão paramétrica descendente são banda larga e os meios atômicos para armazenamento tem uma banda estreita. Dentre os progressos feitos na direção de armazenar pulsos de luz em meios atômicos densos (21, 22, 23, 24), em 2020 Carvalho et al. demonstraram que é possível melhorar a absorção de um pulso fraco em uma transição de dois fótons em cascata aumentando a área do pulso de controle (forte) (25). Embora factível e mensurável de forma consistente, a melhoria reportada foi de apenas 0.3% em relação a uma absorção linear simples.

Em 2001, Silberberg et al. demonstraram como é possível alcançar grandes aumentos numa absorção de dois fótons ressonante com um ajuste de fase espectral aplicada por um Modulador Espacial de Luz (MEL). A chave para isso foi entender a existência de uma interferência destrutiva, ao analisar o modelo da transição da perspectiva espectral. Aplicando a alteração, Silberberg et al. obtiveram um aumento de 600% no processo.

Embora o modelo proposto por Carvalho et al. apresente um nível de complexidade muito mais alto que o descrito por Silberberg et al., eles atendem o mesmo princípio de absorção de dois fótons. Dessa forma, iremos abordar o problema de forma similar. Vamos utilizar um MEL para alterar fase espectral do pulso a ser absorvido pela amostra densa. Entretanto, devido ao aumento da dificuldade em comparação ao problema do Silberberg et al., não podemos assumir uma abordagem analítica.

Como o número de parâmetros a serem otimizados é grande, vamos recorrer ao uso de um algoritmo adaptativo para encontrar as respostas ótimas, assim como tem sido feito por outros grupos (26, 27, 28, 29, 30). Dentre os algoritmos adaptativos, o Algoritmo Genético (AG) se destaca pela sua capacidade de encontrar máximos globais (31). Além disso, existem outros exemplos de uso do AG em problemas similares de otimização de fase espectral para aumento de fluorescência por absorção de um pulso de femtosegundo (32, 33).

Quanto ao problema físico, começaremos testando a nossa abordagem com o AG no problema do Silberberg et al. (34), cujos resultados já são conhecido, como uma simulação de referência. Em seguida, trabalharemos a absorção de um pulso por um único átomo sendo excitado por dois lasers simultaneamente. Depois, ainda com um átomo individual, vamos observar como é a interação com um pulso fraco que foi distorcido por uma amostra ressonante, também conhecido como pulso 0π. Após compreender todos esses resultados, aí sim entraremos no problema de absorção em uma amostra densa. Ao decorrer da discussão, veremos que os resultados de otimização podem fornecer melhorias acima de 2500%, dependendo das

circunstancias, além de demonstrar uma simplicidade notável, revelada pelo AG.

Para estes tópicos, iremos iniciar com o Capítulo 5 onde apresentaremos o AG e as adaptações necessárias para atendender nossas necessidades. No Capítulo 6, visitaremos o problema apresentado pelo Silberbeg et al.. Depois, no Capítulo 7, vamos explorar o problema de otimização de uma transição de dois fótons em cascata ressonante para um único átomo, incluindo o problema de absorção de um pulso que foi modificado por um meio ressonante. Finalmente, no Capítulo 8, vamos abordar o problema de otimização de fase espectral para um meio atômico denso.

2 CARACTERÍSTICAS BÁSICAS DE CAOS DETERMINÍSTICO

Previamente, durante o mestrado, tivemos a oportunidade de construir e caracterizar um laser de modos travados em fibra de Itérbio (Yb) (9). Ao longo da caracterização pudemos observar traços da dinâmica não linear e rota para o caos no sistema. No Capítulo 3 iremos apresentar uma caracterização mais completa e objetiva do regime multi-estável desse mesmo laser. Para isso, antes de abordar os resultados experimentais, vamos reservar o presente capítulo para falar sobre aspectos gerais de sistemas com caos unimodal (ou seja, sistemas cujo regime caótico pode ser descrito por uma função de um único máximo) que serão relevantes para nossa análise, a saber, vamos utilizar o mapa logístico para tratar de mapas de retorno, diagramas de bifurcação e coeficiente de Lyapunov.

2.1 MAPA LOGÍSTICO E ASSINATURAS BÁSICAS DE SISTEMAS CAÓTICOS

Em seu histórico artigo sobre estabilidade e periodicidade de sistemas dissipativos forçados (1), inicio do que hoje chamamos de Teoria do Caos, Edward Lorenz demonstrou como era possível prever (ou mapear) os máximos locais da solução do seu sistema de equações (nomeadas Equações de Lorenz) a partir do máximo local que o precedia. Lorenz então utilizou essa técnica para mapear os máximos da solução quando o sistema era configurado de forma a apresentar um comportamento que, à primeira vista, parecia aleatório ou ruidoso. De forma surpreendente, o que Lorenz descobriu e revelou foi que, na verdade, existia uma relação ainda determinística dentro daquela situação aparentemente imprevisível. Tal era a previsibilidade da relação entre os valores dos máximos locais que era perfeitamente possível mapeá-los através de uma função! Esse mapa recebeu o nome de Mapa de Lorenz, em merecida homenagem.

Mais tarde, Robert May utilizou a ideia de mapas para mostrar que era possível entender a rota para o caos sem sequer especificar o sistema dinâmico (17). Através do Mapa Logístico, que relaciona o estado de uma população de animais entre uma geração (x_n) e a próxima (x_{n+1}), descrito por

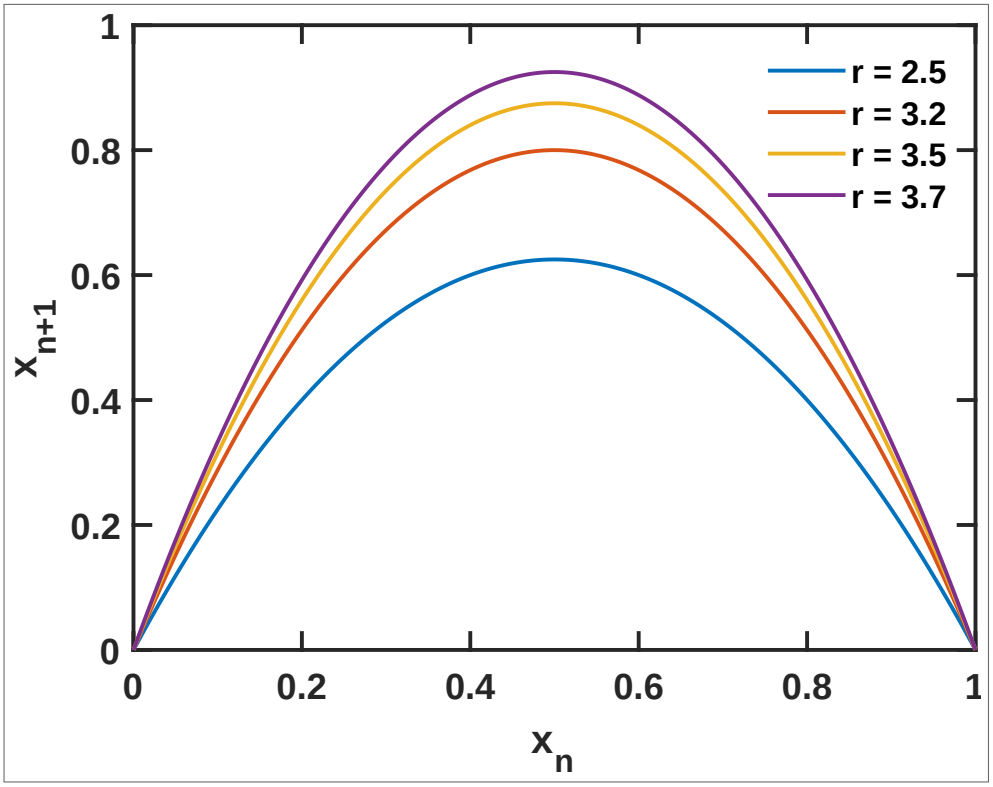
$$x_{n+1} = r x_n (1 - x_n), \tag{2.1}$$

r sendo a taxa de crescimento da população. May também foi capaz de obter multi-estabilidade e caos, cujas características de interesse iremos demonstrar aqui. O grande diferencial da

abordagem de May em relação à Lorenz é a capacidade de determinar o estado de um sistema apenas através de parâmetros e tempos discretos. Enquanto Lorenz foi capaz de capturar o Mapa de Lorenz apenas quando o sistema estava no regime caótico, May demonstrou que podemos ir além. Basta saber como a função do mapa se relaciona com o parâmetro de controle para que sejamos capazes de entender a estabilidade do sistema em qualquer momento, não apenas no regime caótico.

Se observarmos o Mapa Logístico, na Figura 1, podemos notar como a curva se relaciona com o parâmetro r. A curva em si não apresenta nenhuma característica excepcional, sendo apenas a parábola descrita pela Equação 2.1. Porém, quando a utilizamos para prever as populações de cada ano consecutivo, um padrão surpreendente surge ao variarmos o valor de r. Na Figura 2 utilizamos a mesma condição inicial, $x_0 = 0.5$ para entendermos o impacto do parâmetro r na previsão dos pontos. Para valores mais baixos de r, como em (a), temos uma situação estável. Após o transiente, o valor se repete em cada iteração. Aumentar o valor de r não tem impacto na estabilidade enquanto $r < 3$. A partir de $r = 3$ temos uma alteração no comportamento, demonstrada em (b). Ainda há estabilidade, mas com um período dobrado; o valor se repete a cada duas iterações. A resposta continua sendo essa até que cruzamos $r \approx 3.4$, quando novamente temos um dobramento de período, visto em (c). Essa sequência de dobramentos de período continua infinitamente, porém em intervalos de r cada vez menores. Avaliando numericamente o valor de r para cada dobramento no mapa logístico, podemos identificar que a transição para período-2 acontece para $r = 3$, período-4 em $r \sim 3.4$, período-8 em $r \sim 3.5$, período-16 em $r \sim 3.56$, período-32 em $r \sim 3.569$, e assim por diante, tendendo a um número muito específico: $r = 3.569946...$. Ao ultrapassar esse valor temos o surgimento de um comportamento não-periódico (matematicamente, período infinito), visível em (d). Note que essa situação não aparenta ter um padrão. E, comparando com as figuras anteriores, vemos que o número de iterações para remover o comportamento transiente é muito curto (por volta de 5 iterações) ou seja, a ausência de padrão não tem origem em comportamento transiente. De fato, é possível verificar esse comportamento aperiódico em qualquer trecho de iterações, deixando claro que se trata de comportamento caótico determinístico, não de transientes nem oscilações de período longo. Com isso fica claro que o mapa é suficiente para demonstrar o comportamento do sistema, da estabilidade ao caos, sem a necessidade de especificar a equação dinâmica. O mapa logístico tem muitas propriedades, porém algumas são de pouca relevância para este trabalho. Iremos nos ater aos tópicos do nosso interesse, mas o leitor pode encontrar mais detalhes com um tratamento bastante acessível em (2).

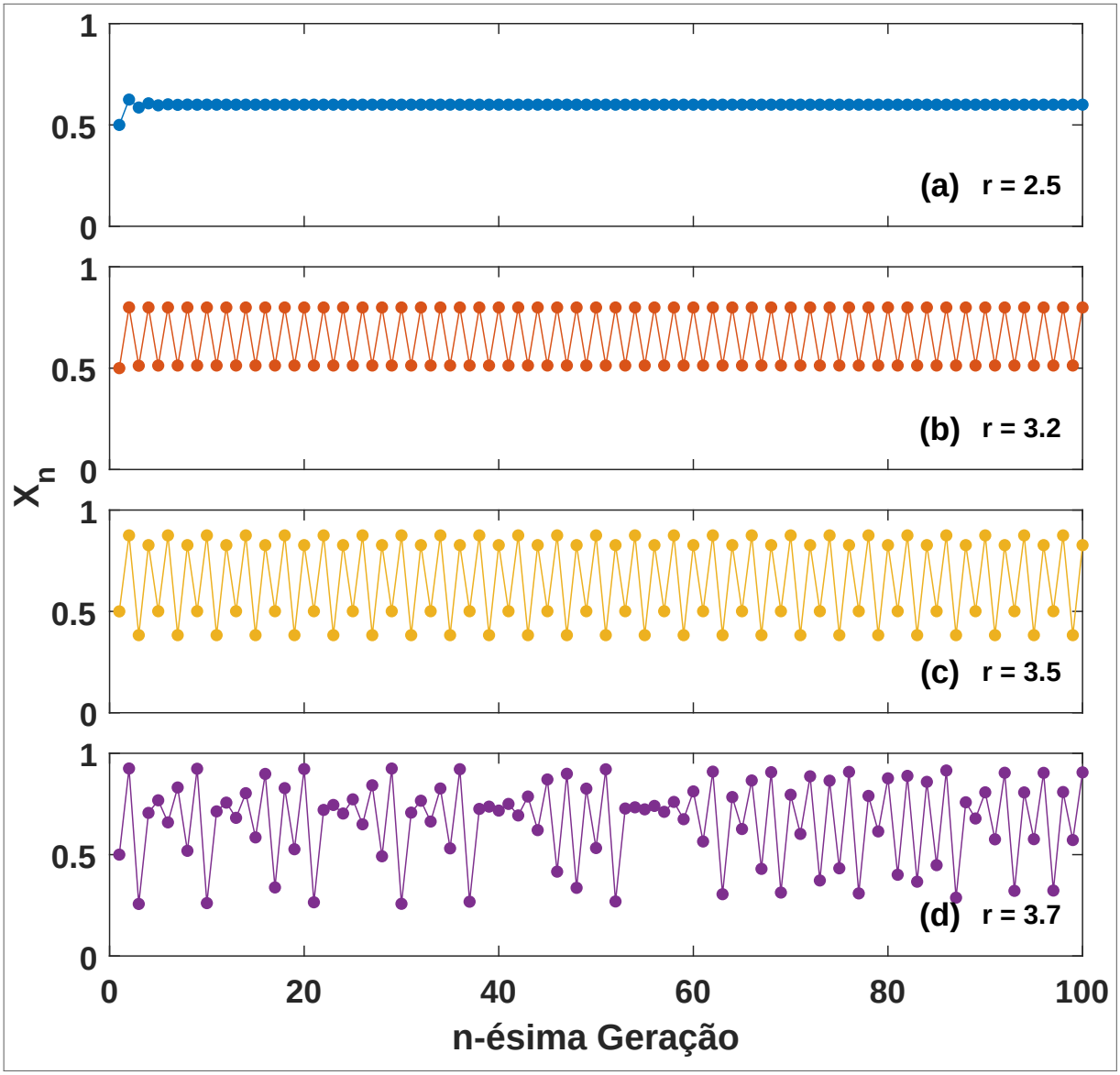
Figura 1 – Mapa Logístico, dado pela Equação 2.1, para alguns valores representativos de r.



Fonte: Elaborado pelo autor (2022)

Uma maneira alternativa de visualizar a relação dos valores previstos pela Equação 2.1 é através do Diagrama de Bifurcação. Na Figura 3 (a) construímos o Diagrama de Bifurcação utilizando 1000 pontos, após o transiente, gerados pela equação do Mapa Logístico para cada valor de r. Dessa forma somos capazes de visualizar rapidamente quantos valores de x_n são permitidos, dado o valor do parâmetro r. A fim de evitar o comportamento transiente, que confundiria a compreensão do gráfico, os 100 primeiros pontos, para cada valor de r, foram removidos. A vantagem dessa forma de apresentar os pontos é que ela nos permite entender rapidamente que tipo de estabilidade teremos para um dado valor de r, sem a necessidade de observarmos a sequencia de iterações, como na Figura 2. Aqui, os pontos de transição de estabilidade ficam claros, assim como o comportamento caótico. Note que dentro dos intervalos caóticos os pontos do diagrama chegam a formar regiões continuas totalmente preenchidas. Isso porque a característica aperiódica do caos permite que todos os pontos que poderiam ser acessados sejam, de fato, acessados se esperarmos tempo suficiente. A Figura 3 (b) mostra o valor do Coeficiente de Lyapunov para cada valor de r e pode ser comparado diretamente com a Figura 3 (a). O Coeficiente de Lyapunov é uma medida de quão caótico é o sistema. O valor indica a o tipo de comportamento exponencial que as soluções terão no espaço de

Figura 2 – Pontos gerados em cada iteração da Equação 2.1 para os valores de r mostrados na Figura 1. Apenas os pontos são significativos nos gráficos; a linha serve apenas para ajudar o leitor a identificar a ordem dos pontos. Os quatro gráficos partem da mesma condição inicial $x_0 = 0.5$. (a) demonstra o comportamento estável. (b) uma situação de bi-estabilidade. (c) ainda é periódico, mas com um ciclo de 4 valores diferentes. (d) é uma situação caótica. O leitor pode verificar que o padrão dos pontos não se repete em (d), mesmo quando são parecidos.

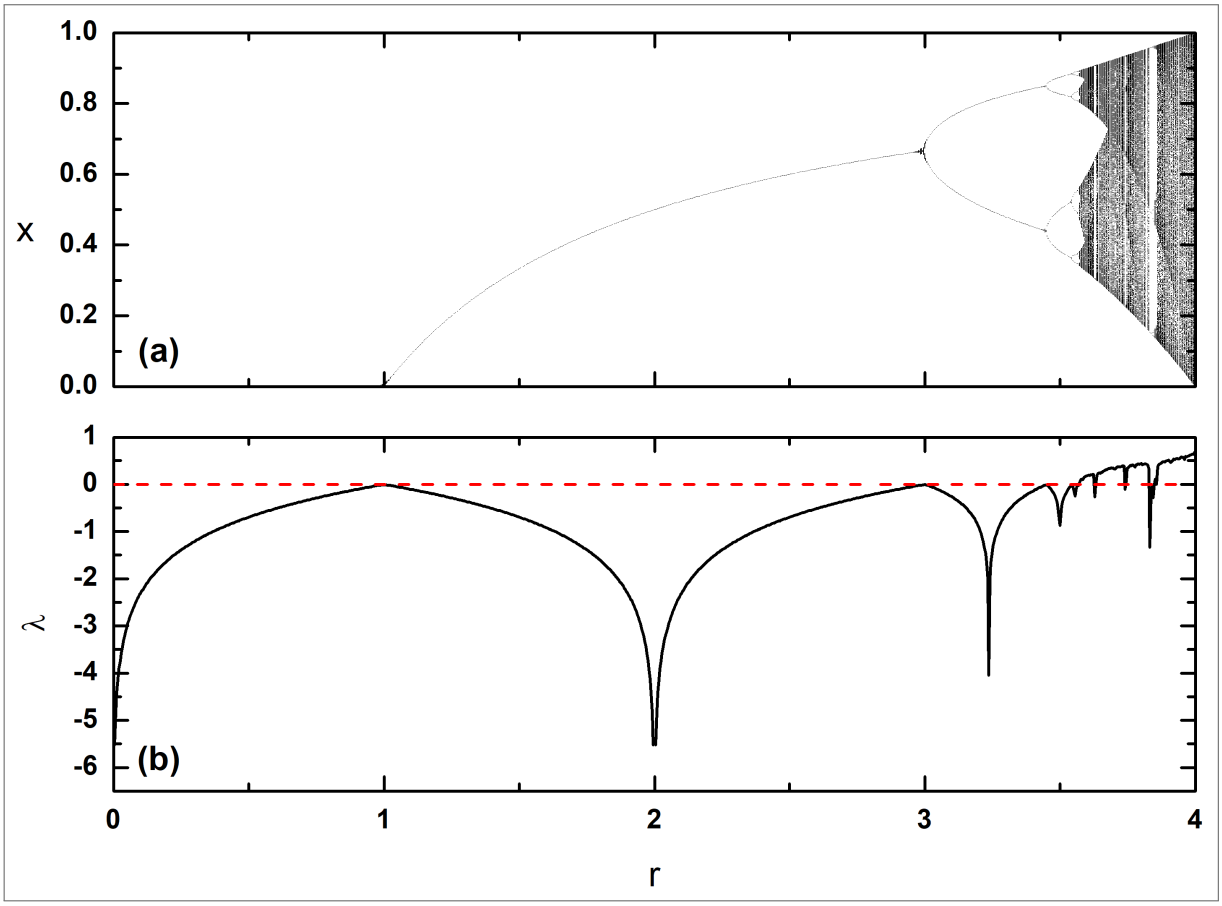


Fonte: Elaborado pelo autor (2022)

fase. Valores negativos indicam uma tendência de se aproximarem exponencialmente e valores positivos indicam uma tendência a se afastarem exponencialmente. Falaremos mais sobre essa quantidade. Mas faremos isso em uma seção exclusiva, por se tratar de uma característica quantitativa do caos. Por ora, terminaremos de explicar as propriedades qualitativas.

Até agora nós utilizamos a Equação 2.1 para gerar os pontos dos nossos gráficos. Mas numa situação experimental tipicamente temos os pontos antes de sermos capazes de entender qual é a relação entre eles. Esse será o nosso contexto no capítulo seguinte. Sendo assim, é

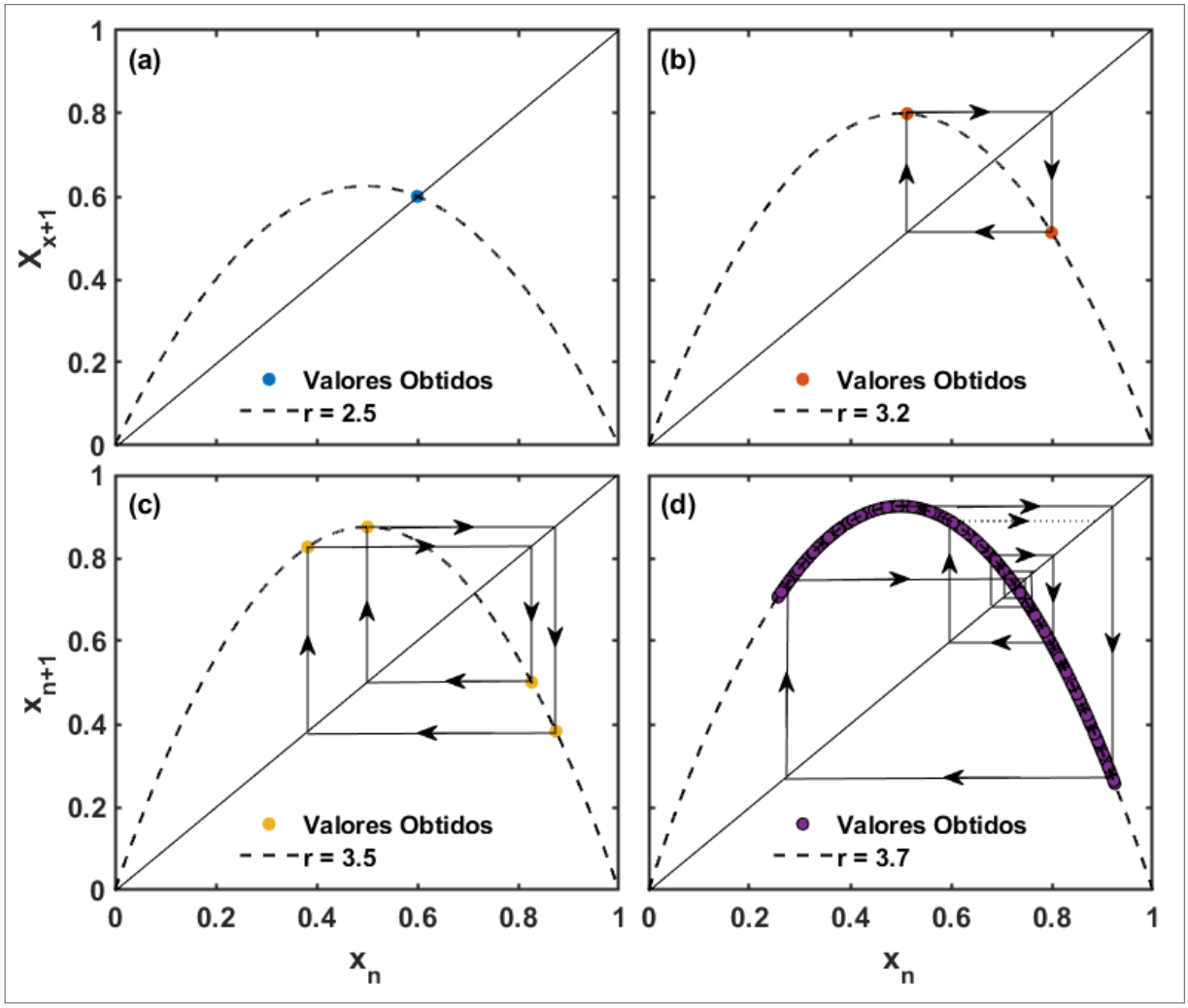
Figura 3 – (a) Diagrama de Bifurcação do Mapa Logístico. (b) Coeficiente de Lyapunov em função de r. A linha vermelha apenas marca o zero, que determina a mudança de estabilidade entre periódica e caótica. Nestes gráficos foram utilizados 1000 pontos previstos pela Equação 2.1 para cada valor de r. Para evitar as oscilações transientes, os 100 primeiros pontos foram ignorados. Esses diagrama nos permite entender prontamente que tipo de estabilidade é acessada para um determinado valor de r e relacionar diretamente com o valor do coeficiente de Lyapunov calculado sob as mesmas condições.



Fonte: Elaborado pelo autor (2022)

interessante entendermos como podemos obter a Equação 2.1 a partir dos pontos que ela gera. O Mapa de Retorno mostra como o ponto da iteração x_n se relaciona com o da iteração x_{n+1}, como pode ser visto na Figura 4. Assim como na Figura 3 (a), foram utilizados 1000 pontos, após os 100 primeiros, para evitar o comportamento transiente, em cada valor de r. Em cada um dos gráficos foi colocado a curva do Mapa Logístico correspondente, para ajudar o leitor a entender como os pontos se relacionam com as curvas, e uma reta que mostra a linha de $x_n = x_{n+1}$, que será utilizada para demonstrar o diagrama de órbitas (maneira iterativa de entender a sequência de pontos no mapa de retorno). Na situação estável, (a), vemos que todos os 1000 pontos tem exatamente o mesmo valor, com $x_n = x_{n+1}$, ficando exatamente na intersecção do mapa logístico com a reta. O comportamento estável leva todas as iterações a terem o mesmo valor, como vimos anteriormente. Essa é a única situação que permite

Figura 4 – Mapa de Retorno utilizando os mesmos valores da Figura 1. Note que os pontos do gráficos sempre caem sobre a curva que os descreve, em preto tracejado. Em (a) temos a situação estável, onde $x_n = x_{n+1}$ em todas as iterações. Essa é a única situação que permite $x_n = x_{n+1}$, visto que o valor se repete indefinidamente. (b) tem dois pontos entre os quais os valores alternam; (c) tem quatro. (d) demonstra uma situação diferente dos demais, o regime caótico faz com que os pontos fiquem espalhados ao longo de um intervalo contínuo de valores, redesenhando parte da curva do Mapa Logístico. Para essa figura foram utilizados 1000 pontos, partindo do 101º para evitar o comportamento transiente, em cada gráfico.



Fonte: Elaborado pelo autor (2022)

$x_n = x_{n+1}$ pois, uma vez sendo iguais, serão iguais em todas as iterações. Em (b) temos dois pontos, dada a situação de período dobrado (ou período-2), onde temos $x_n = x_{n+2}$. Para entender como os dois pontos surgem, tome, por exemplo, um dos pontos da Figura 2 (b) que tem valor próximo a 0.5 para ser o x_n. Então, seguindo a sequência dos pontos, x_{n+1} tem valor próximo a 0.8. Isso nos dá o ponto no mapa de retorno da Figura 4 (b) com $x_n = 0.5$. Para o outro ponto, basta voltarmos para a Figura 2 (b) e escolhermos um dos pontos com valor próximo a 0.8 para ser x_n. Novamente, seguindo a sequência de pontos vemos que $x_{n+1} = 0.5$. Isso nos dá o segundo ponto na Figura 4 (b). Fazendo isso para

todos os pontos da Figura 2, formamos cada gráfico respectivo na Figura 4. Porém, é possível também saber a sequência de pontos utilizando apenas o Mapa de Retorno, sem recorrer a Figura 2. Exemplificando ainda com a Figura 4 (b), partindo novamente do ponto com $x_n \approx 0.5$, seguimos a linha horizontal em direção à reta $x_n = x_{n+1}$ (seguindo a seta). Da reta, partimos na direção vertical, retornando para a curva do Mapa Logístico e chegando ao ponto com $x_n \approx 0.8$ (sempre seguindo a seta). Esse procedimento é capaz de nos dizer a sequência de pontos dos nossos dados. Perceba que repetindo o processo, agora partindo de $x_n \approx 0.8$, retornamos ao ponto com $x_n \approx 0.5$. Esse processo serve para todos os mapas de retorno. Indo para a Figura 4 (c) temos mais um dobramento de período (ou período-4), com $x_n = x_{n+4}$. Note que o diagrama de órbitas, apesar de mais longo, ainda é cíclico. Finalmente em (d) temos caos, no qual o período é virtualmente infinito. A forma de obter o gráfico não muda em relação à forma explicada para período-2. O que mudou foi o comportamento do sistema. A relação de igualdade entre os pontos não é possível nesse estágio, por isso os valores variam sem nunca se repetir e eventualmente formam um contínuo de pontos. Isso fica claro quando tentamos desenhar o diagrama de órbitas. Note que o diagrama continua indefinidamente, nunca fechando um ciclo. Esse comportamento é tal que os pontos são capazes de redesenhar parte da curva do Mapa Logístico. Se não tivéssemos a relação da Equação 2.1, seríamos capazes de obtê-la a partir dos pontos em (d). Isso é de grande utilidade para uma situação experimental, visto que não é possível encontrar uma relação a partir de (a) nem de (b). E mesmo a partir de (c) ainda seria difícil de chegar a uma descrição precisa.

Todas as formas de visualização apresentadas nesta seção são úteis para identificar rota para o caos em um experimento ou em um conjunto de dados. Entretanto, mesmo sendo capazes de relacionar os dados com uma mapa, como na Figura 4 (d), temos apenas uma noção qualitativa do regime caótico. Agora passamos discutir uma forma quantitativa de medir o caos no sistema: o coeficiente de Lyapunov.

2.2 COEFICIENTE DE LYAPUNOV

Considere que x_n é a condição de um sistema não linear, que pode ser mapeado, tal que n indica a n-ésima iteração do mapa. Dessa forma, x_0 é a condição inicial do sistema. Tome uma condição inicial próxima, descrita por $x_0 + \delta_0$, onde δ_0 é um valor tão pequeno quanto desejado. Se, ao longo das iterações, formos capazes de descrever a evolução da separação dessas duas soluções através de $|\delta_n| \approx |\delta_0| e^{n\lambda}$, chamamos então λ de coeficiente (ou expoente)

de Lyapunov. Quando λ é negativo, as soluções se tornam mais próximas a cada iteração; o que caracteriza o regime periódico. Se λ for positivo, as soluções se distanciarão infinitamente; caracterizando o regime caótico. Isso, por si só, já nos dá uma avaliação quantitativa da estabilidade do sistema. Mas é possível ter uma descrição mais prática.

Para avaliar o regime caótico em dados experimentais, podemos partir da definição de λ e chegar a uma formulação que permite avaliar o mapa de retorno diretamente. Note que podemos escrever λ como

$$\lambda \approx \frac{1}{n} ln \left| \frac{\delta_n}{\delta_0} \right|. \tag{2.2}$$

Lembramos então que $x_{n+1} = f(x_n) = f(f(x_{n-1})) = \cdots = f(f(f(\cdots f(x_0) \cdots)))$. Para facilitar o entendimento, vamos utilizar a notação $f_n(x_0)$ para indicar que o mapa foi aplicado n vezes, de forma que $x_{n+1} = f_{n+1}(x_0)$. Isso nos permite escrever o n-ésimo ponto do mapa em termos do primeiro. Com isso, podemos relacionar a evolução da separação, δ_n, entre os pontos x_0 e $x_0 + \delta_0$ como $\delta_n = f_n(x_0 + \delta_0) - f_n(x_0)$. Utilizando essa descrição na Equação 2.2, temos:

$$\lambda \approx \frac{1}{n} ln \left| \frac{f_n(x_0 + \delta_0) - f_n(x_0)}{\delta_0} \right|. \tag{2.3}$$

Tomando o limite $\delta_0 \to 0$, temos a definição da derivada:

$$\lambda \approx \lim_{\delta_0 \to \infty} \frac{1}{n} ln \left| \frac{f_n(x_0 + \delta_0) - f_n(x_0)}{\delta_0} \right|; \tag{2.4}$$

$$\lambda \approx \frac{1}{n} ln \left| \frac{d}{dx} f_n(x_0) \right|; \tag{2.5}$$

$$\lambda \approx \frac{1}{n} ln \left| \frac{d}{dx} f(f(f(\cdots f(x_0) \cdots))) \right|; \tag{2.6}$$

$$\lambda \approx \frac{1}{n} ln \left| \frac{df(x_{n-1})}{dx} \frac{df(x_{n-2})}{dx} \cdots \frac{df(x_1)}{dx} \frac{df(x_0)}{dx} \right|; \tag{2.7}$$

$$\lambda \approx \frac{1}{n} \sum_{i=0}^{n-1} ln \left| \frac{d}{dx} f(x_i) \right|. \tag{2.8}$$

Como essa expressão visa avaliar a estabilidade do sistema, é importante que ela seja avaliada para $n \to \infty$:

$$\lambda \approx \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} ln \left| \frac{d}{dx} f(x_i) \right|. \tag{2.9}$$

O coeficiente de Lyapunov é dado pela Equação 2.9 se o limite com $n \to \infty$ existir. λ descrito dessa forma é especialmente útil quando estamos tratando com dados experimentais, pois ele torna possível quantificar a estabilidade do sistema a partir dos n pontos discretos obtidos através do experimento, bastando apenas que tenha-se a curva que descreve o comportamento do sistema. E, como vimos antes, uma das vantagens do caos unimodal é que podemos obter essa curva também a partir dos pontos experimentais se o sistema atingir o regime caótico (período infinito). Aplicaremos essa técnica no capítulo seguinte.

Por fim, podemos usar a Equação 2.9 para verificar quantitativamente a estabilidade do Mapa Logístico em função de r. A Figura 3 (b) mostra o valor de λ para cada valor de r utilizado para gerar o diagrama de bifurcação em (a). A característica mais evidente é a transição de sinal que marca o a mudança entre comportamento periódico e caótico. Comparando com o diagrama de bifurcação, vemos que o coeficiente de Lyapunov marca inclusive as janelas de estabilidade, dentro da região caótica, com uma mudança repentina de sinal. Uma outra característica marcante é como o valor de λ se aproxima de zero em cada uma das mudanças de período. Essa marca pode ser útil para identificar quão distante um certo regime se encontra de uma mudança.

Com isso, temos todas as ferramentas que iremos utilizar para caracterizar o regime caótico no nosso experimento. Os traços de dinâmica não linear que observamos para o Mapa Logístico são válidos para qualquer sistema que seja dissipativo o suficiente a ponto de gerar um mapa de retorno unidimensional com um único máximo, chamados de mapas unimodais, como o que apresentaremos no próximo capítulo.

3 REGIME MULTI-ESTÁVEL EM LASER DE MODOS TRAVADOS EM FIBRA DE YB

Passamos a verificar como a dinâmica não linear de multi estabilidade se manifesta em um laser de modos travados em fibra de Yb. Nosso propósito é demonstrar como somos capazes de identificar regimes dinâmicos variados mesmo quando não temos acesso às equações dinâmicas do laser, usando uma abordagem semelhante à apresentada no Capítulo 2. Veremos também como é possível identificar mudanças de estabilidade apenas com leituras de potência média, facilitando drasticamente a identificação de uma mudança de estabilidade.

3.1 CAOS DETERMINÍSTICO EM LASERS DE MODOS TRAVADOS EM FIBRA DE YB

O sistema em questão é o laser da Figura 5, que tivemos a oportunidade de montar e caracterizar em regime de operação estável durante o mestrado (9). Aqui nós apresentamos essas características de forma breve, uma vez que nosso foco agora é a caracterização do regime caótico. Caso o leitor esteja interessado em mais informações da caracterização e operação do laser, descrevemos todo o equipamento utilizado para a montagem, diagramas de níveis para operação laser, mecanismos de travamentos de modos, medidas e instrumentos utilizados para caracterização em (9).

Em operação de modos travados passivo estável, esse sistema apresenta comprimento de onda central em $\sim$1025nm, com $\sim$20nm de largura espectral, pulso com duração de $\sim$140fs, taxa de repetição de $\sim$130MHz e potência média de $\sim$80mW. Na Figura 5 (a) vemos que a cavidade é em anel e que o laser de bombeamento é de 975nm. Especificamente, é um diodo da empresa JDSU, modelo S30-7602-720 controlado por um controlador de corrente produzido pela Thorlabs, modelo LDC 240 C. Essa informação é relevante pois iremos tratar a corrente deste laser como parâmetro de varredura. A relação entre a corrente e a potência de bombeamento é 0.68 mW/mA. Dentro da cavidade, os instrumentos estão indicados pelos acrônimos correspondentes em inglês. O WDM é um multiplexador por divisão de comprimento de onda, o YDF é o meio de ganho em fibra dopada com Yb, o PBS é um divisor de feixe polarizador, o GP é um par de grades de difração e o FI é um isolador de Faraday. Fora da cavidade, BS é um divisor de feixe e D é um fotodetector de 1 GHz montado em nosso laboratório. O circuito da cavidade se inicia com o laser de bombeamento, que envia sinal de forma unidirecional para o WDM. Assim, a YDF é bombeada e a emissão laser (1025 nm) segue

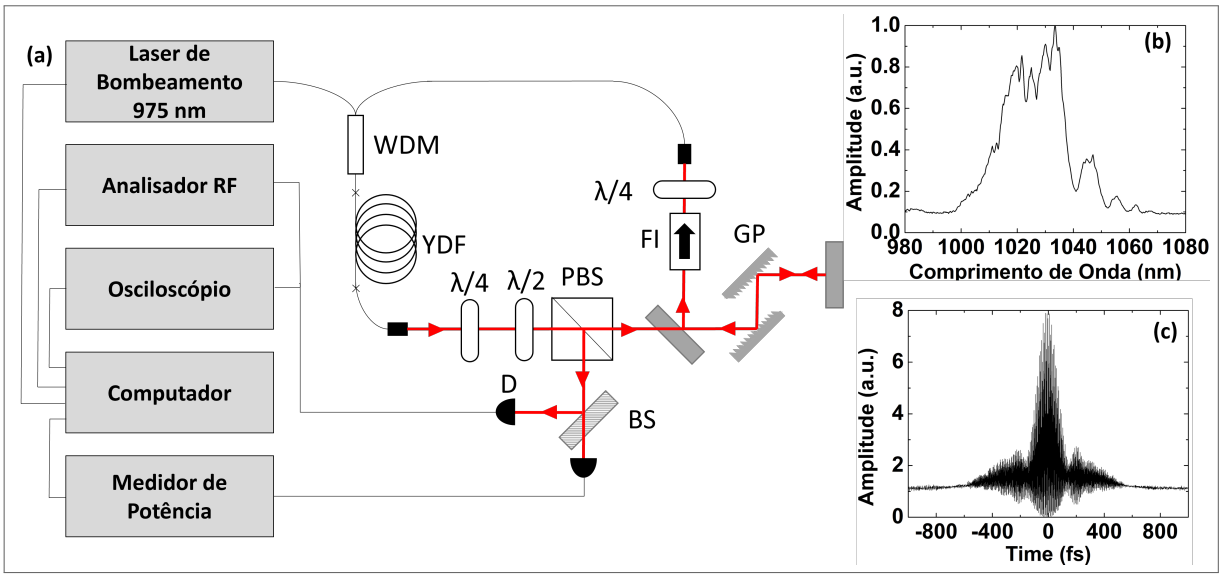
para o par de placas de onda, onde ajustamos a polarização do feixe para controle da potência de saída no PBS. O sinal transmitido segue então para o par de grades de difração, onde compensamos a dispersão normal que o feixe de 1025 nm sofre dentro da fibra de vidro. Após essa etapa o feixe é levemente deslocado verticalmente para atingir o espelho semi circular, indo então em direção ao isolador de Faraday (que impede qualquer feixe contra propagante dentro da cavidade). Por fim, o feixe passa pela placa de $\lambda/4$ com o objetivo de introduzir elipticidade na polarização do feixe antes de ser reintroduzido na fibra ótica, realimentando o meio de ganho. Sem um feixe de polarização elíptica não é possível obter o travamento de modos passivo, pois o efeito Kerr, nesse laser, atua de forma que a polarização do feixe gira no sentido anti-horário devido a uma diferença no índice de refração de cada componente do feixe. Essa diferença é dada pela expressão

$$\Delta n^{(i)} = \frac{2}{3} n_2 I^{(j)}, \qquad i, j = x, y; \quad i \neq j. \tag{3.1}$$

$\Delta n^{(i)}$ é a diferença de índice de refração induzida na direção i, n_2 é o índice de refração não linear e $I^{(j)}$ é a intensidade do feixe na polarização j. Essa diferença no índice de refração irá atrasar uma componente do feixe em relação a outra, de forma que o resultado final será um giro anti-horário de polarização se o feixe tiver polarização elíptica. Como o efeito é não linear (ou seja, depende da intensidade do feixe), um feixe com uma determinada intensidade não sofre o mesmo giro de polarização que um feixe com o dobro da sua intensidade, por exemplo. Esse efeito combinado à seleção de polarização dada pelo PBS forma o mecanismo de absorvedor saturável, necessário para o travamento de modos passivo, no laser em fibra de Yb. Retornando à Figura 5, (b) mostra o espectro ótico do laser quando operando em modos travados estável. (c) mostra a autocorrelação interferométrica, que tem a aparência típica para laser de Yb (35, 36) (com um perfil de aparência gaussiana no centro e alguns lóbulos laterais). Com ela podemos saber a largura temporal do pulso e verificar se a compressão do pulso está sendo ótima. Na autocorrelação interferométrica, a existência de *chirp* destruiria o padrão de interferência que vemos na 5 (c) e converteria-o em um padrão de autocorrelação de intensidade. Ou seja, a ausência de um patamar acima do nível 1 indica que o pulso está comprimido ao limite. A compressão do pulso dentro da cavidade é necessária nesse sistema pois, ao se propagar pela fibra, o pulso experimenta uma dispersão de velocidade de grupo normal, gradualmente separando as frequências do pulso e alargando-o. Essa dinâmica de alargamento e compressão dentro da cavidade é comum em um laser em fibra ótica de Yb

(37, 38), em contraste com o laser de Er, que tipicamente opera sem compressão (39, 8). Combinando as informações apresentadas, vemos que o laser em questão opera em condições típicas para o uso em laboratório.

Figura 5 – (a) Laser de modos travados em fibra de Yb e o sistema de aquisição de dados. WDM é o multiplexador por divisão de comprimento de onda. YDF é o meio de ganho, em fibra de Yb. PBS é o divisor de feixe polarizador. GP é o par de grades de difração. FI é o isolador de Faraday. D é o fotodetector. (b) Espectro ótico em modos travados estável. (c) Autocorrelação interferométrica do pulso em operação estável. (b) e (c) foram capturados logo após o PBS. A taxa de repetição do laser é de ∼130MHz com potência média de ∼80mW.



Fonte: Elaborado pelo autor (2022)

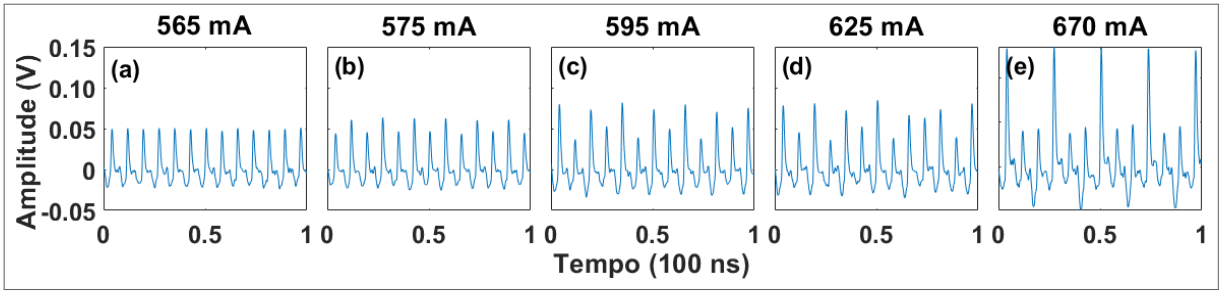
Porém, assim como vimos no Capítulo 2 que um sistema que apresenta condições estáveis não implica que está à prova de multi estabilidade, variando-se os parâmetros, o laser da Figura 5 também pode apresentar rota para o caos. Para este laser, os parâmetros são a polarização, a dispersão total e a intensidade do pulso. A placa de $\lambda/4$ próximo ao FI controla a polarização do pulso dentro da fibra ótica. O GP controla a dispersão total. O par $\lambda/4+\lambda/2$ próximo ao PBS e a corrente de bombeamento controlam a intensidade do pulso. Com tantos parâmetros à disposição, é possível acessar várias condições de estabilidade diferentes (9). Mas iremos nos concentrar em utilizar a corrente como parâmetro de varredura, mantendo os outros fixos. Desta forma é possível manter um controle mais preciso, visto que a corrente do bombeamento é facilmente controlável pelo computador. Para isso, desenvolvemos um programa em LabVIEW para controlar todo o sistema de medidas (9).

Dentro das possibilidades de parâmetros fixos, uma condição se destacou pelo comportamento da estabilidade sob variação da corrente do bombeamento. Iremos apresentar os dados obtidos nesta varredura, pois ela apresenta caos unimodal. Começando pela série temporal,

a Figura 6 apresenta algumas amostras de janelas capturadas no osciloscópio para diferentes valores de corrente do bombeamento. Embora as imagens mostrem curvas contínuas, apenas o valor de pico representa uma quantidade relevante. Visto que o pulso em questão é da ordem de 100fs, a forma das curvas na Figura 6 não representam a forma do pulso. É apenas uma resposta eletrônica ao impulso recebido no fotodetector D. Porém, a energia do pulso é proporcional ao valor dos picos da figura. Dessa maneira, nós temos uma série temporal muito similar ao apresentado na Figura 2, onde apenas os valores dos picos representavam quantidades relevantes. Com isso, iremos proceder com uma análise similar à mostrada anteriormente.

Partindo de uma condição estável em ∼565mA, vemos que a serie temporal em (a) não apresenta variação nos valores dos picos. Mas a estabilidade muda para valores de corrente mais altas, como visto em (b). Em ∼575mA, temos dois valores de pico alternando periodicamente, configurando período-2. Seguindo para (c), com ∼595mA, temos quatro valores de pico, ou seja, período-4. Em (d), ∼625mA, finalmente chegamos a um ponto onde os valores dos picos são aperiódicos. O laser apresenta caos determinístico. Note que temos a mesma sequência apresentada na Figura 2. Porém, diferente do caso numérico apresentado no Capítulo 2, observar infinitos dobramentos de período num sistema experimental não é possível devido a instabilidades e ruídos no sistema. Ainda assim, podemos seguir em frente e observar período-3 em (e), com ∼670mA. Esse regime é marcado por um pico de grande intensidade, seguido por dois de intensidade menor. Iremos explorar isso na sessão seguinte. Por ora, vamos nos concentrar no mapeamento do caos de acordo com o proposto no Capítulo 2.

Figura 6 – Série temporal para valores crescentes de corrente. (a) ∼565mA. Um único valor de pico é observado. Estabilidade regular. (b) ∼575mA. Dois valores de pico. Período-2 (bi-estabilidade). (c) ∼595mA. Quatro valores de pico. Período-4. (d) ∼625mA. Valores dos picos não se repetem. Caos determinístico. (e)∼670mA. Três valores de pico. Período-3.



Fonte: Elaborado pelo autor (2022)

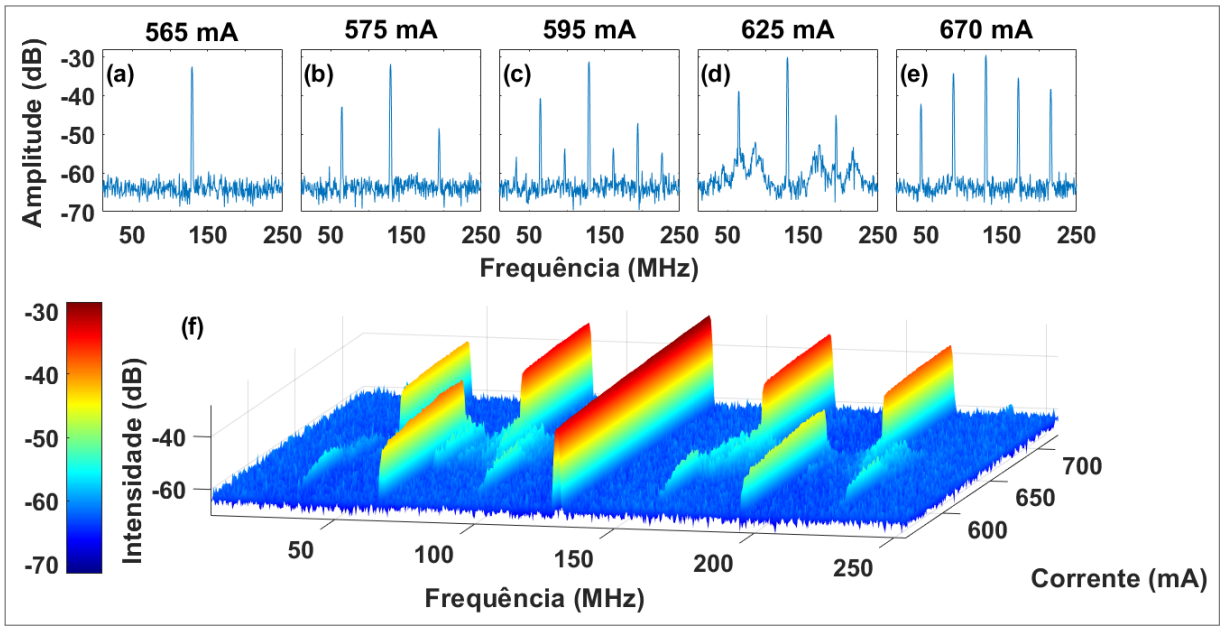
Uma maneira alternativa e prática de observar a rota para o caos neste experimento é utilizando o analisador de Rádio-Frequência (RF), que nos permite enxergar as frequências que compõem a série temporal da Figura 6. A Figura 7 traz esses dados em dois formatos

diferentes. Iniciando com a visualização mais rotineira do espectro de RF de um laser de modos travados, em (a), vemos um único pico representando a taxa de repetição do laser. Isso corresponde à série temporal mostrada na Figura 6 (a), onde temos um comportamento repetitivo bem definido pelos pulsos do laser, com uma frequência de repetição dos pulsos se destacando muito acima de todas as outras. Na figura (b) e (c), o espectro de RF reflete o surgimento do período-2 e período-4, respectivamente, com um pico extra em (130MHz/2) $\sim$65MHz, em (b), e mais um pico extra em (130MHz/4) $\sim$32.5MHz, em (c). Em (d) finalmente vemos o surgimento de um traço distinto dos demais: um contínuo de frequências aparece no espectro de RF. O comportamento aperiódico revelado na Figura 6 (d) surge como uma distribuição contínua de frequências de repetição no espectro de RF, identificando o caos determinístico do sistema de forma muito mais direta que a observação da série temporal. A figura (e) segue homóloga à Figura 6 (e), com um pico em (130MHz/3) $\sim$43.3MHz. Os picos adicionais das figuras (b) à (e) são os batimentos das frequências principais, comentadas aqui. Se colocarmos todos os espectros de RF obtidos durante nossa varredura, obtemos uma figura tridimensional que permite uma análise muito mais ampla do que foi apresentado anteriormente. Em (f) mostramos todas as 160 curvas de espectro de RF, num intervalo que vai de 560mA à 720mA, com incrementos de 1mA entre cada curva. Dessa forma vemos todo o desenvolvimento da rota para o caos no laser, como os pontos de surgimento de multi estabilidade e caos, semelhante ao mapa de bifurcação. De 560mA até 566mA, temos o laser funcionando de forma padrão, apenas com a frequência de repetição surgindo no espectro de RF 3D. De 567mA até 584mA, temos período-2. De 585mA até 610mA, surge mais um pico, configurando período-4. De 611mA até 652mA, temos a janela de caos do sistema. Então, de 653mA até 720mA retornamos à condição periódica, com período-3.

Podemos comparar a Figura 7 (f) com o mapa de bifurcação gerado pela mesma varredura. A Figura 8 revela as mesmas características que demonstramos no parágrafo anterior. Naturalmente, as transições de estabilidade podem ser vistas para os mesmos valores de corrente que na Figura 7 (f). As marcações de I à V, em vermelho, mostram as posições das quais as Figuras 6 e 7 de (a) até (e) foram retiradas. Para cada valor de corrente, um total de 100 picos (pontos) retirados da série temporal gerada pelo osciloscópio foram utilizados. Note como a região em torno da marcação IV preenche um intervalo de amplitudes, deixando evidente o comportamento caótico do sistema.

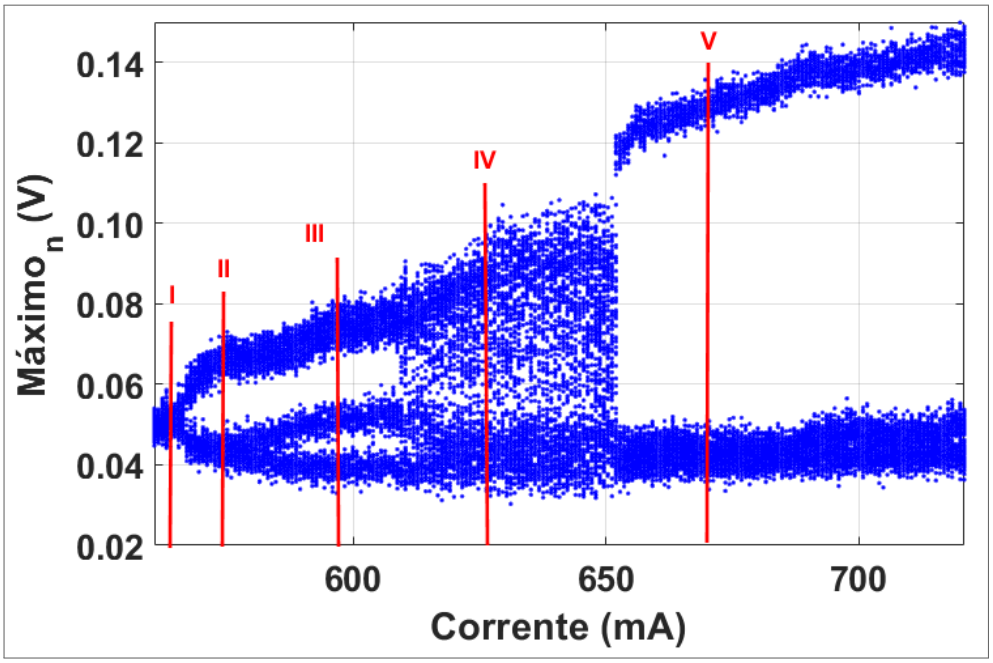
Devido ao ruído gaussiano, oriundo dos instrumentos de detecção, a visualização dos ramos no período-4 e período-3 se torna difícil no mapa de bifurcação da Figura 8. Para ajudar o

Figura 7 – Espectro de RF do laser sob diferentes condições de corrente. (a) mostra a taxa de repetição do laser (∼130MHz), correspondente à Figura 6 (a). (b), (c), (d) e (e) são respectivamente relacionados aos homólogos da Figura 6. Note como a Figura (e) apresenta um contínuo de frequências, ao invés de apenas picos. (f) nos permite ter uma visão completa da rota para o caos, com todas as nossas capturas de RF colocadas em sequência.



Fonte: Elaborado pelo autor (2022)

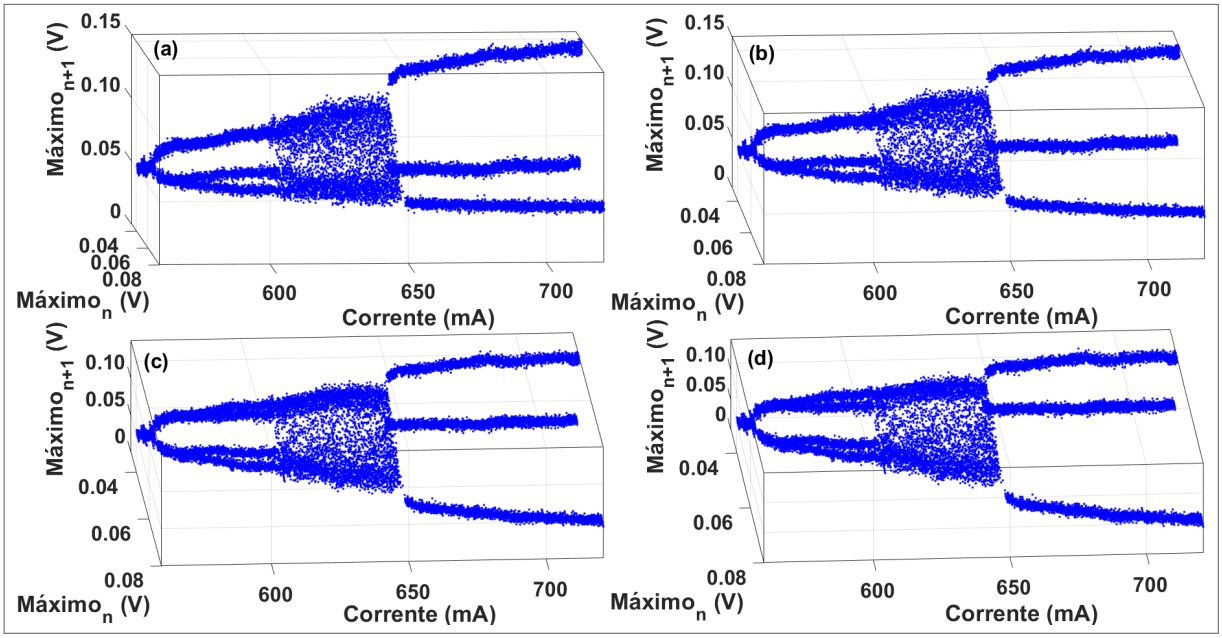
Figura 8 – Diagrama de bifurcação correspondente à varredura mostrada na Figura 7 (f). As marcações de I à V indicam as posições dos dados utilizados para fazer as Figuras 6 e 7 de (a) a (e). Embora não seja possível visualizar os ramos do período-4 e período-3 perfeitamente devido ao ruído gaussiano, eles se tornam claros quando visualizamos o mapa de bifurcação em três dimensões, mostrado na Figura 9.



Fonte: Elaborado pelo autor (2022)

leitor a enxergar todas as ramificações, montamos um mapa de bifurcação tridimensional, mostrado na Figura 9. Este mapa 3D é montado de forma simples, empilhando os mapas de retorno para cada valor de corrente, de forma que o mapa de bifurcação 2D não é nada mais que uma visão (ou projeção) lateral do mapa 3D. As figuras de (a) a (d) desenvolvem uma rotação lenta da figura tridimensional, partindo de uma posição equivalente ao visto na Figura 8 e gradualmente buscando uma visão superior, em (d), que permite a identificação de todas as ramificações no mapa de bifurcação.

Figura 9 – Diagrama de bifurcação tridimensional correspondente à varredura mostrada na Figura 7 (f). Esse mapa é formado empilhando os mapas de retorno para cada valor de corrente. A projeção lateral desse mapa forma a Figura 8. As figuras de (a) a (d) seguem uma rotação lenta, partindo da condição de visualização da Figura 8, até atingir uma perspectiva superior que permite a verificação de todos os ramos do mapa de bifurcação.

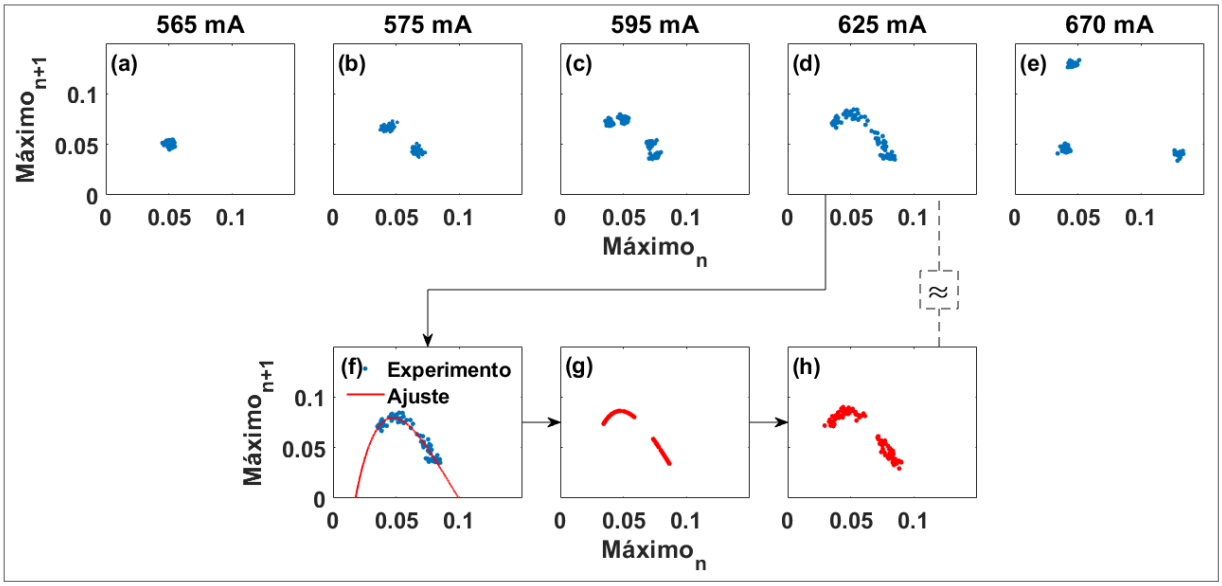


Fonte: Elaborado pelo autor (2022)

Indo em direção da caracterização quantitativa, trazemos os mapas de retorno na Figura 10. Cada um dos mapas de retorno da figura é formado por 100 pontos. Assim como temos feito, as figuras de (a) a (e) estão relacionadas aos recortes I à V da Figura 8 (a). Além de podermos confirmar o comportamento estável em (a), essa representação também nos permite mensurar o ruído gaussiano, oriundo dos instrumentos. Iremos utilizar essa informação em breve, a fim de verificarmos a qualidade da nossa função de ajuste. (b) e (c) mostram o comportamento do período-2 e período-4, nos permitindo ver o início da formação do caos unimodal. Em (d) temos a curva que estávamos esperando. Note que os pontos estão todos distribuídos em torno de uma curva bem definida. Mesmo com o ruído gaussiano, a característica unimodal ainda é bem distinta. Embora deparar-se com comportamento caótico num laser de tamanha

complexidade dinâmica seja esperado, encontrar caos unimodal é surpreende pela simplicidade. Ser capaz de definir os máximos de um sistema tão complexo, diante de um comportamento tão rico, com apenas uma função é uma enorme vantagem, pois aponta para a possibilidade de um modelo simplificado para a dinâmica do laser.

Figura 10 – Mapas de retorno experimentais e previstos pelo ajuste. Cada figura é formada por 100 pontos. De (a) a (f) temos os mapas de retorno correspondentes as regiões de I a V da Figura 8 (a), respectivamente. (d) se destaca pela característica caótica, com os pontos distribuídos por uma região ampla. (f) trás o ajuste feito com a função $Axe^{Bx} + C$. (g) é o mapa de retorno com os pontos gerados pela função de ajuste. (h) mostra o resultado de (g) somado a ruído gaussiano, para simular o ruído dos aparelhos experimentais, evidenciando a qualidade do ajuste através da semelhança entre (h) e (d).



Fonte: Elaborado pelo autor (2022)

A partir da Figura 10 (d), podemos tentar fazer um ajuste a fim de entender qual a relação matemática entre um pulso e o que o precede. Apesar da forma da curva sugerir um ajuste parabólico o resultado desse ajuste não é satisfatório, pois não é capaz de reproduzir os dados experimentais (e também leva a um Coeficiente de Determinação (R^2) menor, de 0.85). Entretanto, a curva $Axe^{Bx} + C$, com coeficientes $A = -15.2$, $B = -21$ e $C = -0.18$, foi capaz de imitar os dados experimentais, com $R^2 \approx 0.93$. Esse ajuste pode ser visto na figura (f), em linha vermelha, superpondo os pontos experimentais, em azul. Podemos utilizar a função do ajuste para gerar pontos no mapa de retorno, da mesma forma que fizemos na Figura 4. Esses pontos podem ser visto na Figura 10 (g). Note que algumas semelhanças podem ser notadas, como a separação entre as duas regiões contínuas ocupadas pelos pontos e os intervalos limites da curva. Mas quando adicionamos ruído gaussiano com média 0 e desvio padrão 0.002 aos pontos gerados pela função, assim como registrado nos dados experimentais,

a semelhança entre os gráficos deixa evidente que o ajuste é capaz de prever o comportamento do laser. Em (h) vemos que a previsão do comportamento, com o ruído gaussiano incluso, se aproxima dos dados experimentais em (d). A curva escolhida para o ajuste é especial, pois foi inspirada na referência (40), página 186, onde uma curva similar é mostrada como resultado de um mapa de retorno que pode ser retirado a partir das equações de Rössler, famosas por serem as equações dinâmicas mínimas para a existência de comportamento caótico. Então, temos nosso conjunto de dados sendo descrito pela mesma relação que teve como origem o sistema dinâmico mais simples possível capaz de apresentar rota para o caos. Embora essa conexão guarde uma promessa instigante sobre o modelo do laser, infelizmente não tivemos tempo para investigar a extensão das similaridades entre as equações de Rössler e o nosso sistema. Então deixamos essa ideia como uma sugestão para futuros trabalhos.

Prosseguindo com a análise, agora que temos uma curva capaz de descrever bem o comportamento do regime caótico, podemos utilizar a Equação 2.9 para confirmar de forma quantitativa o caos no nosso sistema. Utilizando nosso ajuste como suporte para calcular a derivada em cada um dos 100 pontos experimentais, como visto na Figura 10 (f), obtemos $\lambda_f \approx 0.14$, confirmando que o sistema realmente está em regime caótico. Podemos ir um pouco além e comparar esse resultado com a previsão do próprio ajuste. Se calcularmos o coeficiente de Lyapunov para os 100 pontos em (g), obtemos $\lambda_g \approx 0.20$. Esse valor é bastante próximo do que obtemos em (f), mas o resultado se aproxima ainda mais quando consideramos também o ruído gaussiano, como em (h). Para os dados em (h), obtemos $\lambda_h \approx 0.16$, ficando ainda mais parecido com o resultado experimental e confirmando as similaridades entre o experimento e o ajuste.

Com isso, temos uma boa caracterização do caos no nosso sistema. Embora não tenhamos abordado o período-3, o surgimento dessa dinâmica faz parte da sequência padrão na rota para o caos unimodal. É fato que o regime de período-3 não se encaixa tão bem no ajuste. Mas, dadas as dificuldades de reproduzir caos unimodal em um sistema experimental, cremos que a razão seja o surgimento de outros fatores que são menos relevantes para valores de corrente menores. Deixamos a sugestão de uma investigação mais profunda para futuros trabalhos. Ainda assim, mostrarmos que mesmo um laser tão complexo é capaz de apresentar caos unimodal. Os resultados apresentados nessa seção foram publicados e podem ser encontrados em (3). Entretanto, nós exploramos um pouco mais a caracterização da rota para o caos no laser. Como ficou evidente nas Figuras 6, 8 e 10, a energia dos pulsos sofre grandes variações entre cada volta quando em condições de multi estabilidade. Isso levanta o questionamento

sobre se, e como, isso tem algum impacto na potência média do laser. Vamos explorar esse cenário na sessão seguinte.

3.2 TRANSMISSÃO NÃO LINEAR INTERNA EM BIFURCAÇÕES NO LASER DE MODOS TRAVADOS EM FIBRA DE YB

Vimos como podemos identificar multi estabilidade no laser de fibra de Yb com o auxílio de osciloscópios, analisadores de RF e algum processamento de dados. Essas técnicas são capazes de dar uma descrição muito clara do comportamento caótico. Por outro lado, não exploramos uma das medidas mais corriqueiras ao se trabalhar com um laser: a medida da potência média, usando um detector lento. Mesmo para lasers de modos travados, a medida da potência média é utilizada para estimar a potência de pico do pulso ao invés de uma medida direta. No nosso laser, o regime de modos travados utiliza um PBS em combinação com a rotação não linear da polarização (9). Dessa forma, é evidente que a transmissão não linear tem um papel fundamental na formação dos pulsos de tamanhos diferentes, vistos na Figura 6. A questão menos óbvia é se essa dinâmica deixa traços na potência média e se é possível utilizar essa medida simples para caracterizar a rota para o caos. Iremos investigar essa questão agora.

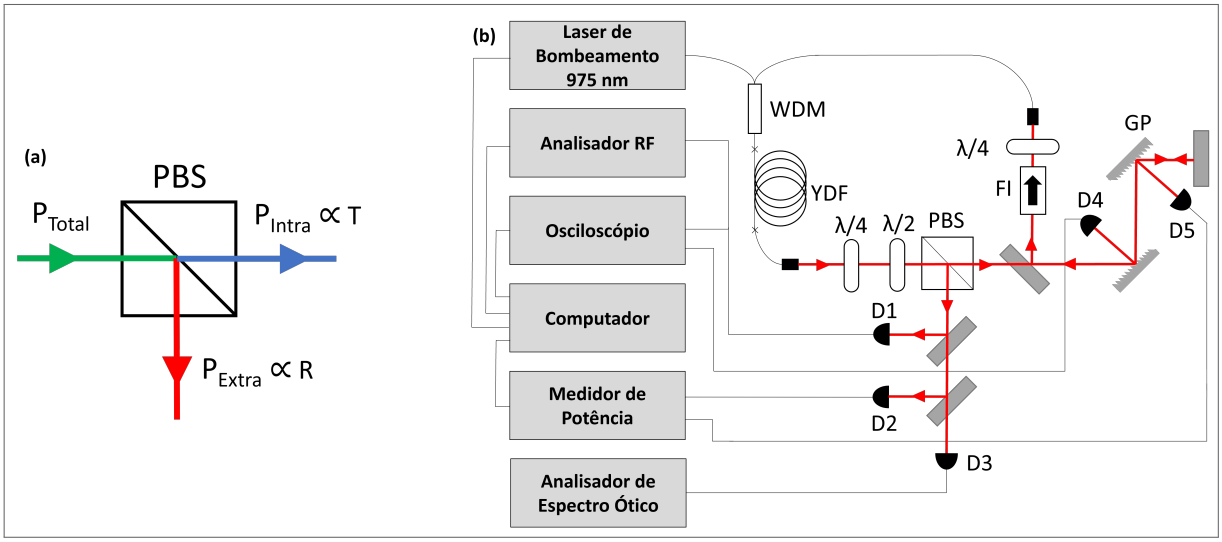
Para abordar a nossa proposta, será fundamental ter conhecimento do que acontece em torno do PBS. Iremos então observar a relação entre o feixe antes do PBS e o feixe de saída. Visto que estamos interessados principalmente na potência média, iremos nos referir a esses feixes por P_{Total} e P_{Extra}, como pode ser visto na Figura 11 (a). Entretanto, medir diretamente P_{Total} iria impedir o funcionamento do laser, então vamos somar P_{Extra} e P_{Intra} para obter o valor de P_{Total}. Porém, novamente, medir P_{Intra} diretamente também iria impedir o funcionamento do laser. Sendo assim, iremos utilizar os feixes de ordem zero das grades de difração para realizar as medidas, visto que esses feixes surgem naturalmente e não apresentavam utilidade pratica até então. A forma como os detetores foram montados pode ser vista na Figura 11 (b). A menos dos detetores, o sistema é o mesmo da Figura 5 (a), inclusive os detetores D1 e D2. D3 é um analisador de espectro ótico. Dentro da cavidade, foram incluídos D4 e D5. D4 é um detetor de 1GHz, assim como D1, que irá monitorar o trem de pulsos. D5 é um medidor de potência. Ambos D2 e D5 tem largura de banda de aproximadamente 50 Hz. Visto que iremos realizar as medidas de P_{Extra} e P_{Intra} nas posições de D2 e D5, lembrando que esse sistema tem uma quantidade considerável de perdas (9), é necessário realizar uma

calibração para relacionar o valor medido com o valor desejado. Para calibrar os detetores nós aproveitamos o fato de que o laser de Yb é capaz de funcionar em regime CW mesmo quando bloqueamos a transmissão após o PBS, devido ao grande ganho que a fibra dopada é capaz de oferecer. Com isso, podemos medir a potência do laser funcionando em regime CW logo após o meio de ganho e comparar diretamente com uma medida após o PBS, fazendo uma leitura da potência transmitida e a potência refletida. Conhecendo esses dois valores, basta relacionar o que foi medido após o PBS com a potência medida na posição de cada detector específico. Os fatores multiplicativos necessários para D2 e D5 são 1.3 e 17.9, respectivamente. Isso nos permite reconstruir o valor de potência total ao somar as medidas calibradas de D2 e D5. No PBS, podemos considerar $P_{Total} = P_{Extra} + P_{Intra}$, pois as perdas intrínsecas do PBS são insignificantes. Com isso, a transmissão, T, e a reflexão, R, podem ser definidas como

$$R = \frac{P_{Extra}}{P_{Total}} \tag{3.2}$$

$$T = \frac{P_{Intra}}{P_{Total}} \tag{3.3}$$

Figura 11 – (a) Diagrama dos feixes do laser em torno do PBS e as nomenclaturas que iremos adotar. (b) Atualização do experimento, com a inclusão dos novos instrumentos de medida em relação a Figura 5 (a). Além dos detetores D1 e D2, que estavam presentes anteriormente, foram adicionados o D3, D4 e D5. D3, fora da cavidade, é um analisador de espectro ótico. Dentro da cavidade temos D4, um detetor de 1 GHz para observação do trem de pulsos no osciloscópio, e D5, um medidor de potência.

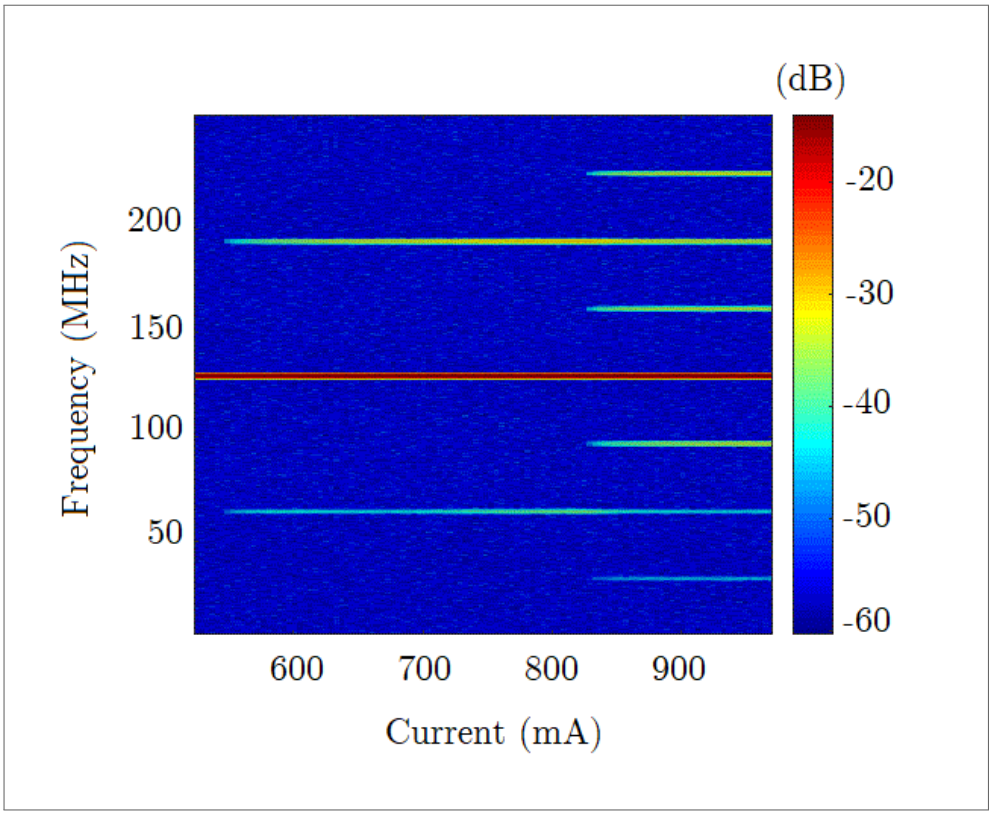


Fonte: Elaborado pelo autor (2022)

Iniciamos nosso estudo observado a resposta do sistema a alterações de corrente, como fizemos anteriormente. Os parâmetros fixos, posição das placas de onda e separação do par de

grades de difração para controle de dispersão, foram levemente alterados para esse experimento devido a necessidade de encaixar os detetores D4 e D5 no mesmo espaço onde ficam os elementos da cavidade do laser. Podemos visualizar rapidamente a dinâmica da varredura que iremos trabalhar utilizando o analisador de RF. Na Figura 12 vemos que nossa varredura vai de 525 mA à 969 mA. Temos período-1 de 525 mA à 545 mA, período-2 de 526 mA à 829 mA e período-4 830 mA à 969 mA. Essa captura foi feita com o feixe extracavidade, no detector D1.

Figura 12 – Evolução do feixe extracavidade ao longo da varredura que iremos estudar. Capturado em D1. A varredura apresenta período-1, período-2 e período-4.

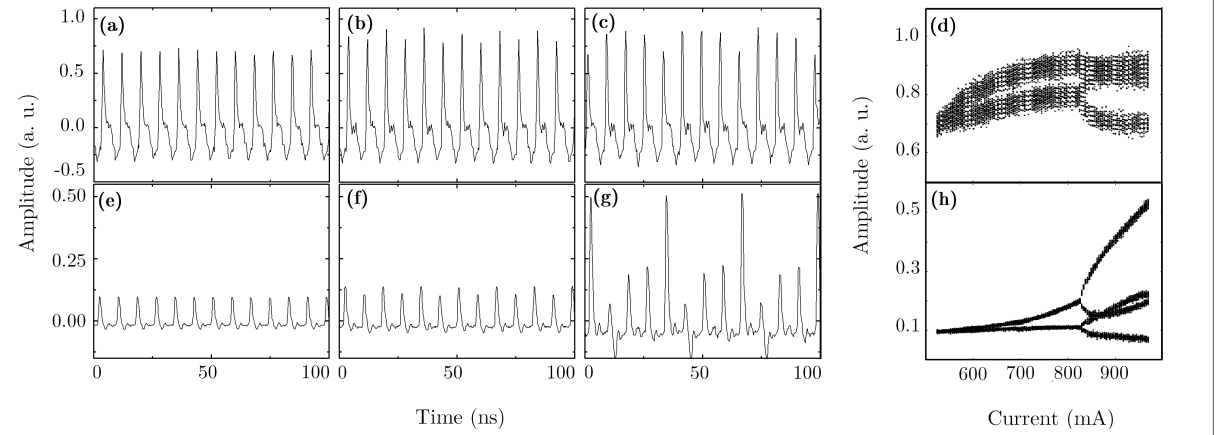


Fonte: Campos et al.(41) (2020a)

Mas agora também examinamos a resposta dentro da cavidade. Na Figura 13 podemos ver o trem de pulsos e o diagrama de bifurcação gerados pelo feixe transmitido e o refletido. (a), (b) e (c) mostram a aparência da série temporal com período-1 (529 mA), período-2 (721 mA) e período-4 (957 mA), respectivamente, do feixe refletido. (d) mostra o diagrama de bifurcação, dando uma visão do comportamento do sistema numa varredura completa. Como o sinal em D4 é consideravelmente mais fraco, o ruído se torna muito mais evidente. Ainda assim é possível enxergar os pontos de bifurcação, embora os ramos do período-4 fiquem superpostos devido ao ruído. Perceba que a rota para o caos no feixe intracavidade reflete o

que foi mostrado pelo feixe extracavidade na Figura 12. Também é interessante notar, em (d), como o sinal satura após $\sim$ 800mA. Isso irá contrastar com o feixe refletido.

Figura 13 – (a)-(c), trem de pulsos apresentando período-1 a período 4 dentro da cavidade, T. (d) é o diagrama de bifurcação gerado pela varredura de corrente. (e)-(f), trem de pulsos do complementar, R, apresentando a mesma sequência de bifurcação. (h) é o diagrama de bifurcação do feixe refletido. Algumas diferenças são perceptíveis entre o feixe refletido e o transmitido, apesar das bifurcações ocorrerem para o mesmo valor de corrente em ambos.



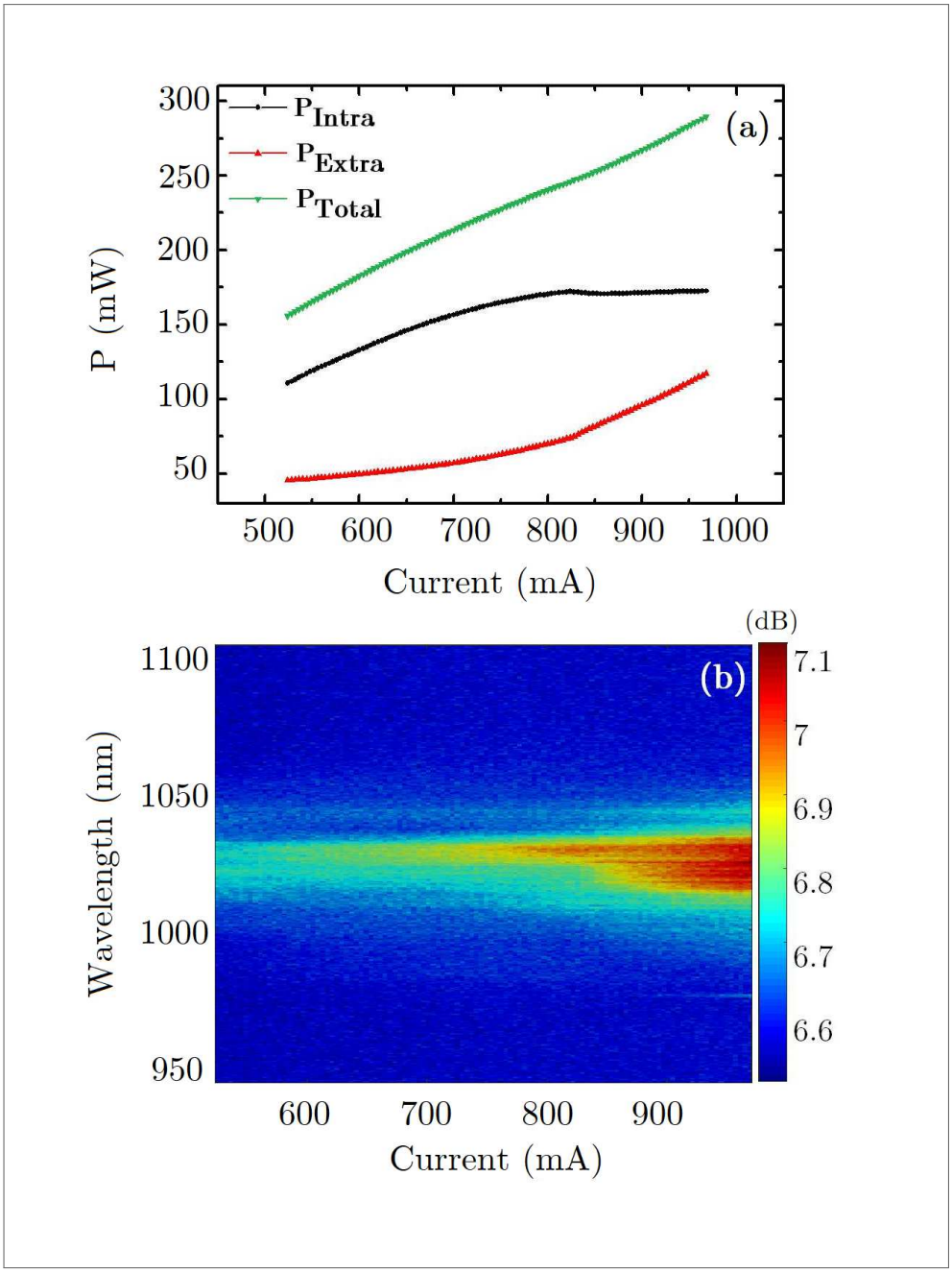
Fonte: Campos et al.(41) (2020a)

As Figuras 13 (e), (f) e (g) mostram as séries temporais do feixe refletido, complementares a (a), (b) e (c), respectivamente. Enquanto as séries do feixe transmitido tem uma variação de amplitude pequena, o feixe refletido mostra grandes diferenças entre os pulsos. O diagrama de bifurcação do feixe refletido, (h), mostra uma situação reversa ao que foi visto em (d). Enquanto (d) satura quando após 800 mA, (h) apresenta um crescimento mais rápido. Isso está relacionado com o rotação não-linear de polarização. Pulsos com maior valor de pico experimentam uma rotação mais intensa e, consequentemente, serão mais refletidos pelo elemento polarizador, o PBS. Isso automaticamente explica a saturação vista em (d). A partir de 800 mA a amplitude dos pulso aumenta de forma que qualquer energia extra é jogada para a direção ortogonal pelo efeito de rotação não linear da polarização, fazendo com que exista um ponto de saturação para a amplitude do pulso intracavidade.

A saturação da amplitude dos pulsos se reflete em P_{Intra} e P_{Extra}, como podemos ver na Figura 14 (a). O ponto de saturação, visto anteriormente na Figura 13 (d), pode ser claramente identificado na curva da P_{Intra}. Porém, aqui não estamos observando as curvas geradas pelos pulsos individuais, mas sim o efeito médio de aproximadamente dois milhões de pulsos. Isso é diferente, pois implica que apesar da rotação não linear da polarização alterar os pulsos

de forma individual, o comportamento do laser é alterado como um todo. Ilustrando com um contraste, deveria ser perfeitamente plausível que os pulsos de maior amplitude fossem modificados de forma que os menores compensassem o efeito médio, mantendo P_{Intra} sem o efeito de saturação na potência média. Mas a Figura 14 (a) deixa claro que a saturação não é apenas um efeito local, mas sim global, para o laser.

Figura 14 – (a) Potência intracavidade, P_{Intra}, potência extracavidade, P_{Extra} e a soma das duas, P_{Total}. Demonstrando o comportamento não linear e saturação, vistos na Figura 13, refletidos na potência média. (b) Espectro ótico ao longo da varredura, mostrando uma transição clara a partir do ponto de saturação.



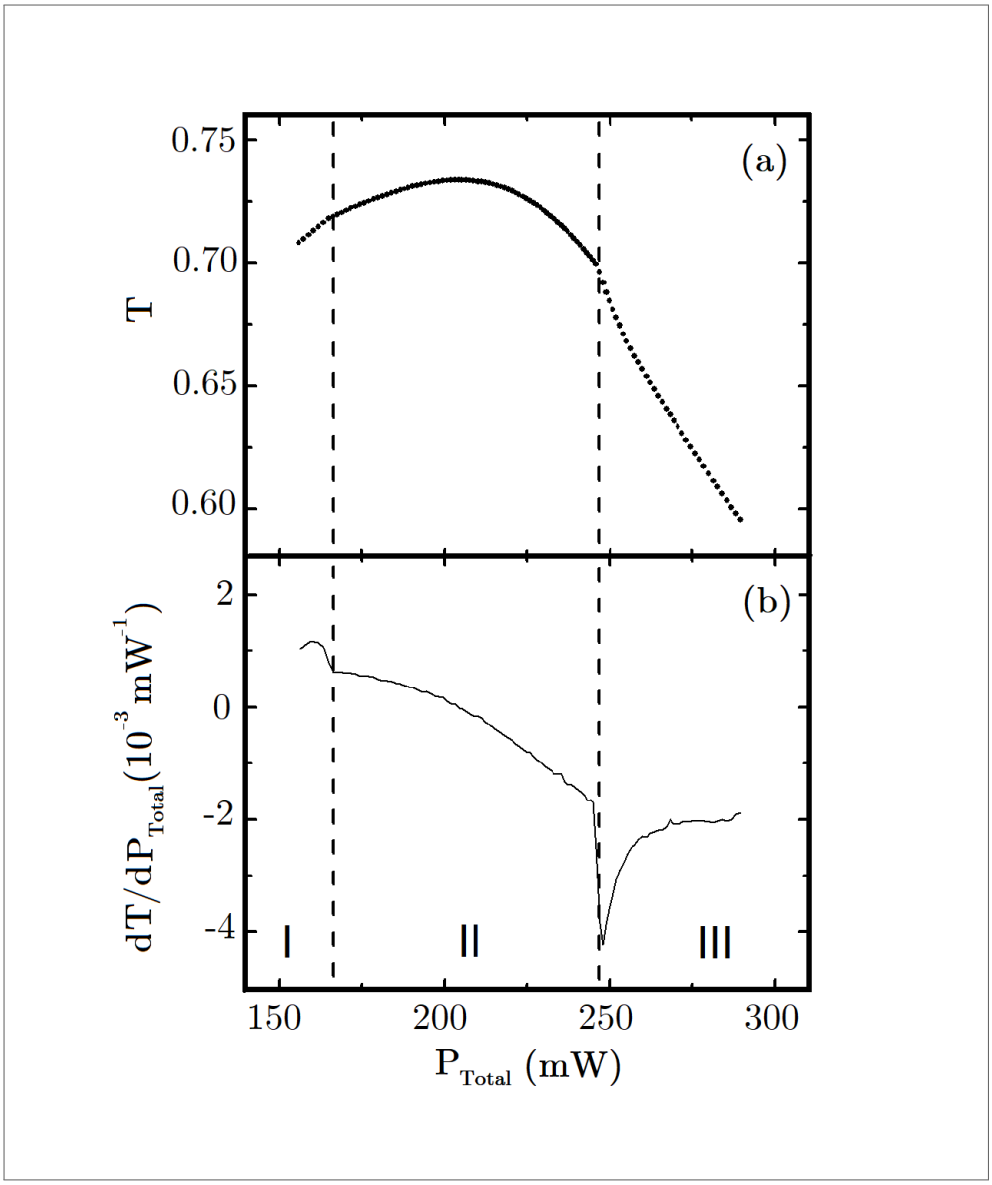
Fonte: Campos et al.(41) (2020a)

Quando adicionamos P_{Intra} à sua contraparte, P_{Extra}, vemos que a evolução de P_{Total} é linear com a corrente, como costuma ocorrer com um laser CW após o limiar, seguindo o ganho de pequeno sinal. Isso implica que o comportamento de P_{Extra}, com um aumento na taxa de potência versus corrente, é apenas um comportamento complementar ao de P_{Intra}. Ou seja, estamos vendo como a rotação não linear da polarização distribui a potência média, não apenas a energia dos pulsos individuais, à medida que variamos a corrente do laser de bombeio. Esse comportamento de saturação é acompanhado de uma mudança no espectro ótico de P_{Extra}, como pode ser visto na Figura 14 (b). O espectro mais largo e com maior amplitude explica a energia por pulso particularmente mais alta nesta região, visto na Figura 13 (g). Sabemos que a energia do pulso pode ser calculada integrando o seu espectro ótico. Visto que na Figura 14 (b) o espectro tem muito mais energia após 800 mA, sabemos imediatamente que teremos mais energia por pulso. Além disso, uma largura de banda maior implica em uma menor largura temporal dos pulsos que, por sua vez, tem como consequência valores de pico mais altos (se Limitado por Transformada de Fourier (LTF)). Embora esses traços de saturação e alargamento de espectro estejam claramente associados com o período-4 e que também seja possível identificar o ponto de bifurcação do período-4 a partir da potência média, a curva não parece revelar nada sobre a bifurcação para período-2, da forma que está apresentada. Vale salientar que não era possível capturar simultaneamente o espectro ótico de P_{Intra}, pois só possuímos um único espectrômetro. Além disso, não havia espaço para incluir um espectrômetro que pudesse capturar a informação dentro da cavidade. Visto que essa medida era de menor relevância para esse estudo e diante das dificuldades, optamos por não realizá-la.

Partimos então para uma forma diferente de observar a potência média. Na Figura 15 vemos a curva de transmissão, T, e sua derivada. Em (a) percebemos que a curva de transmissão, sim, é capaz de identificar as transições do regime caótico. As linhas tracejadas verticais marcam as transições de período-2 e período-4. Uma mudança brusca na curva é vista em torno das marcas. Essas mudanças contrastam com a variação lenta, apresentada ao longo das secções da curva. Ou seja, a potência média do laser detém informação da dinâmica na taxa de repetição, por mais que essa informação não fosse acessível nas medidas de potência individuais. A curva da transmissão, que é não linear devido a rotação não linear de polarização, serve como uma caracterização mais completa do laser, em comparação com a potência média e o espectro ótico, e suficiente para identificar quando o sistema apresenta transições dinâmicas de rota para o caos. Embora não seja possível identificar qual transição ocorreu, essa medida ainda representa uma grande vantagem para o dia-a-dia no laboratório, visto que uma varre-

dura em corrente e o mapeamento da curva de transmissão são procedimentos extremamente simples. Uma vez que a maioria das aplicações busca um laser em regime estável, essa medida permite identificar se houve uma transição dinâmica durante uma varredura na potência do bombeamento. Uma forma alternativa e mais clara de enxergar as transições é a partir da derivada da curva de transmissão não linear, vista em (b), a qual permite ver de forma ainda mais evidente as mudanças bruscas ocorridas na curva de transmissão. Com isso, fica claro que é a rota para o caos deixa traços na potência média do laser e que é possível realizar investigações rápidas de transições dinâmicas nesse sistema.

Figura 15 – (a) curva de transmissão não linear, T. (b) Derivada da curva de transmissão não linear. Note como ambas contém mudanças bruscas em torno das linhas tracejadas, que indicam as bifurcações de período-2 e período-4, deixando claro que é possível identificar transições dinâmicas a partir de medidas de potência média.



Fonte: Campos et al.(41) (2020a)

Com isso, encerramos nossas investigações experimentais da dinâmica caótica no laser de modos travados em fibra de Yb. Publicamos os resultados apresentados nesta sessão em (41) e, em seguida, continuamos a investigar a dinâmica não linear deste laser, porém não mais o regime caótico, o que rendeu uma outra publicação. Neste terceiro trabalho, investigamos transições discretas na separação temporal e fase relativa dos sólitons ligados deste mesmo sistema. Embora esse assunto fuja do escopo desta tese, o leitor interessado pode encontrá-lo em (42).

4 PULSOS LIMITADOS POR TRANSFORMADA DE FOURIER E ABSORÇÃO MULTI-FOTÔNICA

Passaremos agora a discutir o problema de absorção-multi fotônica e como realizar uma otimização espectral que favoreça a transição para o estado excitado. A primeira proposta de ajuste espectral com o intuito de aumentar a Absorção de Dois Fótons (ADF) de um pulso foi dada por Dudovich et al. (34). O foco dessa primeira proposta era demonstrar que pulsos LTF são incapazes de maximizar uma ADF devido a uma interferência destrutiva causada pela fase espectral desse tipo de pulso. Então, analisando a equação da absorção, o autor experimenta duas modificações espectrais que resultam em um aumento na absorção do pulso pelo átomo. A melhor delas, do nosso interesse, se utilizou de uma MEL para aplicar uma Máscara de Fase (MF) de $\pi/2$ entre as frequências ressonantes e foi capaz de aumentar a fluorescência causada pela absorção do pulso em sete vezes.

Nosso objetivo é aplicar o mesmo princípio à Absorção Sequencial de Dois Fótons (ASDF). Porém, a complexidade do modelo da ASDF que iremos analisar é tal que propor analiticamente uma MF capaz de otimizar o processo não é trivial. Por isso lançaremos mão do AG como ferramenta para encontrar a MF ótima para o nosso problema. Além disso, iremos observar as soluções encontradas pelo AG aumentando a complexidade do modelo gradualmente, partindo de uma ADF e adicionando elementos até chegar à ASDF. Dessa forma seremos capazes de compreender como cada mudança afeta a solução proposta pelo AG, observando sua origem analítica e revelando novos comportamentos ao longo das análises. Sendo assim, vamos iniciar nosso estudo observando o modelo de ADF para entender porque pulsos LTF não são capazes de maximizar uma ADF.

4.1 MODELO PARA ABSORÇÃO DE DOIS FÓTONS

Como foi dito anteriormente, nosso intuito é maximizar a população do estado excitado de um átomo, a qual pode ser descrita como $\rho_{ee}(t) = c_e(t)c_e^*(t)$, onde $c_e(t) = \langle e|\hat{U}_I(t)|f\rangle$, $\hat{U}_I(t)$ é o operador de Dyson (operador de evolução temporal na representação de interação), $|f\rangle$ é o estado fundamental e $|e\rangle$ é o estado excitado. Considere agora um potencial, $\hat{V}(t)$, gerado pelo campo elétrico de um pulso LTF de sinal, $E_s(t)$, e um segundo campo elétrico de um pulso, também LTF, de controle, $E_c(t)$, tal que possamos escrever

$$\hat{V}(t;\tau) = \sum_n -\mu_{fn} E_s(t)|f\rangle\langle n| - \mu_{ne} E_c(t-\tau)|n\rangle\langle e| + h.c.. \tag{4.1}$$

μ_{ij} é o dipolo elétrico da transição $i \to j$, τ é o atraso temporal entre os campos, $|n\rangle$ é o n-ésimo estado intermediário e $h.c.$ é a sigla para indicar o hermitiano conjugado da expressão. Passando para a representação de interação, $\hat{V}_I(t) = e^{i\hat{H}_0 t/\hbar}\hat{V}(t)e^{-i\hat{H}_0 t/\hbar}$, temos:

$$\hat{V}_I(t;\tau) = \sum_n \left[-\mu_{fn} e^{-i\omega_{fn}t} E_s(t)|f\rangle\langle n| - \mu_{ne} e^{-i\omega_{ne}t} E_c(t-\tau)|n\rangle\langle e| + h.c.\right], \tag{4.2}$$

onde $\omega_{ij} = \omega_j - \omega_i$, $\omega = E/\hbar$. Expandindo o operador de Dyson na série de Dyson, podemos escrever c_e da seguinte forma:

$$c_e(t) = \langle e|\hat{U}_I(t)|f\rangle, \tag{4.3}$$

$$c_e(t) = \langle e|\left[1 - \frac{i}{\hbar}\int_{-\infty}^{t} dt'\hat{V}_I(t')\hat{U}_I(t')\right]|f\rangle, \tag{4.4}$$

$$c_e(t) = \langle e|\left[1 - \frac{i}{\hbar}\int_{-\infty}^{t} dt'\hat{V}_I(t') + \left(-\frac{i}{\hbar}\right)^2 \int_{\infty}^{t}\int_{\infty}^{t'} dt'dt''\hat{V}_I(t')\hat{V}_I(t'') + \dots\right]|f\rangle. \tag{4.5}$$

Observando a forma do potencial na Equação 4.2, vemos que

$$\langle e|\left[1 - \frac{i}{\hbar}\int_{-\infty}^{t} dt'\hat{V}_I(t')\right]|f\rangle = 0. \tag{4.6}$$

Em contrapartida

$$\langle e|\left[\left(-\frac{i}{\hbar}\right)^2 \int_{\infty}^{t}\int_{\infty}^{t'} dt'dt''\hat{V}_I(t')\hat{V}_I(t'')\right]|f\rangle = \\ -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2}\int_{-\infty}^{t}\int_{-\infty}^{t'} dt'dt'' e^{i\omega_{ne}t'} e^{i\omega_{fn}t''} E_c^*(t'-\tau)E_s^*(t''), \tag{4.7}$$

dando a primeira contribuição não nula da expansão. Podemos ainda escrever os campos da Equação 4.7 no formato $E(t) = \mathcal{E}(t)e^{i\omega t} + c.c.$, resultando em

$$c_e(t;\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2}\int_{-\infty}^{t}\int_{-\infty}^{t'} dt'dt'' e^{i\omega_{ne}t'} e^{i\omega_{fn}t''} \\ \times \left[\mathcal{E}_c^*(t'-\tau)e^{-i\omega_c(t'-\tau)} + c.c.\right]\left[\mathcal{E}_s^*(t'')e^{-i\omega_c t''} + c.c.\right]. \tag{4.8}$$

Aplicando a aproximação da onda girante, podemos simplificar a expressão para

$$c_e(t;\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{t}\int_{-\infty}^{t'} dt'dt'' e^{i(\omega_{ne}-\omega_c)t'} e^{i(\omega_{fn}-\omega_s)t''} \mathcal{E}_c^*(t'-\tau)\mathcal{E}_s^*(t''). \quad (4.9)$$

É possível escrever a integral em termos da amplitude não deslocada:

$$c_e(t;\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{t-\tau}\int_{-\infty}^{t'+\tau} dt'dt'' e^{i(\omega_{ne}-\omega_c)t'} e^{i(\omega_{fn}-\omega_s)t''} \mathcal{E}_c^*(t')\mathcal{E}_s^*(t''). \quad (4.10)$$

Nós estamos interessados nas soluções estáveis ($t \to \infty$), ou seja

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{t'+\tau} dt'dt'' e^{i(\omega_{ne}-\omega_c)t'} e^{i(\omega_{fn}-\omega_s)t''} \mathcal{E}_c^*(t')\mathcal{E}_s^*(t''), \quad (4.11)$$

que é a expressão base para a ADF. Entretanto, o problema de interferência destrutiva torna-se mais evidente quando observamos a expressão no espaço das frequências.

Usando a definição da transformada de Fourier para escrever a amplitude dos campos da Equação 4.11 no espaço das frequências, temos

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{t'+\tau} dt'dt'' e^{i(\omega_{ne}-\omega_c)t'} e^{i(\omega_{fn}-\omega_s)t''} \times \left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} d\omega' \tilde{\mathcal{E}}_c^*(\omega') e^{i\omega' t'}\right]\left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} d\omega'' \tilde{\mathcal{E}}_s^*(\omega'') e^{i\omega'' t''}\right], \quad (4.12)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{t'+\tau} dt'dt'' \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'') \times \frac{1}{2\pi} e^{i[(\omega_{ne}-\omega_c)+\omega']t'} e^{i[(\omega_{fn}-\omega_s)+\omega'']t''}, \quad (4.13)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'') \times \frac{1}{2\pi}\int_{-\infty}^{\infty} dt' e^{i[(\omega_{ne}-\omega_c)+\omega']t'} \int_{-\infty}^{t'+\tau} dt'' e^{i[(\omega_{fn}-\omega_s)+\omega'']t''}, \quad (4.14)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'') \times \frac{1}{2\pi}\int_{-\infty}^{\infty} dt' e^{i[(\omega_{ne}-\omega_c)+\omega']t'} \left[\frac{e^{i[(\omega_{fn}-\omega_s)+\omega'']t''}}{i[(\omega_{fn}-\omega_s)+\omega'']}\right]_{-\infty}^{t'+\tau}. \quad (4.15)$$

Podemos desprezar o termo com $t'' \to \infty$ pela aproximação da onda girante, restando

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'')}{(\omega_{fn}-\omega_s)+\omega''} \times \frac{1}{2\pi}\int_{-\infty}^{\infty} dt' e^{i[(\omega_{ne}-\omega_c)+\omega']t'} e^{i[(\omega_{fn}-\omega_s)+\omega'](t'+\tau)}, \quad (4.16)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'')}{(\omega_{fn}-\omega_s)+\omega''} e^{i[(\omega_{fn}-\omega_s)+\omega'']\tau} \times \frac{1}{2\pi}\int_{-\infty}^{\infty} dt' e^{i[(\omega_{ne}-\omega_c)+\omega']t'} e^{i[(\omega_{fn}-\omega_s)+\omega'']t'}, \quad (4.17)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'')}{(\omega_{fn}-\omega_s)+\omega''} e^{i[(\omega_{fn}-\omega_s)+\omega'']\tau} \times \frac{1}{2\pi}\int_{-\infty}^{\infty} dt' e^{i[(\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega'+\omega'']t'}. \quad (4.18)$$

A integral no tempo é uma Delta de Dirac na forma

$$\delta(x-a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i(x-a)k} dk,$$

sendo assim

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\omega' d\omega'' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(\omega'')}{(\omega_{fn}-\omega_s)+\omega''} e^{i[(\omega_{fn}-\omega_s)+\omega'']\tau} \times \delta((\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega'+\omega''). \quad (4.19)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-[(\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega'])}{(\omega_{fn}-\omega_s)-[(\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega']} \times e^{i\{(\omega_{fn}-\omega_s)-[(\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega']\}\tau}, \quad (4.20)$$

$$c_e(\tau) \approx -\sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i\omega_c\tau} \int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-[(\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega'])}{-[(\omega_{ne}-\omega_c)+\omega']} \times e^{-i[(\omega_{ne}-\omega_c)+\omega']\tau}, \quad (4.21)$$

$$c_e(\tau) \approx \frac{e^{i[\omega_c-(\omega_{ne}-\omega_c)]\tau}}{i\hbar^2} \times \sum_n \mu_{fn}\mu_{ne} \int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-[(\omega_{ne}-\omega_c)+(\omega_{fn}-\omega_s)+\omega'])}{(\omega_{ne}-\omega_c)+\omega'} e^{-i\omega'\tau}, \quad (4.22)$$

onde já podemos perceber que o denominador impõe uma mudança de fase de π em torno do polo da integral. Entretanto, vamos apenas trabalhar especificamente com o caso de ressonância de dois fótons ($\omega_{fn}+\omega_{ne}=\omega_s+\omega_c$) e pulsos gaussianos LTF, de forma que analisar a Equação 4.22 sem essas considerações traria dificuldades desnecessárias à discussão. Portanto, vamos agora partir para uma ADF ressonante antes de continuar nossa análise.

4.2 ABSORÇÃO DE DOIS FÓTONS RESSONANTE

Com a condição de ressonância para dois fótons, $\omega_{fn}+\omega_{ne}=\omega_s+\omega_c$, podemos escrever

$$c_e(\tau) \approx \frac{e^{i[\omega_c-(\omega_{ne}-\omega_c)]\tau}}{i\hbar^2} \sum_n \mu_{fn}\mu_{ne} \int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{(\omega_{ne}-\omega_c)+\omega'} e^{-i\omega'\tau}. \quad (4.23)$$

Usando a condição de ressonância, também podemos escrever em termos da frequência do campo de sinal:

$$c_e(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fn}-\omega_s)]\tau}}{i\hbar^2} \sum_n \mu_{fn}\mu_{ne} \int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{(\omega_{fn}-\omega_s)-\omega'} e^{-i\omega'\tau}. \quad (4.24)$$

A Equação 4.24 já é capaz de nos mostrar alguns comportamentos da ADF. Mas podemos avaliar com um maior nível de detalhes observando a relação com os fótons individuais, separando em dois tipos de transição: a Não Ressonante com os Fótons Individuais (NRFI) e a Ressonante com os Fótons Individuais (RFI).

4.2.1 Transições Não Ressonantes com os Fótons Individuais

Se $|\omega_{ij}-\omega_k| > \Delta\omega_k/2$, $\Delta\omega_k$ sendo a largura de linha do campo k, a região espectral na qual a mudança de fase ocorre não afetará a integração pois o campo elétrico só será não nulo antes ou depois da mudança de fase. É possível verificar a baixa relevância dos polos em transições NRFI fazendo

$$c_e^{NRFI}(\tau) \approx \frac{e^{i[\omega_c+(\omega_{fn}-\omega_s)]\tau}}{i\hbar^2}\sum_n \mu_{fn}\mu_{ne}\int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{\left[\frac{-(\omega_{fn}-\omega_s)}{-(\omega_{fn}-\omega_s)}\right]\left[(\omega_{fn}-\omega_s)-\omega'\right]} e^{-i\omega'\tau}, \quad (4.25)$$

$$c_e^{NRFI}(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fn}-\omega_s)]\tau}}{i\hbar^2}\sum_n \frac{\mu_{fn}\mu_{ne}}{\omega_{fn}-\omega_s}\int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{1-\frac{\omega'}{\omega_{fn}-\omega_s}} e^{-i\omega'\tau}. \quad (4.26)$$

Lembrando que estamos considerando $|\omega_{ij}-\omega_k| > \Delta\omega_k$, podemos assumir que $\tilde{\mathcal{E}}_k^*(|\omega-\omega_k| \geq |\omega_{ij}-\omega_k|) \approx 0$, ou seja $\omega/(\omega_{ij}-\omega_k) < 1$, o que nos permite utilizar a expansão da série geométrica no denominador da integral, dando

$$c_e^{NRFI}(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fn}-\omega_s)]\tau}}{i\hbar^2}\sum_n \frac{\mu_{fn}\mu_{ne}}{\omega_{fn}-\omega_s}\int_{-\infty}^{\infty} d\omega' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')e^{-i\omega'\tau} \times\left[1+\frac{\omega'}{\omega_{fn}-\omega_s}+\left(\frac{\omega'}{\omega_{fn}-\omega_s}\right)^2+\ldots\right]. \quad (4.27)$$

Observando os dois primeiros termos da expansão, temos

$$c_e^{NRFI}(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fn}-\omega_s)]\tau}}{i\hbar^2}\sum_n \frac{\mu_{fn}\mu_{ne}}{\omega_{fn}-\omega_s}\int_{-\infty}^{\infty} d\omega' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')e^{-i\omega'\tau} \times\left(1+\frac{\omega'}{\omega_{fn}-\omega_s}\right). \quad (4.28)$$

Porém, devido à paridade,

$$\int_{-\infty}^{\infty} d\omega' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')e^{-i\omega'\tau}\frac{\omega'}{\omega_{fn}-\omega_s} = 0, \quad (4.29)$$

restando

$$c_e^{NRFI}(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fn}-\omega_s)]\tau}}{i\hbar^2}\sum_n \frac{\mu_{fn}\mu_{ne}}{\omega_{fn}-\omega_s}\int_{-\infty}^{\infty} d\omega' \tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')e^{-i\omega'\tau}. \quad (4.30)$$

A Equação 4.30 deixa claro que os níveis NRFI contribuem com a ADF sem problemas de interferência destrutiva pois a mudança de fase induzida pelo polo da equação 4.24 não tem relevância para a integração. Em contrapartida, a irrelevância do polo também significa que esse tipo de absorção tem um potencial de contribuição muito inferior ao de qualquer transição RFI pois não terá acesso ao grande fator multiplicativo provido pela singularidade. Devido a essa baixa contribuição, não iremos considerar as transições NRFI ao longo do trabalho, focando apenas nas RFI.

4.2.2 Transições com Fótons Individuais Ressonantes

Se consideramos qualquer transição com um nível $|r\rangle$ RFI, podemos partir da Equação 4.24 e escrever

$$c_e^{RFI}(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fr}-\omega_s)]\tau}}{i\hbar^2}\mu_{fr}\mu_{re}\Bigg[i\pi\tilde{\mathcal{E}}_c^*(\omega_{fr}-\omega_s)\tilde{\mathcal{E}}_s^*(-(\omega_{fr}-\omega_s))e^{i(\omega_{fr}-\omega_s)\tau} + \text{p.v.}\int_{-\infty}^{\infty}d\omega'\frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{(\omega_{fr}-\omega_s)-\omega'}e^{-i\omega'\tau}\Bigg], \quad (4.31)$$

$$c_e^{RFI}(\tau) \approx -\frac{e^{i[\omega_c+(\omega_{fr}-\omega_s)]\tau}}{i\hbar^2}\mu_{fr}\mu_{re}\Bigg[i\pi\tilde{\mathcal{E}}_c^*(\omega_{fr}-\omega_s)\tilde{\mathcal{E}}_s^*(\omega_{re}-\omega_c)e^{i(\omega_{fr}-\omega_s)\tau} + \text{p.v.}\int_{-\infty}^{\infty}d\omega'\frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{(\omega_{fr}-\omega_s)-\omega'}e^{-i\omega'\tau}\Bigg], \quad (4.32)$$

onde p.v. indica que a integral deve ser avaliada pelo método do valor principal de Cauchy.

A integral da Equação 4.32 deixa claro que uma transição de dois fótons ressonante que também é RFI detém um grande potencial de absorção, muito maior do que o caso NRFI, devido a integral que contém uma singularidade. Porém, quando consideramos pulsos gaussianos LTF, cuja fase espectral é constante ao longo do espectro, perde-se esse potencial de amplificação devido a mudança de sinal (fase de π) imposta pelo denominador da integral. Com um pulso LTF temos uma interferência destrutiva, induzida pela própria transição, que mina o potencial de absorção disponível na integral. Esse será o principal problema que atacaremos ao longo dos capítulos seguintes. Um segundo aspecto revelado pela Equação 4.32 é que a posição do polo em relação ao pico do espectro ótico vai determinar a quantidade que está sendo amplificada pela singularidade. Ou seja, se a singularidade (ressonância) se dá exatamente no pico do espectro, teremos um maior fator de amplificação do processo do que se escolhermos a ressonância em algum ponto vizinho. Essas características afetam diretamente o resultado da absorção e, consequentemente, também impactam a maneira de abordar o problema de otimização de fase espectral para a ADF. Iremos observar cada um desses casos nos próximos capítulos.

5 ALGORITMO GENÉTICO

Vamos agora discutir uma outra parte do problema de otimização, que consiste em otimizar uma ADF fazendo ajustes na fase espectral do pulso que irá ceder o fóton para a transição que parte do nível fundamental. A mudança de fase que nós desejamos aplicar ao pulso será executada por um Modulador Espacial de Luz (MEL), mas a dificuldade consiste em conhecer exatamente que mudança de fase espectral deve ser aplicada a fim de otimizar a ADF. Se escrevemos o pulso como um campo elétrico em função da frequência angular, separando amplitude e fase espectral, temos $E(\omega) = \mathcal{E}(\omega)e^{i\phi(\omega)} + c.c.$; $\mathcal{E}(\omega)$ a amplitude e $\phi(\omega)$ a fase espectral; podemos perceber que existe um continuum de fases espectrais que podem ser modificadas. Isso implica que nosso problema tem, em essência, um número infinito de parâmetros, o que coloca em perspectiva a dificuldade que precisamos superar.

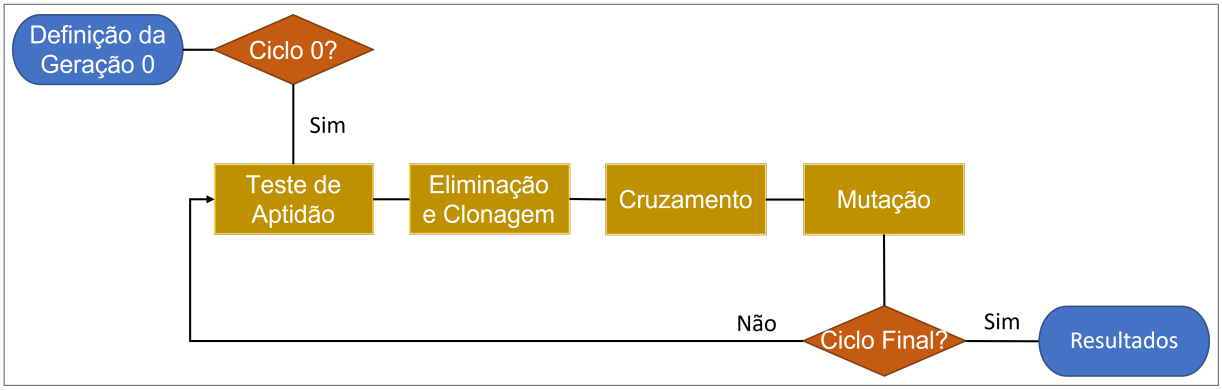
É fato que um MEL tem apenas um número limitado de entradas. No nosso caso, 128 entradas, visto que buscamos um aparato experimental similar ao encontrado em (43, 44). Porém, a resolução limitada não torna nosso problema menos complicado. Além da eficiência diminuída pela resolução, continuamos com o problema de definirmos o valor de 128 parâmetros. Se consideramos um MEL com apenas 8-bits (256 possíveis valores) em cada entrada, teremos 256^{128} possíveis respostas para o nosso problema de otimização. Se fossemos testar cada uma dessas possibilidades, precisaríamos executar $\sim 2 \times 10^{308}$ experimentos. Tal varredura seria impossível mesmo em termos computacionais. Por exemplo, suponha que temos à disposição um processador que fosse capaz de analisar um trilhão de hipóteses por segundo (pelo menos mil vezes mais rápido que os processadores atuais), ainda assim seriam necessários $\sim 2 \times 10^{296}$ segundos ou $\sim 5 \times 10^{285}$ milênios para concluirmos a tarefa. Sendo assim, foi necessário buscarmos a solução sem recorrer a uma varredura completa do espaço de soluções. Utilizando um algoritmo adaptativo, especificamente AG, fomos capazes obter soluções que otimizam a ADF. Neste capítulo explicaremos o funcionamento básico do AG e como nós o aplicamos ao nosso problema.

5.1 ALGORITMO GENÉTICO PARA UM INDIVÍDUO COM UM ÚNICO CROMOSSOMO

O AG tem esse nome devido à maneira como imita processos típicos de adaptação genética em seres vivos para resolver problemas de otimização que não necessitam ter qualquer

correlação com biologia. Os detalhes do sucesso do AG fogem do escopo desse trabalho (para isso, sugerimos (31)). Aqui vamos apresentar a estrutura do algoritmo a fim de que o leitor possa compreender melhor a ferramenta que utilizamos ao longo de todo o nosso estudo e que ajustes foram necessários para adequar o AG ao nosso problema. O ciclo do AG tem cinco etapas, as quais iremos discutir: definição de população, teste de aptidão, eliminação e clonagem, cruzamento, mutação. O fluxograma do AG pode ser visto na Figura 16. Como não iremos discutir exemplos além da nossa aplicação, sugerimos (45) como suporte didático.

Figura 16 – Fluxograma do AG.



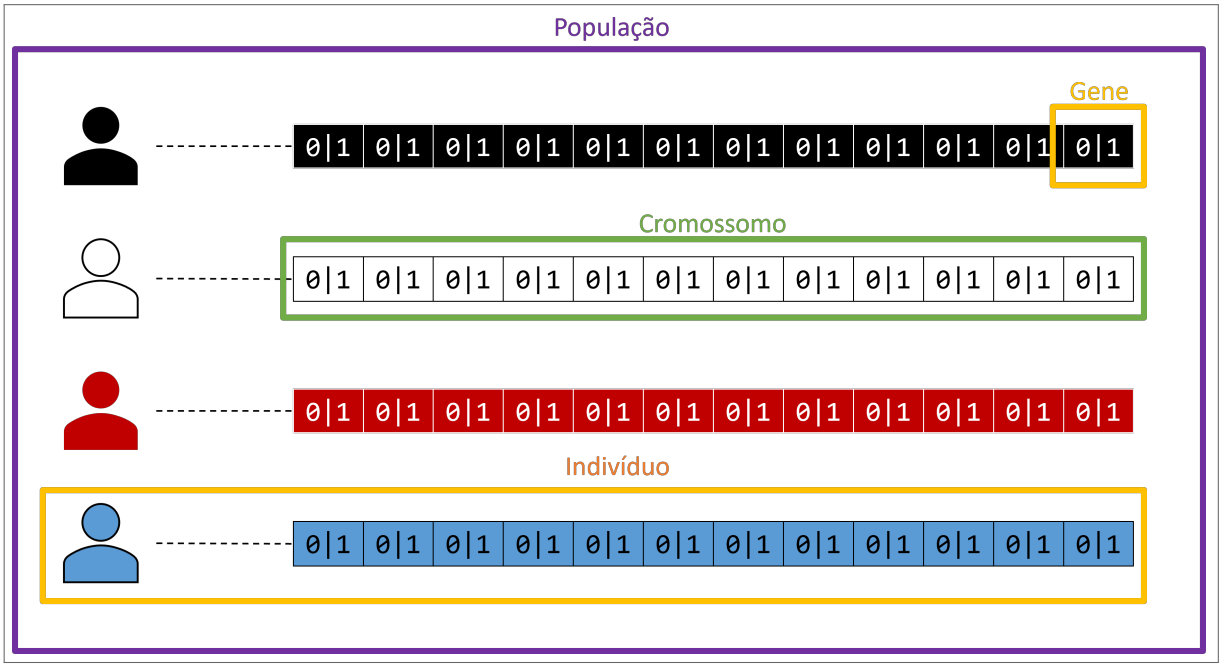
Fonte: Elaborado pelo autor (2022)

5.1.1 Definição da População

No lugar de uma varredura extensiva, o AG usa um pequeno conjunto de hipóteses, as quais serão modificadas ao longo das iterações do programa, a fim de investigar o espaço de soluções e encontrar a resposta ótima ao problema proposto. Esse conjunto de hipóteses é costumeiramente chamada de "população", fazendo alusão à inspiração do AG. Cada uma dessas hipóteses, as quais chamamos de "indivíduos", é constituída de um ou mais vetores (ou *arrays*) aos quais também dá-se o nome de "cromossomos". Cada um desses vetores é formado por um conjunto de valores binários, também chamados de "genes". Por fim, após uma população passar por um ciclo (ou *loop*) completo, reinicia-se o ciclo. Chamamos de "geração n" a população do n-ésimo ciclo. Um exemplo de população pode ser visto na Figura 17, onde está representada uma população de quatro indivíduos com doze genes (ou 12-bits).

Não há restrições nem demandas sobre as características da população na geração 0. Citando alguns exemplos, é possível iniciar com indivíduos todos idênticos, ou cada um deles ser uma hipótese que o programador julgar próxima da resposta final, ou simplesmente valores

Figura 17 – Exemplo de um conjunto de hipóteses (população) e sua estrutura interna. Cada hipótese é indivíduo com um vetor (cromossomo) cujas entradas (genes) são binários. Neste exemplo, temos uma população com 4 indivíduos de 12 genes (ou 12-bits).



Fonte: Elaborado pelo autor (2022)

sorteados aleatoriamente. As características da população inicial podem alterar o número de passos necessários para encontrar a resposta ótima, mas geralmente não alteram a qualidade do resultado se o algoritmo tiver tempo suficiente para atingir o ponto de saturação. Definir as características da população é o ponto de início do AG, assim como também é o ponto de reinício. Mas a partir da geração 1 as características são todas herdadas da geração anterior e das mudanças impostas pelos processos do ciclo.

5.1.2 Teste de Aptidão

Nesta etapa, o AG submete cada hipótese (indivíduo) às condições sob as quais desejamos obter uma resposta ótima. Qualquer quantidade pode ser utilizada como teste no AG, seja ele simulado ou experimental, bastando apenas que exista alguma correlação entre o mensurável e o parâmetro utilizado como cromossomo para que o AG seja capaz de buscar uma resposta ótima. Ao longo do nosso trabalho, iremos utilizar equações que modelam ADF como condições teste. Nosso objetivo é buscar maiores valores de população no estado excitado do átomo. Porém, no futuro, quando o experimento pronto para ser executado, a condição teste será modificada para uma medida direta do observável ao invés de equações que modelam o

problema. Já na referência (45), por exemplo, a condição teste é o valor da voltagem em um braço de alavanca, onde o autor busca valores de voltagem maiores.

Para executar o teste, converte-se a sequência binária que constitui o cromossomo em valor decimal (*float*) e utiliza-se esse valor decimal como entrada (*input*) no teste, que por sua vez terá um valor de saída (*output*). Esse valor de saída agora servirá como rótulo daquele indivíduo específico, dando ao AG uma medida de quão bom aquele indivíduo é em realizar aquela tarefa. Toda a população é submetida ao teste, o que encerra o processo de teste de aptidão.

5.1.3 Eliminação e Clonagem

Uma vez que temos todos os indivíduos rotulados, podemos eliminar os menos eficientes. Tipicamente elimina-se apenas um dos indivíduos a fim de não prejudicar a diversidade de hipóteses (indivíduos), mas algumas aplicações podem se beneficiar ao eliminar mais indivíduos ruins. O objetivo é manter um equilíbrio entre a diversidade de hipóteses e a capacidade de remover rapidamente hipóteses catastroficamente ruins. Esse equilíbrio é determinado principalmente pelo tamanho da população, mas outros fatores podem ser relevantes quando consideramos as especificidades de cada aplicação. Na nossa aplicação mantemos o padrão de eliminar apenas o pior valor resposta no teste, visto que nossa população não tem um grande número de indivíduos (apenas um grande número de cromossomos e genes).

Uma vez que eliminamos um indivíduo da nossa população, é necessário preencher o espaço vazio. Do contrário, a população iria diminuir a cada ciclo ficando sem nenhum indivíduo depois de apenas algumas gerações. Sendo assim, clona-se o indivíduo com melhor valor resposta ao teste, mantendo assim o número de indivíduos fixos ao longo de todas as gerações e dando algum favorecimento à melhor resposta daquela geração.

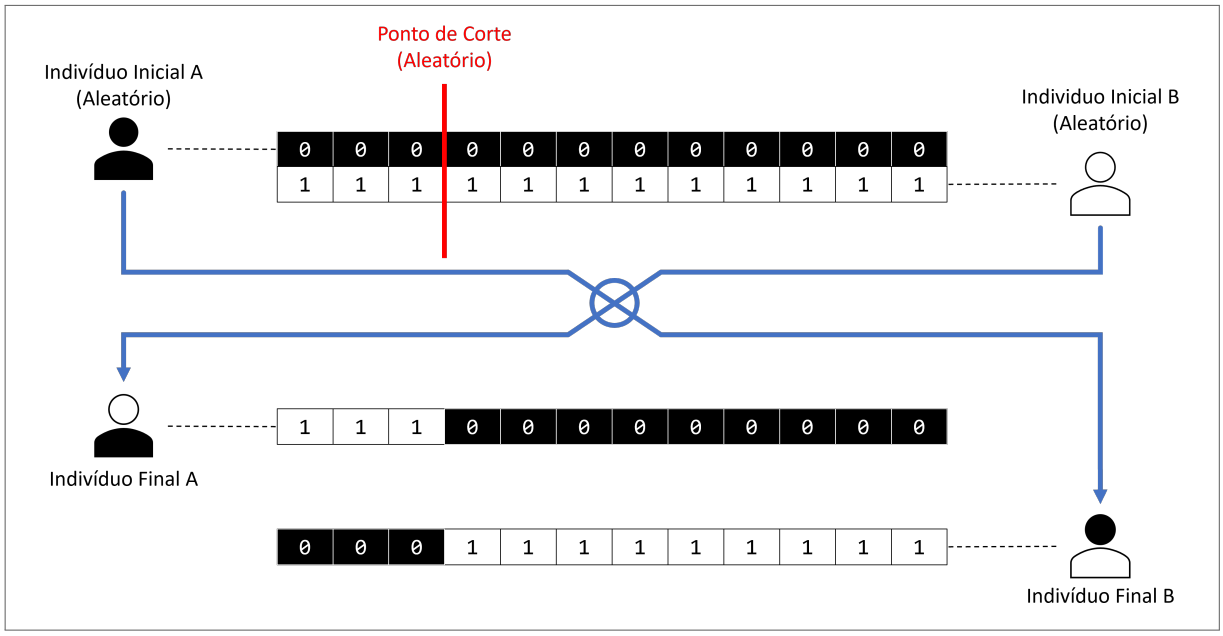
5.1.4 Cruzamento

Esse é o primeiro passo na formação da nova geração e é a etapa responsável pela investigação eficiente do espaço de soluções, misturando as informações presentes nos indivíduos. Essa tarefa pode ser executada de muitas maneiras, mas a forma mais comum é inspirada na troca de informação dos cromossomos nos seres vivos.

Na maneira padrão, escolhe-se dois indivíduos aleatoriamente e corta-se o cromossomo em

algum ponto também aleatório. O ponto de corte é o mesmo para cada par de cromossomos. Então, é feito uma permuta de informação usando o ponto de corte como pivô entre o par de cromossomos. Parte da informação é mantida e a outra parte é trocada com o outro cromossomo. A Figura 18 demonstra o funcionamento do cruzamento. No nosso código, a taxa de cruzamento entre os indivíduos é de 60%.

Figura 18 – Exemplo de cruzamento entre dois cromossomos. O cromossomo A tem genes iguais a 0 apenas para facilitar a visualização da troca. O mesmo vale para o cromossomo B.



Fonte: Elaborado pelo autor (2022)

5.1.5 Mutação

A etapa final do ciclo do AG e na definição da nova geração. É aqui onde se cria nova informação para dar condições ao AG de buscar respostas não disponíveis nas gerações anteriores. Para executar a mutação, escolhe-se um indivíduo aleatório e também escolhe-se um de seus genes aleatoriamente. Então, escreve-se um valor binário, aleatório, para aquele gene. Isso encerra a etapa de mutação.

A mutação é delicada pois ela modifica aleatoriamente os genes. Uma taxa muito alta de mutação pode impedir que o AG seja capaz de otimizar um processo. Uma taxa muito baixa de mutação pode fazer com que o tempo necessário para encontrar a resposta ótima seja desnecessariamente longo. Ainda assim, é comum manter uma taxa de mutação que modifique apenas um único gene por geração. Por exemplo, (45) utiliza uma taxa de 0.1% para seus 10

indivíduos de 12 genes, resultando em uma única mutação por geração. Nós utilizaremos uma taxa de mutação de 0.015% pois teremos um número grande de genes (20480), o que dará 4 mutações por geração. Porém, em outras aplicações é possível ter uma taxa de mutação mais alta, compensando-a com uma taxa de eliminação também maior.

5.2 INDIVÍDUOS COM MAIS DE UM CROMOSSOMO

Utilizando a definição padrão, com um indivíduo de um único cromossomo, não seria possível formar uma hipótese para o nosso problema de otimização com MEL. Um cromossomo gera apenas um único valor real, enquanto o MEL necessita de 128 valores para formar uma hipótese. Sendo assim, foi necessário adaptar a estrutura da população a fim de comportar nossas hipóteses. Essa mudança levou a pequenas modificações nos processos do AG, as quais iremos explicar em seguida.

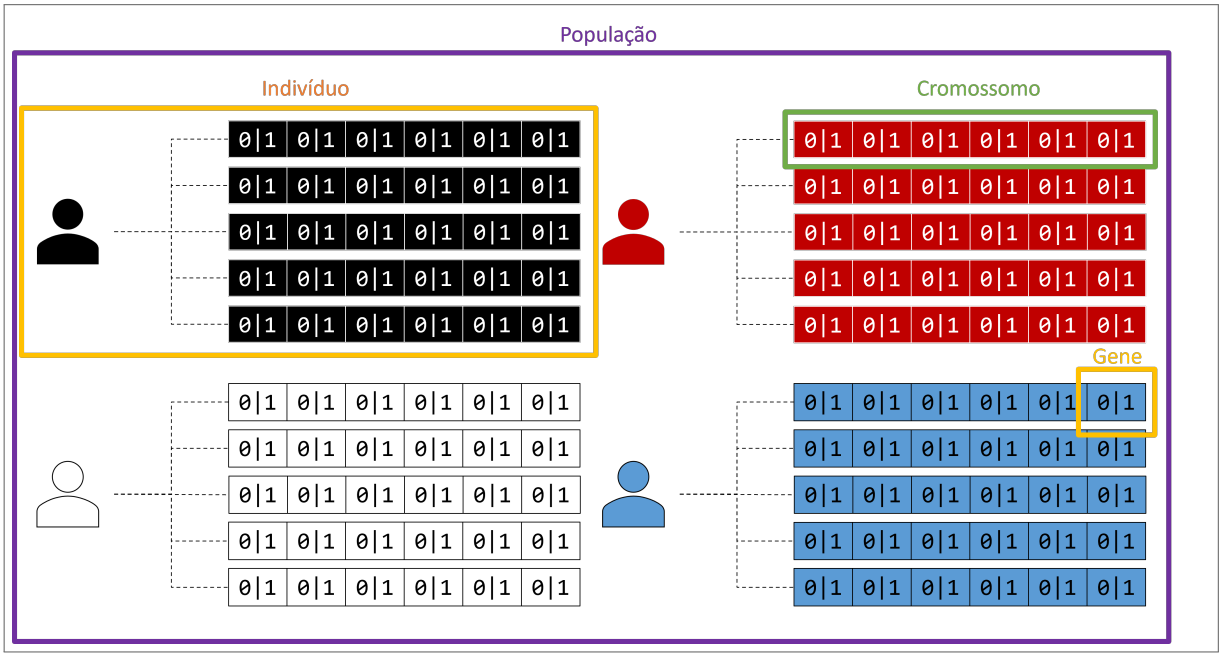
5.2.1 População II

Buscando manter a inspiração do AG intacta, optamos por adicionar mais cromossomos ao nosso indivíduo, assim como os seres vivos tem vários cromossomos. Cada um desses cromossomos agora representa uma das entradas do MEL e o indivíduo representa nossa hipótese de forma apropriada. A Figura 19 traz um exemplo com a nova estrutura. No nosso problema, utilizaremos uma população com 20 indivíduos de 128 cromossomos compostos por 8 genes (8-bits).

5.2.2 Teste de Aptidão II

Assim como antes testávamos hipóteses, também testaremos hipóteses na versão adaptada. Porém, nossa hipótese (indivíduo) agora não é mais constituída de um único cromossomo, mas sim de vários. Sendo assim, no processo de teste de aptidão, testaremos o indivíduo utilizando cada um dos seus cromossomos simultaneamente. Da mesma forma que antes, o valor do resultado do teste continuará sendo atribuído ao indivíduo como um todo.

Figura 19 – Exemplo de um conjunto de hipóteses (população) e sua estrutura interna. Cada hipótese (indivíduo) agora é um conjunto de vetores (cromossomos) cujas entradas (genes) são binários. Neste exemplo, a população é composta de 4 indivíduos com 5 cromossomos de 6 genes (ou 6-bits).



Fonte: Elaborado pelo autor (2022)

5.2.3 Eliminação e Clonagem II

Seguindo o mesmo raciocínio do teste de aptidão, eliminaremos um indivíduo (com todos os seus cromossomos) no processo de eliminação e, consequentemente, precisaremos repor um indivíduo (com um conjunto inteiro de cromossomos) no processo de clonagem.

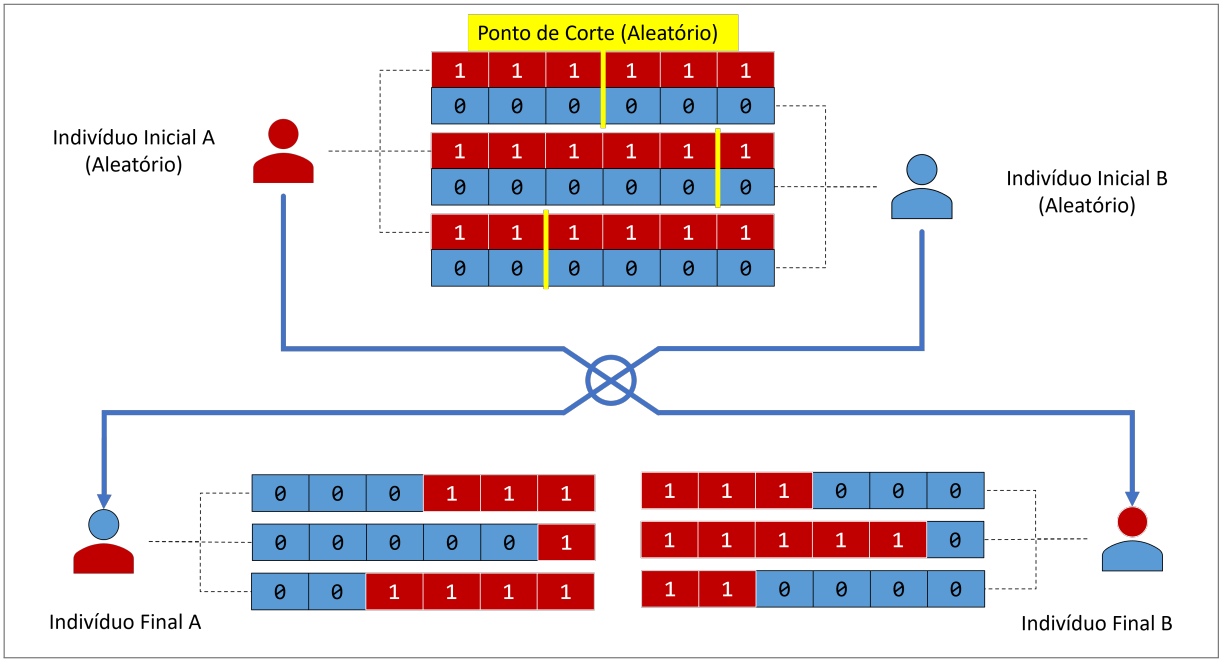
5.2.4 Cruzamento II

Esse foi o processo mais afetado pela adaptação. Para realizar o cruzamento, selecionamos dois indivíduos aleatoriamente e as trocas de informação são realizadas entre pares de cromossomos da mesma entrada. Além disso, cada uma das entradas tem um ponto de corte próprio. A Figura 20 exemplifica com um cruzamento entre dois indivíduos de 3 cromossomos com 6 genes.

5.2.5 Mutação II

A mutação foi o processo menos modificado, pois trata-se apenas de uma escolha aleatória de um gene. A única diferença agora é que antes de escolhermos um cromossomo precisamos

Figura 20 – Exemplo de cruzamento entre dois indivíduos. O indivíduo A tem genes iguais a 0 apenas para facilitar a visualização da troca. O mesmo vale para o indivíduo B.



Fonte: Elaborado pelo autor (2022)

escolher um indivíduo. Sendo assim, na versão adaptada da mutação escolhe-se um indivíduo aleatoriamente, depois escolhe-se um de seus cromossomos aleatoriamente e, dentro desse cromossomo, é escolhido um gene aleatoriamente para receber um valor binário aleatório.

6 FASE ESPECTRAL ÓTIMA PARA ABSORÇÃO DE DOIS FÓTONS RESSONANTE DE UM ÚNICO PULSO

6.1 SISTEMA E MODELO

Vamos iniciar nossas análises a partir das condições específicas propostas por Dudovich et al. (34), por termos o conhecimento prévio da literatura e também por serem mais simples, com um único laser. Isso nos permitirá introduzir a nossa abordagem de forma didática ao comparar os resultados do AG com a solução proposta em 2001. O sistema em questão é um átomo de Rb, cujas transições $5S_{1/2} \rightarrow 5P_{3/2}$, de 780.2 nm, e $5P_{3/2} \rightarrow 5D_{5/2,3/2}$, de 776.0 nm, com um pulso centrado em 778.1 nm de 18 nm de largura (FWHM) compõem os dados de interesse para modelo. O MEL tem 128 entradas e o laser passa por uma grade de difração, permitindo que seja colocado sobre o MEL com intervalos de 1 rad/ps de espectro ótico em cada entrada. O sistema está representado na Figura 21. Na Equação 4.32, essas condições são o equivalente a fazer $\tilde{\mathcal{E}}_s(\omega) = \tilde{\mathcal{E}}_c(\omega) = \tilde{\mathcal{E}}_l(\omega)$ e $\tau = 0$. Com isso, podemos escrever

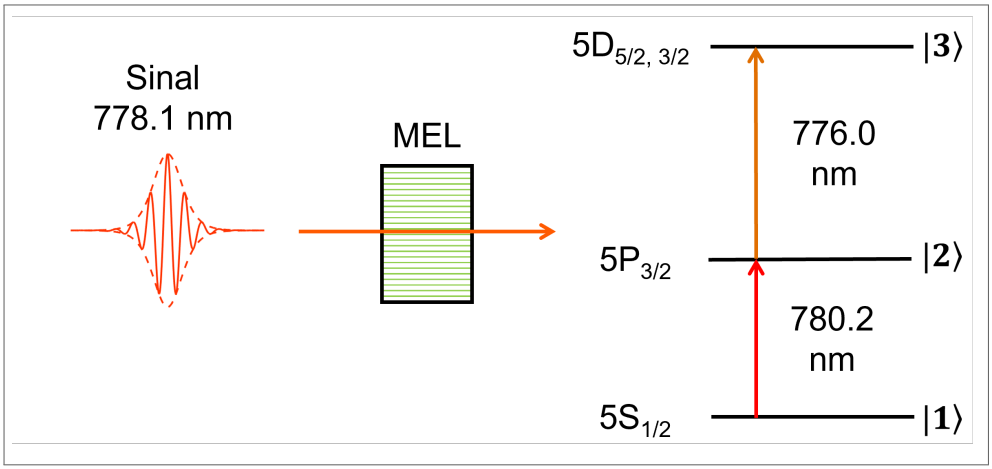
$$c_e^{RFI}(\tau) \approx -\frac{\mu_{12}\mu_{23}}{i\hbar^2}\left[i\pi\tilde{\mathcal{E}}_l^*(\omega_{12}-\omega_l)\tilde{\mathcal{E}}_l^*(\omega_{23}-\omega_l) + \text{p.v.}\int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_l^*(\omega')\tilde{\mathcal{E}}_l^*(-\omega')}{(\omega_{12}-\omega_l)-\omega'}\right]. \quad (6.1)$$

Embora escrever no espaço das frequências nos permita ter uma melhor leitura do problema, a simulação é realizada utilizando o formato temporal da equação, baseado na equação 4.11, que fica

$$c_e(\tau) \approx -\frac{\mu_{12}\mu_{23}}{\hbar^2}\int_{-\infty}^{\infty}\int_{-\infty}^{t'} dt'dt'' e^{i(\omega_{23}-\omega_l)t'} e^{i(\omega_{12}-\omega_l)t''} \mathcal{E}_l^*(t')\mathcal{E}_l^*(t''), \quad (6.2)$$

Como foi dito no início do Capítulo 4, e Equação 6.1 foi utilizada para propor uma solução ao problema de interferência destrutiva da ADF. Visto que há uma diferença de fase de π entre os termos de frequência somados acima e abaixo da singularidade, é possível usar um MEL para realizar uma correção de fase espectral e tirar proveito da amplificação dada pelo polo da integral. Mas aqui é preciso ter dois cuidados especiais. Como um único pulso está sendo utilizado para realizar as duas transições, a fase aplicada pelo MEL no campo será contabilizada duas vezes. Dentro da integral, na Equação 6.1, o campo $\tilde{\mathcal{E}}_l^*$ aparece duas vezes, substituindo o campo $\tilde{\mathcal{E}}_s^*$ e o campo $\tilde{\mathcal{E}}_c^*$ da Equação 4.32. Por essa razão, a mudança de fase aplicada sobre o campo será de $\pi/2$, ao invés de uma mudança de π.

Figura 21 – Sistema do átomo de Rb para transição ressonante de dois fótons com um único laser, que irá ceder os dois fótons necessários para as transições $5S_{1/2} \rightarrow 5P_{3/2}$, de 780.2 nm, e $5P_{3/2} \rightarrow 5D_{5/2,3/2}$, de 776.0 nm. O laser é centrado em 778.1 nm com 18 nm de largura (FWHM). O MEL, que tem 128 entradas com resolução espectral de 1 rad/ps, aplicará a mudança de fase de acordo com o determinado pelo procedimento do AG.



Fonte: Elaborado pelo autor (2022)

Além disso, as alterações devem ser aplicadas em torno de cada frequência de transição, formando uma janela de fase de, ao invés de uma simples alteração de π em torno de uma única singularidade. Os pontos de transição são $\omega_{12} - \omega_l$ e em $\omega_{23} - \omega_l = -(\omega_{12} - \omega_l)$. A segunda singularidade pode ser verificada observando a Equação 4.19. Se mudamos nossa perspectiva na utilização de $\delta((\omega_{ne} - \omega_c) + (\omega_{fn} - \omega_s) + \omega' + \omega'')$, integrando em ω' ao invés de ω'', temos como resultado

$$c_e(\tau) \approx \sum_n \frac{\mu_{fn}\mu_{ne}}{i\hbar^2} e^{i[\omega_c - (\omega_{ne} - \omega_c)]\tau} \times \int_{-\infty}^{\infty} d\omega'' \frac{\tilde{\mathcal{E}}_c^*(-[(\omega_{ne} - \omega_c) + (\omega_{fn} - \omega_s) + \omega''])\tilde{\mathcal{E}}_s^*(\omega'')}{(\omega_{ne} - \omega_c) - \omega''} e^{-i\omega''\tau}. \quad (6.3)$$

Note que a mudança de perspectiva também mudou a posição do polo. A equação continua representando as mesmas quantidades e valores, apenas deixamos de olhar para o problema da perspectiva do pulso de sinal e passamos a observar da perspectiva do pulso de controle. Como cada pulso atua em uma transição diferente a aparência da equação muda e, da mesma forma, a singularidade apontada por cada equação também deve mudar. De uma forma ou de outra, o comportamento integrado é o mesmo nos dois casos. Aplicando as condições que usamos nesse capítulo, observamos o surgimento da singularidade que buscávamos. Fazendo $\tilde{\mathcal{E}}_s(\omega) = \tilde{\mathcal{E}}_c(\omega) = \tilde{\mathcal{E}}_l(\omega)$ e $\tau = 0$ na Equação 6.3, chegamos a

$$c_e^{RFI}(\tau) \approx \frac{\mu_{12}\mu_{23}}{i\hbar^2}\left[i\pi\tilde{\mathcal{E}}_l^*(\omega_{23}-\omega_l)\tilde{\mathcal{E}}_l^*(\omega_{12}-\omega_l) + \text{p.v.}\int_{-\infty}^{\infty} d\omega'' \frac{\tilde{\mathcal{E}}_l^*(-\omega'')\tilde{\mathcal{E}}_l^*(\omega'')}{(\omega_{23}-\omega_l)-\omega''}\right], \quad (6.4)$$

que é equivalente a expressão 6.1 na perspectiva do pulso que realiza a transição superior. Mas, nesse caso, o pulso que realiza a transição superior é o mesmo que realiza a transição inferior. Como consequência, vemos que o numerador e resíduos são idênticos entre as equações. O que surpreende é a singularidade, que não permaneceu a mesma. Ainda que o comportamento integrado seja o mesmo, essa sutileza está nos dizendo que o mesmo pulso experimenta duas singularidades diferentes. E duas singularidades implica na necessidade de dois ajustes de fase.

Do ponto de vista fenomenológico, a sutileza da janela de fase presente nas equações é mais simples de ser compreendida. Visto que temos uma diferença de fase de π entre as frequências acima e abaixo da ressonância quando o pulso excita uma transição, se um único pulso excita duas transições simultaneamente é razoável supor que ele experimentará duas diferenças de fase de π, uma para cada transição que está sendo excitada.

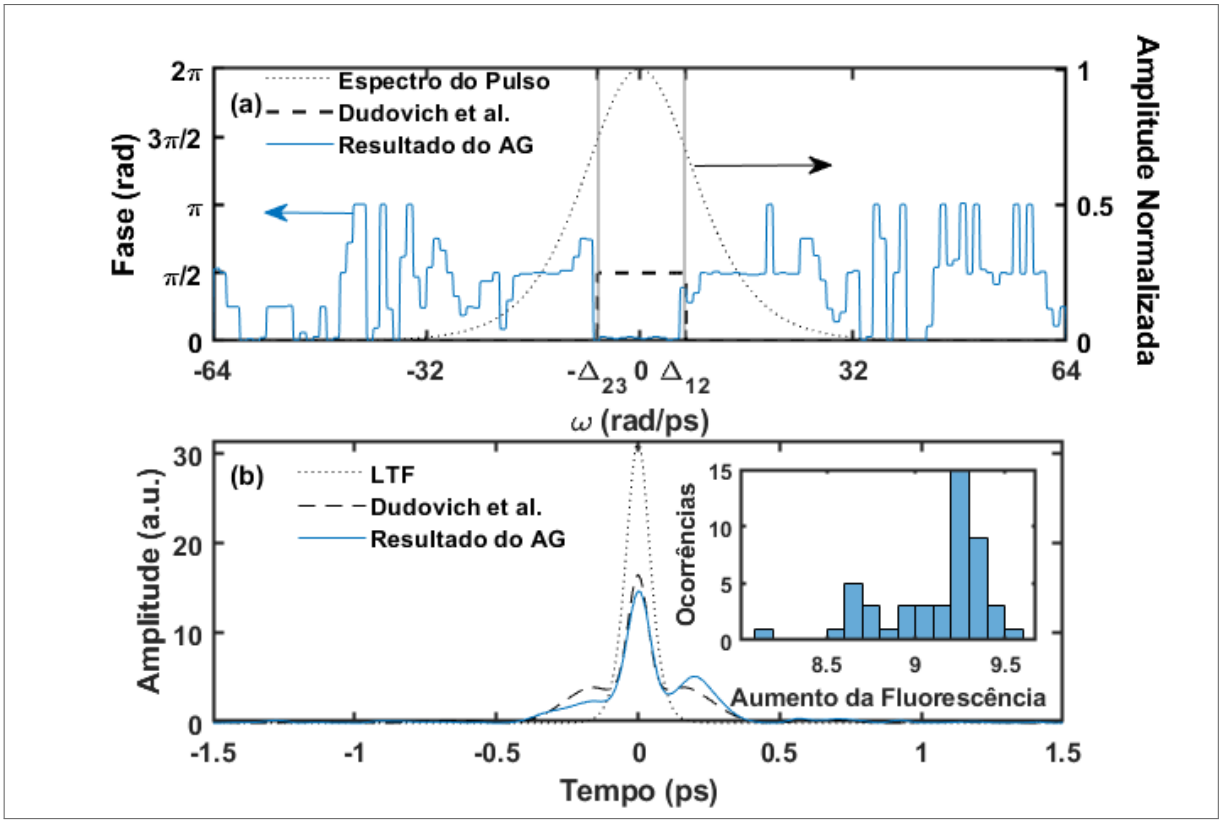
Porém, todas essas são informações conhecidas (34). Nosso interesse é utilizar esse conhecimento como meio de averiguação do nosso método, que faz uso do AG, enquanto buscamos entender as características dos resultados obtidos pelo AG e como eles estão relacionadas com o modelo matemático.

6.2 RESULTADOS COM ALGORITMO GENÉTICO

De posse da Equação 6.2, que utilizamos para testar o desempenho de cada hipótese do AG nesta sessão, a simulação aplica alterações de fase espectral ao pulso antes de aplicá-lo ao modelo. Realizamos uma série de simulações e armazenamos o melhor resultado ao final de cada uma. O código base para as simulações pode ser visto no Anexo A. Agora iniciamos a análise das simulações com a Figura 22 (a), que é a melhor Máscara de Fase (MF) obtida pelo AG com a Equação 6.2 após 48 simulações de 5000 iterações, em azul sólido. Podemos imediatamente verificar que o AG também recorreu a uma solução com uma janela de fase de $\pi/2$ entre as frequências ressonantes, assim como proposto em (34) e comentado na secção anterior. A linha preta pontilhada é o espectro do pulso, para guiar o leitor sobre onde está sendo aplicada cada mudança de fase. No eixo horizontal, $\Delta_{ij} = \omega_{ij} - \omega_l$. A janela de fase proposta em (34) pode ser vista em preto tracejado. A solução encontrada pelo AG está invertida em relação a proposta de Dudovich et al., mas esse fato não constitui

um problema pois apenas a diferença de fase é relevante, como vimos nas equações que descrevem o problema. A investigação aleatória do AG leva a alterações de fase também nas frequências fora da região central. Mas para cada alteração aplicada em uma região cria-se um comprometimento nas outras. Dessa forma, a MF fora da região central acaba com uma aparência aleatória, oriunda das primeiras mudanças aplicadas pelo AG.

Figura 22 – Resultados das otimização do AG a partir da Equação 6.2. No eixo horizontal, $\Delta_{ij} = \omega_{ij} - \omega_l$. (a) Melhor MF, em azul sólido, obtida nas simulações após 5000 iterações. Em preto tracejado está uma janela de fase de $\pi/2$, indicando a solução proposta por Dudovich et al. (34). Em preto pontilhado temos o espectro do pulso, para ajudar o leitor a entender onde cada mudança de fase está sendo aplicada. (b) Comparação da forma temporal do pulso após aplicação da MF mostrada em (a), em azul sólido, e o proposto por Dudovich et al. (34), em preto tracejado. O pulso LTF está em preto pontilhado. A figura inserida mostra a distribuição de resultados de 48 simulações com o AG. A normalização é em relação a fluorescência de um pulso LTF.



Fonte: Elaborado pelo autor (2022)

A Figura 22 (b) compara a forma temporal do pulso, em azul sólido, após a aplicação da MF em (a) com a forma temporal obtida pela proposta de Dudovich et al. (34), em preto tracejado. A semelhança é clara, o que reforça que o AG de fato propõe uma abordagem semelhante a de (34). As pequenas diferenças são remanescentes da investigação aleatória do algoritmo, mas o aumento na fluorescência mostra que essas diferenças são desprezíveis. Em números, após 48 simulações, os resultados finais de amplificação da fluorescência obtiveram um fator multiplicativo médio de 9.1, $\sigma = 0.3$ e máximo de 9.5, enquanto a MF proposta por Dudovich

et al. obteve fator de amplificação de ~ 7. A normalização é em relação a fluorescência de um pulso LTF aplicado na Equação 6.2. Isso revela que o AG não apenas é capaz de nos apontar interpretações analíticas para o problema como também de explorar detalhes menores que podem fornecer um resultado melhor do que o previsto. Também iremos nos utilizar das informações dadas pela figura inserida para evitar a necessidade de executar a simulação tantas vezes. Utilizando os dados do histograma podemos calcular que a probabilidade de uma dentre três simulações obter um resultado entre 9.2 e 9.6 é de $\sim 93\%$, mostrando a eficiência do método. Dessa forma, mantivemos um padrão de três execuções por sistema simulado e, para os capítulos seguintes, iremos nos concentrar apenas no melhor resultado. Finalmente, é importante apontar que sempre deixamos o AG executar suas iterações até que não fosse possível obter respostas melhores. Dessa forma, ao analisar todos os conjuntos de dados, somos capazes de ver a qualidade da convergência do algoritmo. Embora apenas a melhor solução possível seja do nosso interesse, essas informações nos dão confiança de que as modificações necessárias para adaptar o AG para o nosso problema, explicadas no Capítulo 5, não impactaram a qualidade das respostas. Ainda assim, para obter esses resultados foi necessário tomar alguns cuidados com as simulações. Falaremos brevemente sobre eles agora.

6.3 PARÂMETROS DA SIMULAÇÃO

Ao longo das simulações, foi necessário definir alguns parâmetros de operação, sendo eles a eficiência espectral da MF, o tamanho da janela temporal e a resolução temporal. A eficiência espectral da MF é limitada pela resolução espectral ao longo de todo o experimento. Como visamos realizar um experimento para comparar com nossas simulações, observamos os detalhes dos instrumentos disponíveis no laboratório e verificamos que estávamos limitados a uma resolução de 0.5 rad/ps pela grade de difração. Embora essa informação não seja exatamente uma questão numérica, a maneira de aplicar é. Existem formas diferentes de implementar a limitação na resolução instrumental dentro da simulação. Mas, visto que o MEL é discreto e que sua eficiência nas bordas de cada entrada não é a mesma em relação a região central, optamos por utilizar a limitação na resolução como parâmetro de suavização entre as entradas de cada frequência da MF, utilizando a resolução espectral como um fator na eficiência espectral da MF. Então, escolhemos a função logística para realizar a suavização pela sua capacidade de modelar um grande número de comportamentos naturais. A função logística é dada por

$$f(x) = \frac{M}{1 + e^{-A(x-x_0)}}, \tag{6.5}$$

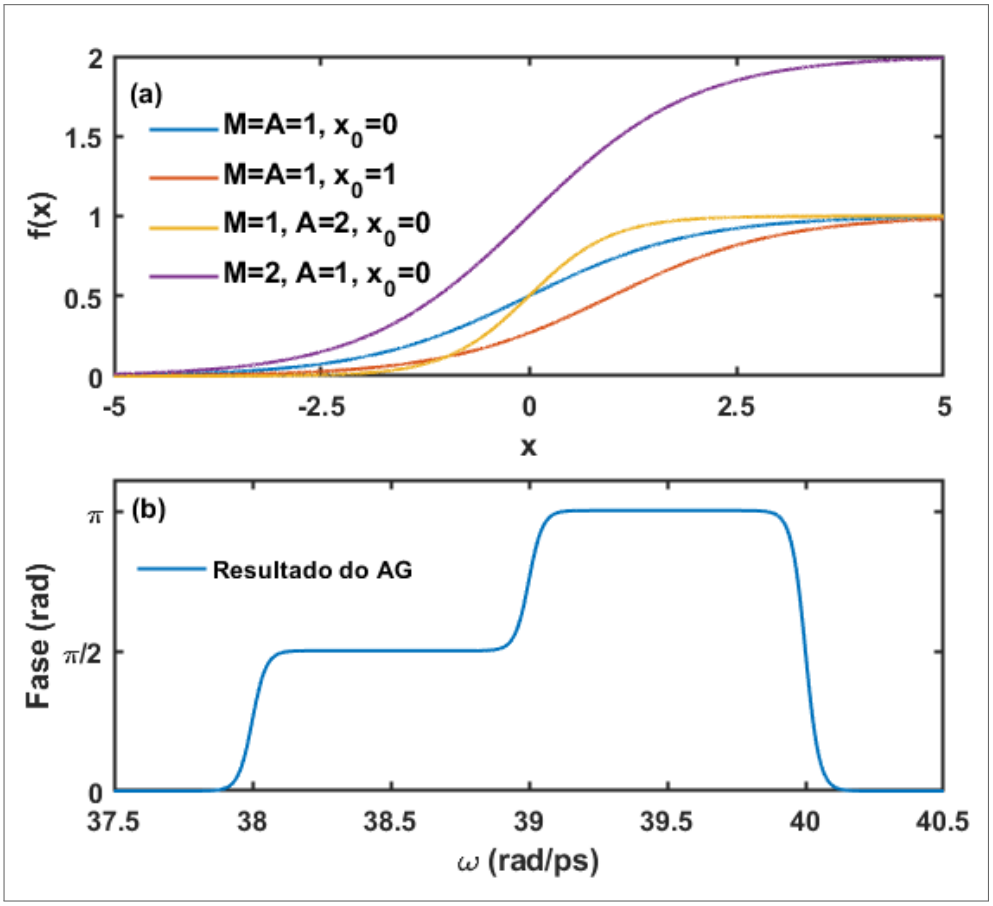
onde M é o valor máximo da função, A é a inclinação da curva e x_0 é o ponto no qual a função atinge o valor $M/2$.

Alguns exemplos de manipulação com a função logística podem ser vistos na Figura 23 (a). Com isso, nós emulamos a combinação de falta de resolução espectral e efeitos de borda em cada entrada do MEL na aplicação das mudanças de fase no pulso. A Figura 23 (b) mostra uma ampliação da Figura 22 (a) para que o leitor seja capaz de visualizar como a suavização é aplicada na MF. No uso da função logística, em cada intervalo de 1 rad/ps mantemos a região central fixa e variamos apenas as bordas. Visto que nossa resolução é de 0.5 rad/ps, tomamos 0.25 rad/ps de cada borda para fazer a suavização com a função logística, mantendo o centro inalterado. Dessa forma, os valores de M e x_0 dependem apenas do ponto no qual a suavização está sendo aplicada. Porém, o valor de A é escolhido como 20/0.5 (=40), para que a suavização não crie degraus em relação a região central, que não é alterada, enquanto usamos o próprio valor de erro para determinar a inclinação da curva.

Uma outra preocupação durante as simulações é saber se a janela temporal (que também pode ser interpretada como resolução espectral) considerada é suficiente para avaliar o problema corretamente. Para a simulação apresentada na Figura 22, testamos a estabilidade numérica variando o número de pontos. Os resultados dessa análise podem ser vistos na Figura 24. Note como é necessário um número mínimo de pontos para que as soluções fiquem na mesma região. As flutuações menores são esperadas devido à natureza probabilística do AG. Após avaliar esse gráfico, decidimos utilizar $\sim$ 32000 pontos na simulação da Figura 22. Queremos deixar claro que esse número de pontos foi utilizado especificamente nessa simulação. A cada mudança de condições, como veremos nos próximos capítulos, é necessário realizar uma nova verificação de estabilidade da solução com o número de pontos. Porém, não iremos apresentar esses dados para as próximas simulações, visto que tornariam a discussão lenta e desviariam o nosso foco. Aqui estamos apenas advertindo sobre a necessidade de fazer essas verificações em caso de reprodução dos nossos resultados.

O último parâmetro a ser observado é a resolução temporal (que também pode ser interpretada como janela espectral). Neste caso, fixamos a janela temporal em $\sim$ 320 ps (baseado no resultado anterior) e variamos a resolução temporal. Para manter a janela temporal em 320 ps é necessário aumentar o número de pontos na mesma proporção que aumentamos a resolução

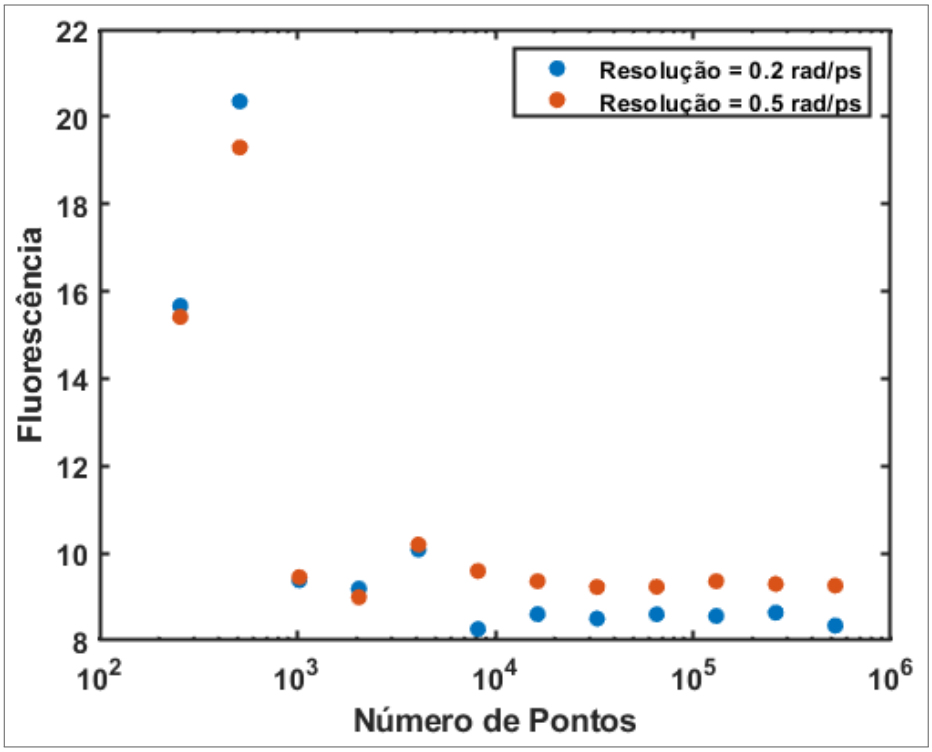
Figura 23 – (a) Exemplos de manipulação da função logística. M é o máximo da função, A é inclinação da curva e x_0 é o ponto onde a função atinge o valor $M/2$. (b) Ampliação da MF mostrada na Figura 22 (a), evidenciando a aplicação da equação logística na suavização da MF simulada.



Fonte: Elaborado pelo autor (2022)

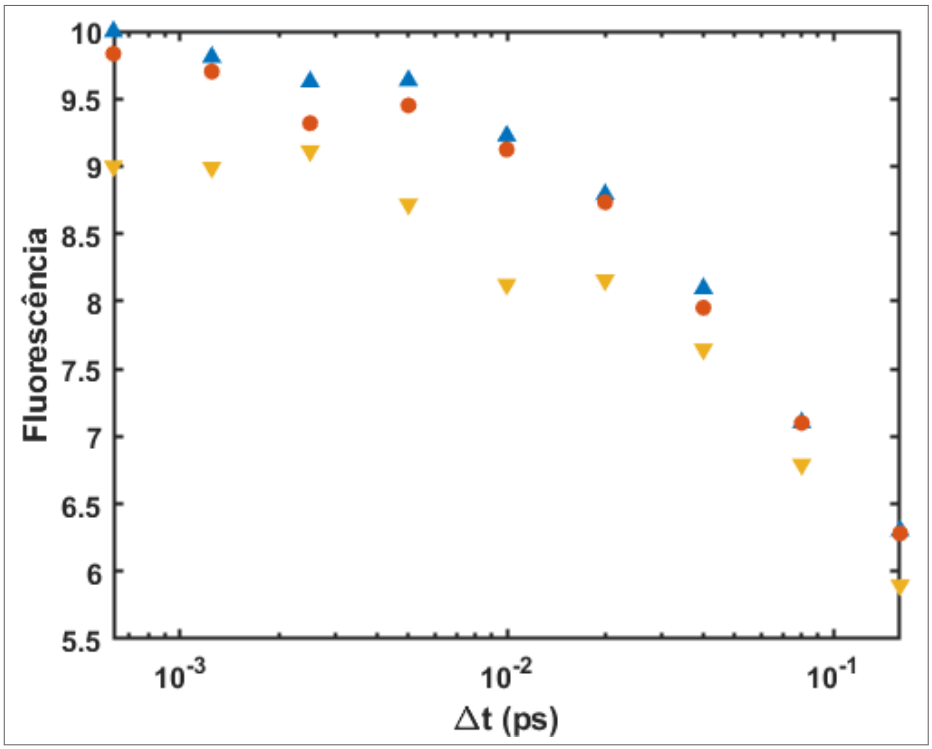
temporal. Com isso, temos o resultado da Figura 25, na qual vemos que é importante manter a resolução temporal abaixo de 0.01 ps. Dessa forma, utilizamos uma resolução de 0.005 ps para os dados da Figura 22. A resolução temporal também é um parâmetro que precisa ser testado sempre que as condições do problema mudam. Assim como com a janela temporal, não iremos apresentar os dados de verificação de estabilidade nos próximos capítulos. Alertamos apenas para a necessidade de realizar tais verificações.

Figura 24 – Dependência do resultado do AG com o número de pontos da simulação. A resolução temporal da simulação é fixada em 0.01 ps. Incluímos resultados com dois valores diferentes de resolução espectral para evidenciar a importância de também determinar esse parâmetro na simulação.



Fonte: Elaborado pelo autor (2022)

Figura 25 – Dependência do resultado do AG com a resolução temporal. A janela temporal foi fixada em ∼320 ps. Incluímos resultados de algumas simulações para cada valor de resolução temporal a fim de que o leitor tenha uma percepção da variação típica do AG.



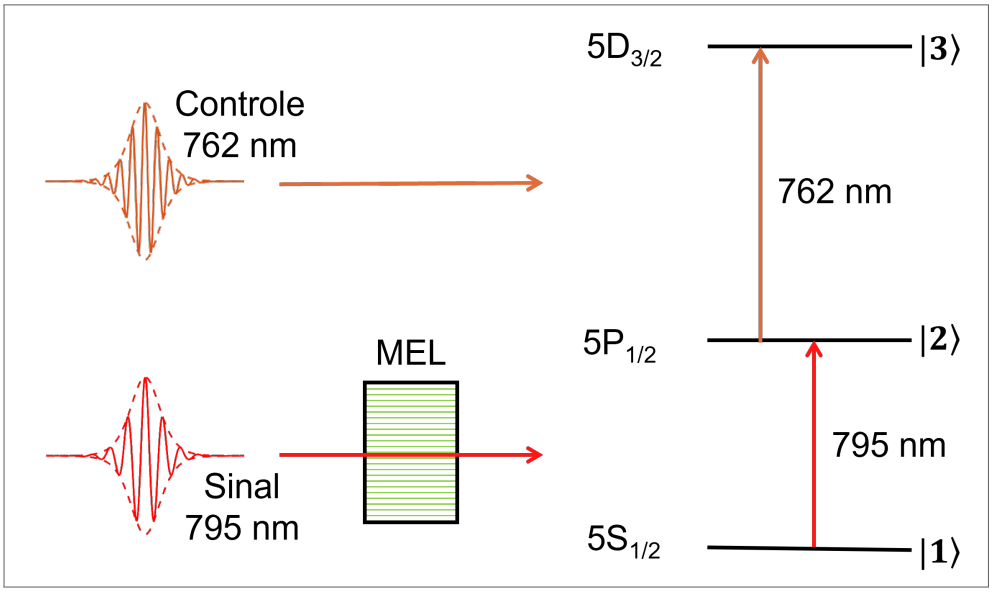
Fonte: Elaborado pelo autor (2022)

7 FASE ESPECTRAL ÓTIMA PARA ABSORÇÃO DE DOIS FÓTONS COM DOIS LASERS

7.1 SISTEMA E MODELO

Com o AG fornecendo respostas compreensíveis e eficientes, podemos utilizá-lo como ferramenta para iniciar a investigação do problema de otimização da Absorção Sequencial de Dois Fótons (ASDF). Visto que temos a intenção de entender a otimização de fase espectral para o problema estudado por Carvalho et al. (25), vamos utilizar o mesmo sistema atômico, mostrado na Figura 26. Temos um átomo de Rb cujas transições $5S_{1/2} \rightarrow 5P_{1/2}$, de 795.0 nm, e $5P_{1/2} \rightarrow 5D_{3/2}$, de 762.0 nm serão utilizadas para nossa simulação. Dessa vez o laser de sinal, centrado em 795.0 nm com 7.5 nm de largura (FWHM), e o laser de controle, centrado em 762.0 nm com 10.4 nm de largura (FWHM), estão em perfeita ressonância com as transições atômicas. O mesmo MEL da seção anterior aplicará a mudança de fase, mas apenas no pulso de sinal. Esse formato é necessário para manter relação com o experimento realizado por Carvalho et al., cujo modelo considera um pulso de controle sem modificações (25).

Figura 26 – Sistema do átomo de Rb para transição ressonante de dois fótons com dois lasers. O laser de sinal irá ceder o fóton necessário para a transição $5S_{1/2} \rightarrow 5P_{1/2}$, de 795.0 nm, e o laser de controle cederá o fóton necessário para a transição $5P_{1/2} \rightarrow 5D_{3/2}$, de 762.0 nm. O laser de sinal é centrado em 795.0 nm com 7.5 nm de largura (FWHM) e o laser de controle é centrado em 762.0 nm com 10.4 nm de largura (FWHM). O MEL, que tem 128 entradas com resolução espectral de 1 rad/ps, aplicará a mudança de fase, apenas no pulso de sinal, de acordo com o determinado pelo procedimento do AG.



Fonte: Elaborado pelo autor (2022)

Para o modelo, essas novas condições são o equivalente a fazermos $\omega_{12} - \omega_s = \omega_{23} - \omega_c = 0$.

A Equação 4.32 então fica

$$c_e^{RFI}(\tau) \approx -\frac{e^{i\omega_c\tau}}{i\hbar^2}\mu_{12}\mu_{23}\left[i\pi\tilde{\mathcal{E}}_c^*(0)\tilde{\mathcal{E}}_s^*(0) - \text{p.v.}\int_{-\infty}^{\infty} d\omega' \frac{\tilde{\mathcal{E}}_c^*(\omega')\tilde{\mathcal{E}}_s^*(-\omega')}{\omega'}e^{-i\omega'\tau}\right], \tag{7.1}$$

e no espaço do tempo

$$c_e(\tau) \approx -\frac{\mu_{12}\mu_{23}}{\hbar^2}e^{i\omega_c\tau}\int_{-\infty}^{\infty}\int_{-\infty}^{t'+\tau} dt'dt''\mathcal{E}_c^*(t')\mathcal{E}_s^*(t''), \tag{7.2}$$

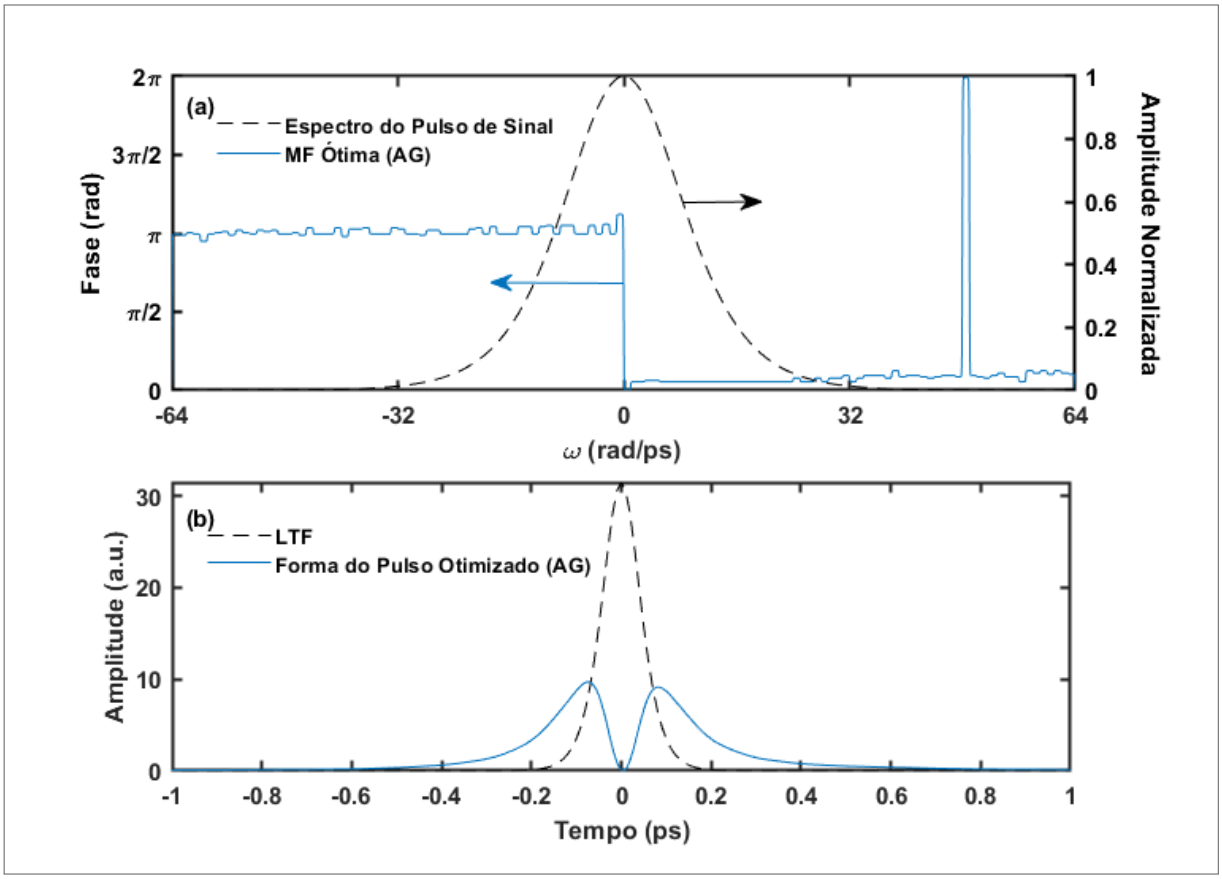
onde já percebemos que ambas as expressões se tornam muito mais simples. A singularidade que é localizada no centro do espectro é nossa fonte de amplificação. Isso é muito vantajoso pois esse é o ponto com maior valor de amplitude (pico de densidade energética), o que nos dará condições de termos valores de absorção ainda mais altos. Ter dois lasers, cada um atuando em apenas uma transição, implica que o problema de duas singularidades e dois saltos de fase não existirá mais. Porém, agora precisamos considerar também o atraso entre os pulsos, representado por τ, dentro da integral. Observando a Equação 7.1 vemos que o atraso cria uma fase espectral linear para a integração, de forma que temos agora duas características de fase que precisam ser ajustadas: o degrau de fase gerado pela singularidade e uma fase linear cuja inclinação é ditada por τ.

7.2 RESULTADOS

Utilizando a Equação 7.2 para o caso mais simples, com $\tau = 0$, obtemos os resultados da Figura 27. Em (a) temos a máscara de fase que otimiza a absorção de dois fótons, em azul sólido. Em preto tracejado está o espectro do pulso de sinal. Note que, de fato, o AG buscou uma máscara com apenas um salto de fase. Em (b) temos a forma do pulso, em azul sólido, após a aplicação da MF em (a). Em preto tracejado está o pulso LTF. A forma do pulso é diferente do que temos na Figura 22 (b). Ao invés de termos dois mínimos locais, agora temos apenas um mínimo central. Temos a suposição que o número de vales na forma temporal do pulso está relacionado ao número de transições de fase espectral. Na Figura 27, cuja fase espectral tem apenas um salto de π, temos apenas um único vale na sua forma temporal. Em contrapartida, na Figura 22, temos dois saltos espectrais e dois vales na forma temporal do pulso. Isso nos leva a crer que o número saltos de fase está relacionado ao número de vales na forma temporal, mas não houve interesse em investigar essa relação a fundo. Deixamos essa investigação como sugestão para futuros trabalhos. Focando agora na diferença mais

surpreendente desse novo resultado, o aumento na absorção chegou a ~ 26 vezes, indo muito além do que foi obtido no capítulo anterior. Podemos fazer uma comparação desse valor com uma outra situação padrão, como segue.

Figura 27 – Resultados das otimização do AG a partir da Equação 7.2 com $\tau = 0$. (a) Melhor máscara de fase, em azul sólido, obtida nas simulações após 5000 iterações. Em preto tracejado está o espectro do pulso. (b) Comparação da forma temporal do pulso após aplicação da MF mostrada em (a), em azul sólido. O pulso LTF está em preto tracejado.



Fonte: Elaborado pelo autor (2022)

Sabemos que o atraso entre os pulsos (τ) é um fator importante na ASDF (25). Mesmo para um par de pulsos LTF, podemos ver na Figura 28 (a) que o atraso tem a capacidade de anular e também de amplificar a transição. A figura mostra o comportamento da ASDF sob diferentes condições de atraso entre os pulsos. O atraso é aplicado no pulso de controle, como proposto na Equação 4.1, e varia em um intervalo de $[-0.5, 1.5[$ ps. A normalização é feita em relação a situação com $\tau = 0$. Note que para valores negativos (pulso de controle chegando antes do pulso de sinal) a ADF se torna menos eficiente ou mesmo impossível. Enquanto isso, para valores positivos (pulso de controle chegando depois do pulso de sinal) temos uma amplificação do sinal de fluorescência que pode chegar até a quatro vezes.

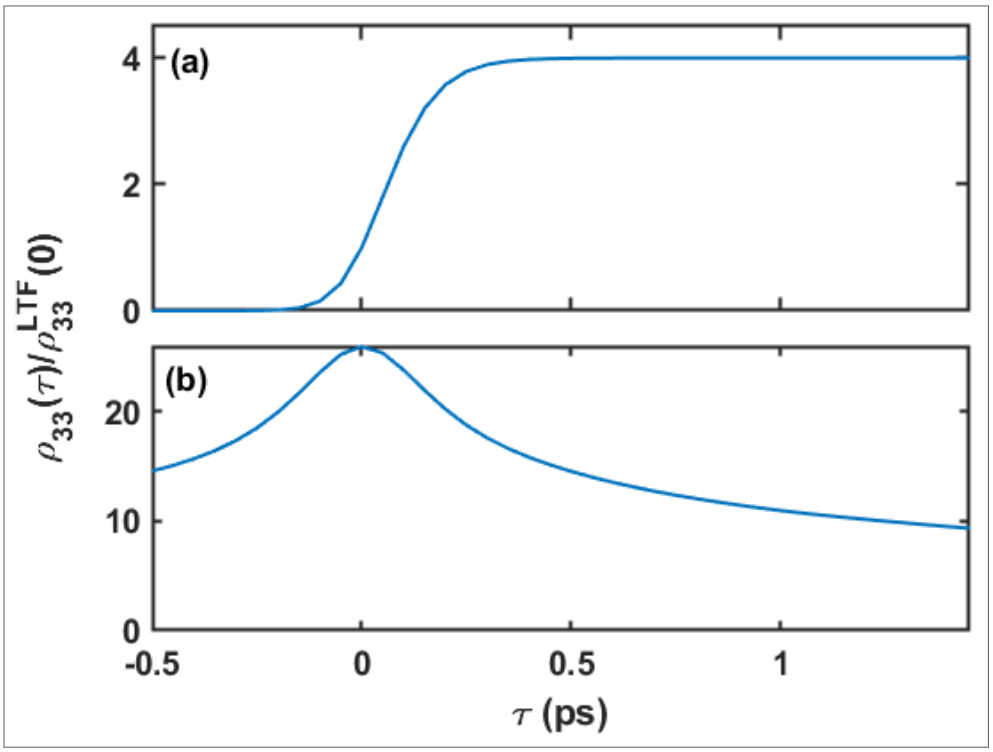
Entretanto, a forma da curva na Figura 28 depende das características do pulso de controle

(25), como demonstramos em (b). Ao aplicarmos a MF da Figura 27 (a), a varredura mostra que não apenas o valor da absorção é aumentado, mas também a forma da curva é alterada. A normalização é feita em relação a situação com $\tau = 0$ da Figura 28 (a). Os dois gráficos apontam à necessidade de investigarmos como a MF se relaciona com o atraso entre os pulsos para a otimização da transição. A fim de realizar essa investigação fizemos uma varredura no atraso, entre $[-0.5, 1.5[$ ps, em passos de 0.05 ps, em cada etapa da rotina do AG, para testar a compatibilidade de cada MF com várias condições de atraso, armazenando a informação do valor ótimo de atraso para aquela MF específica.

Testando os vários valores de atraso ao longo das simulações, obtivemos os resultados da Figura 29. Em (a), note que a simples inclusão do τ teve um grande impacto na MF, em azul sólido. Ao invés de uma MF com aparência de degrau, agora temos uma estrutura periódica. Essa característica periódica vem diretamente do termo $e^{-i\omega'\tau}$, na integral. Note que o período na MF é de ~ 3 em 128 rad/ps. Fazendo $2\pi \times 3/128$ ps obtemos ~ 0.15 ps, que é o valor do atraso utilizado no pulso de controle (o fator de 2π é necessário pois a MF está escrita em termos da frequência angular). Ou seja, com o propósito de otimizar a absorção, o AG buscou compensar qualquer diferença de fase gerada pelo atraso entre os pulsos. Embora seja mais imediato aumentar a absorção através do aumento de τ, como podemos ver na Figura 28, essa é uma abordagem limitada. O fator de aumento na absorção não é maior do que 4. Enquanto isso, numa situação sem diferença de atraso entre os pulsos, o AG é capaz de alcançar um aumento de ~ 26 vezes. Então, para atingir fatores multiplicativos mais altos, o AG lida com a fase imposta por τ simplesmente anulando o seu efeito.

Ainda na Figura 29 (a), a fase inserida por $e^{-i\omega\tau}$ não levou ao desaparecimento do salto de fase imposto pela singularidade. O salto de fase continua exatamente no centro da MF, como havíamos comentando na Secção 7.1. Ou seja, os efeitos sobre a fase espectral estão funcionando como uma simples superposição linear. Essa pode ser uma vantagem considerável para tratar de problemas mais complexos. Em (b) vemos que a forma do pulso, em azul sólido, não sofreu mudanças qualitativas em relação ao visto na Figura 27; apenas está deslocada para a direita em função da compensação do atraso de 0.15 ps no pulso de controle. Quanto ao aumento da absorção na Figura 29, a inclusão de τ não introduziu nada novo, mantendo o aumento de ~ 26 vezes, como no caso sem atraso. Isso confirma que, de fato, o AG precisou anular o efeito do atraso a fim de atingir o mesmo nível de melhoria que foi apresentada na Figura 27. Essa informação é particularmente útil para aplicações experimentais, visto que ajustes de atraso são mais trabalhosos. Utilizar o MEL em conjunto com o AG dispensa a

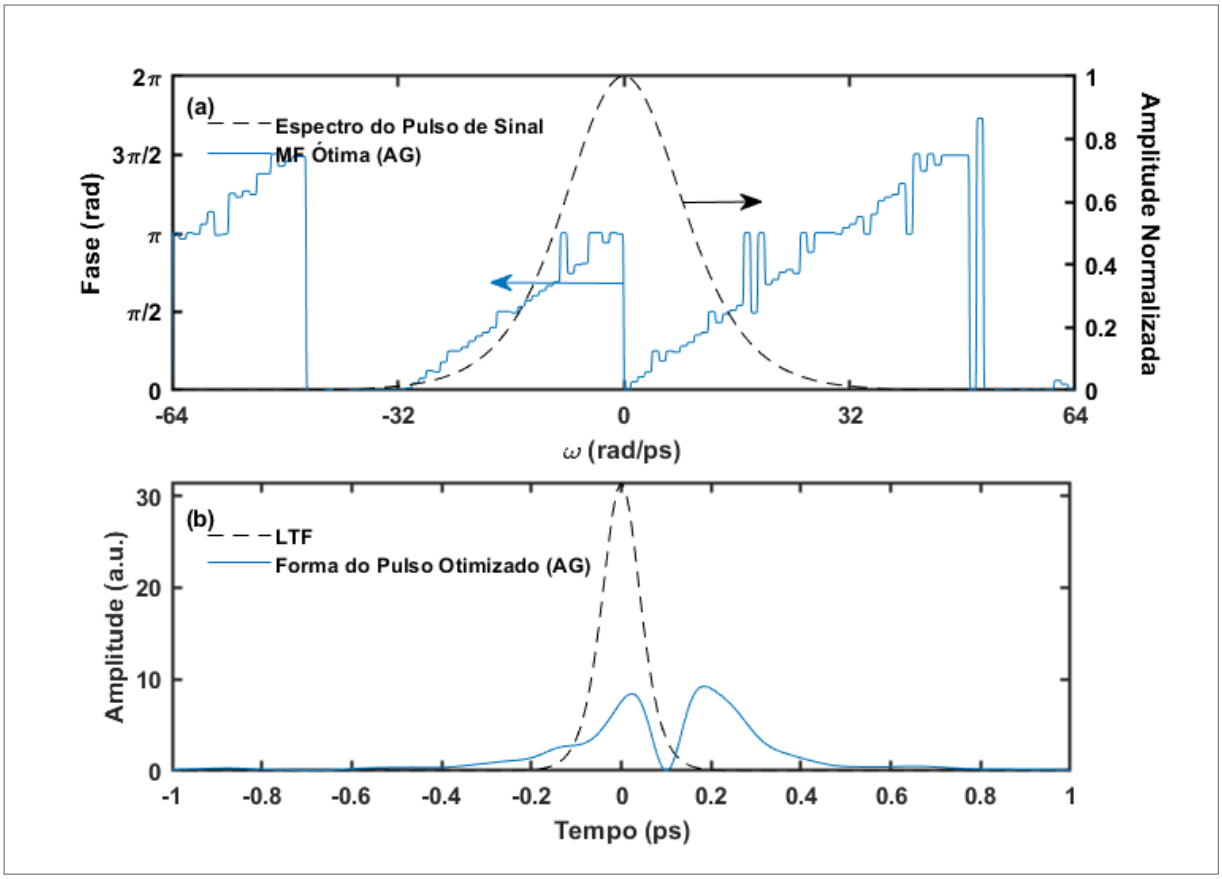
Figura 28 – Dependência da população do estado excitado, ρ_{33}, com o atraso entre os lasers, τ. (a) Curva utilizando pulsos LTF tanto no controle quanto no sinal. (b) Curva utilizando um pulso de sinal modificado pela MF da Figura 27 (a). Note como não apenas os valores mudam, mas também a forma da curva é alterada pela mudança na fase espectral. A normalização é feita em relação a ρ_{33} com $\tau = 0$ e dois pulsos LTF



Fonte: Elaborado pelo autor (2022)

necessidade de ajustes de atraso, facilitando a aplicação experimental da nossa abordagem (dentro da capacidade de ajuste temporal do MEL).

Figura 29 – Resultados das otimização do AG a partir da Equação 7.2. (a) Melhor máscara de fase, em azul sólido, obtida nas simulações após 5000 iterações. Em preto tracejado está o espectro do pulso. (b) Forma temporal do pulso após aplicação da MF mostrada em (a), em azul sólido. O pulso LTF está em preto tracejado.



Fonte: Elaborado pelo autor (2022)

7.3 OTIMIZAÇÃO PARA PULSOS DE ÁREA ZERO

O problema abordado por Carvalho et al., em (25), não lida com apenas um átomo, mas com um conjunto de átomos de Rb. Ou seja, para alcançar o nosso objetivo teremos que tratar com a Densidade Ótica (DO) do meio. Como pode ser visto em (25), esse fato aumenta bastante a complexidade do problema. Ainda assim, vimos que o AG foi capaz de tratar a singularidade e a fase induzida pelo atraso, na seção anterior, como uma simples superposição linear. Um conjunto de átomos ressonantes com um laser fraco irá produzir o efeito de um pulso de área zero (ou pulso 0π). Vamos dar um passo intermediário, observando como o AG lida com as alterações induzidas por um meio ressonante e que tipo de solução será proposta para otimizar a absorção de um pulso 0π por um único átomo. Para isso, vamos primeiramente entender o que é um pulso de área zero e quais são as características relevantes para o nosso problema.

7.3.1 Características de um Pulso de Área Zero

M. D. Crisp deduziu as propriedades de um pulso 0π em 1970 (46). A mais básica das características, de onde vem o nome "pulso 0π", pode ser rapidamente verificada com um exemplo simples. Tome um sistema atômico de dois níveis, $|1\rangle$ o estado fundamental e $|2\rangle$ o estado excitado, tal que seu estado pode ser descrito como $|\Psi(t)\rangle = c_1(t)|1\rangle + c_2(t)|2\rangle$. Podemos colocar esse sistema sob a ação de um campo elétrico de forma que o potencial é dado por

$$\hat{V}(t) = -\mu \left[E(t)|1\rangle\langle 2| + h.c.\right], \tag{7.3}$$

que na representação de interação fica

$$\hat{V}_I(t) = -\mu \left[e^{-i\omega_{12}t}E(t)|1\rangle\langle 2| + h.c.\right]. \tag{7.4}$$

Escrevendo o campo na forma $E(t) = \mathcal{E}(t)cos(\omega t)$), ficamos com

$$\hat{V}_I(t) = -\mu \left[\mathcal{E}(t)cos(\omega t)e^{-i\omega_{12}t}|1\rangle\langle 2| + \mathcal{E}(t)cos(\omega t)e^{i\omega_{12}t}|2\rangle\langle 1|\right]. \tag{7.5}$$

Escrevendo $cos(\omega t)$ na forma exponencial, com $\omega = \omega_{12}$, e aplicando a aproximação da onda girante, resta apenas

$$\hat{V}_I(t) = -\mu \left[\frac{\mathcal{E}(t)}{2}|1\rangle\langle 2| + \frac{\mathcal{E}(t)}{2}|2\rangle\langle 1|\right]. \tag{7.6}$$

Com isso, podemos escrever a evolução temporal dos estados utilizando a equação de Schrödinger, que nos dá

$$i\hbar\frac{\partial}{\partial t}\mathbf{c}(t) = \mathbf{V}(t)\cdot\mathbf{c}(t). \tag{7.7}$$

$$\frac{\partial}{\partial t}\begin{bmatrix} c_1 \\ \\ c_2 \end{bmatrix} = -\frac{i}{2}\begin{bmatrix} 0 & \Omega(t) \\ \\ \Omega(t) & 0 \end{bmatrix}\begin{bmatrix} c_1 \\ \\ c_2 \end{bmatrix}, \tag{7.8}$$

onde usamos a frequência de Rabi, $\Omega(t) = \mu\mathcal{E}(t)/\hbar$. Aplicando a solução padrão para um sistema de E.D.O.s, com $|\Psi(-\infty)\rangle = |1\rangle$, supomos uma solução do tipo

$$\mathbf{c}(t) = \begin{bmatrix} a \\ b \end{bmatrix} e^{f(t)}, \tag{7.9}$$

e, resolvendo o sistema, chegamos a

$$\mathbf{c}(t) = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{\frac{i}{2}\int_{-\infty}^{t}\Omega(t')dt'} + \frac{1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix} e^{-\frac{i}{2}\int_{-\infty}^{t}\Omega(t')dt'}. \tag{7.10}$$

A área do pulso é definida como

$$\Theta(t) = \int_{-\infty}^{t}\Omega(t')dt', \tag{7.11}$$

com a qual finalmente podemos escrever $c_2(t)$ como

$$c_2(t) = \frac{e^{i\Theta(t)/2} - e^{-i\Theta(t)/2}}{2}, \tag{7.12}$$

$$c_2(t) = sen\left(\frac{\Theta(t)}{2}\right). \tag{7.13}$$

Com isso, a probabilidade (P) de encontrarmos o sistema no estado excitado após a passagem de um pulso $(t \to \infty)$ é dada

$$P = sen^2\left(\frac{\Theta}{2}\right). \tag{7.14}$$

Dessa forma um pulso com $\Theta = 0\pi$, chamado de pulso de área zero (ou pulso 0π), é um pulso que não deixa nenhuma excitação no meio $(P = 0)$.

Porém, o resultado de nosso maior interesse é a forma espectral do campo após uma amostra ressonante de comprimento l. Podemos verificar a propagação do pulso $E(z,t)$ na direção z utilizando a equação da onda

$$\frac{\partial^2}{\partial z^2}E(z,t) - \frac{\eta^2}{c^2}\frac{\partial^2}{\partial t^2}E(z,t) = \frac{4\pi}{c^2}\frac{\partial^2}{\partial t^2}P(z,t), \tag{7.15}$$

onde $P(z,t)$ é a polarização induzida no meio atômico pelo próprio $E(z,t)$, η é o índice de refração do meio e c é a velocidade da luz no vácuo. Se consideramos um campo $E(z,t) = \mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}$, a derivada espacial fica

$$\frac{\partial^2}{\partial z^2}E(z,t) = \frac{\partial^2\mathcal{E}(z,t)}{\partial z^2}e^{i\omega(t-\eta z/c)} - 2i\frac{\eta\omega}{c}\frac{\partial\mathcal{E}(z,t)}{\partial z}e^{i\omega(t-\eta z/c)} - \frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}, \tag{7.16}$$

e a derivada temporal

$$\frac{\partial^2}{\partial t^2}E(z,t) = \frac{\partial^2 \mathcal{E}(z,t)}{\partial t^2}e^{i\omega(t-\eta z/c)} + 2i\omega\frac{\partial \mathcal{E}(z,t)}{\partial t}e^{i\omega(t-\eta z/c)} - \omega^2\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}. \quad (7.17)$$

Podemos considerar uma envoltória que varia lentamente em relação à portadora, tal que

$$\frac{\partial^2 \mathcal{E}(z,t)}{\partial z^2} \approx 0, \quad (7.18)$$

$$\frac{\partial^2 \mathcal{E}(z,t)}{\partial t^2} \approx 0, \quad (7.19)$$

nos deixando com

$$\frac{\partial^2}{\partial z^2}E(z,t) \approx -2i\frac{\eta\omega}{c}\frac{\partial \mathcal{E}(z,t)}{\partial z}e^{i\omega(t-\eta z/c)} - \frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}, \quad (7.20)$$

$$\frac{\partial^2}{\partial t^2}E(z,t) \approx 2i\omega\frac{\partial \mathcal{E}(z,t)}{\partial t}e^{i\omega(t-\eta z/c)} - \omega^2\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}. \quad (7.21)$$

A polarização do meio é induzida pelo próprio campo, de forma que podemos escrever $P(z,t) = f(\omega)E(z,t)$. $f(\omega)$ é uma quantidade relacionada à forma de linha atômica. Colocando tudo isso na equação da onda, temos

$$\left[-2i\frac{\eta\omega}{c}\frac{\partial \mathcal{E}(z,t)}{\partial z}e^{i\omega(t-\eta z/c)} - \frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}\right] - \frac{\eta^2}{c^2}\left[2i\omega\frac{\partial \mathcal{E}(z,t)}{\partial t}e^{i\omega(t-\eta z/c)} - \omega^2\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}\right] = \frac{4\pi}{c^2}\frac{\partial^2}{\partial t^2}\left[f(\omega)E(z,t)\right], \quad (7.22)$$

$$-2i\frac{\eta\omega}{c}\frac{\partial \mathcal{E}(z,t)}{\partial z}e^{i\omega(t-\eta z/c)} - \frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)} - 2i\frac{\eta^2\omega}{c^2}\frac{\partial \mathcal{E}(z,t)}{\partial t}e^{i\omega(t-\eta z/c)} + \frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)} = \frac{4\pi}{c^2}f(\omega)\frac{\partial^2}{\partial t^2}E(z,t), \quad (7.23)$$

$$-2i\frac{\eta\omega}{c}\frac{\partial \mathcal{E}(z,t)}{\partial z}e^{i\omega(t-\eta z/c)} - 2i\frac{\eta^2\omega}{c^2}\frac{\partial \mathcal{E}(z,t)}{\partial t}e^{i\omega(t-\eta z/c)} = \frac{4\pi}{c^2}f(\omega)\left[2i\omega\frac{\partial \mathcal{E}(z,t)}{\partial t}e^{i\omega(t-\eta z/c)} - \omega^2\mathcal{E}(z,t)e^{i\omega(t-\eta z/c)}\right], \quad (7.24)$$

$$-2i\frac{\eta\omega}{c}\frac{\partial\mathcal{E}(z,t)}{\partial z}-2i\frac{\eta^2\omega}{c^2}\frac{\partial\mathcal{E}(z,t)}{\partial t}=\frac{4\pi}{c^2}f(\omega)\left[2i\omega\frac{\partial\mathcal{E}(z,t)}{\partial t}-\omega^2\mathcal{E}(z,t)\right]. \tag{7.25}$$

Passando para o espaço de frequências através da transformada de Fourier,

$$-2i\frac{\eta\omega}{c}\frac{\partial\mathcal{E}(z,\omega)}{\partial z}+2\frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,\omega)=\frac{4\pi}{c^2}f(\omega)\left[-2\omega^2\mathcal{E}(z,\omega)-\omega^2\mathcal{E}(z,\omega)\right], \tag{7.26}$$

$$\frac{\eta\omega}{c}\frac{\partial\mathcal{E}(z,\omega)}{\partial z}=-i\frac{\eta^2\omega^2}{c^2}\mathcal{E}(z,\omega)-i\frac{2\pi\omega^2}{c^2}f(\omega)\left[3\mathcal{E}(z,\omega)\right], \tag{7.27}$$

$$\frac{\partial\mathcal{E}(z,\omega)}{\partial z}=-i\left[\frac{\eta\omega}{c}+\frac{6\pi\omega}{\eta c}f(\omega)\right]\mathcal{E}(z,\omega), \tag{7.28}$$

cuja solução é

$$\tilde{\mathcal{E}}(z,\omega)=\tilde{\mathcal{E}}(0,\omega)e^{-i\eta\omega z/c}e^{-i6\pi\omega f(\omega)z/\eta c}. \tag{7.29}$$

Considerando que o meio atômico tem uma forma de linha Lorentziana, podemos escrever $i6\pi\omega f(\omega)/\eta c=\alpha_0/[1-i(\omega-\omega_a)T_2]$, de forma que $\alpha_0 z$ é a DO do meio, ω_a a frequência de ressonância e T_2 é o tempo de meia vida do estado excitado. Ao longo do nosso trabalho, a DO será a quantidade relevante para as simulações.

Já o termo $e^{-i\eta\omega z/c}$ é apenas uma fase relacionada à propagação do pulso. Ou seja, em um meio sem polarização (com $P(z,t)=0$) a Equação da Onda nos dá

$$\tilde{\mathcal{E}}(z,\omega)=\tilde{\mathcal{E}}(0,\omega)e^{-i\eta\omega z/c}. \tag{7.30}$$

Com isso, a mudança na forma espectral do pulso após uma amostra ressonante de comprimento l pode ser escrita de forma resumida como

$$\tilde{E}(l,\omega)=\tilde{E}(0,\omega)e^{-\alpha_0 l/[1-i(\omega-\omega_a)T_2]}. \tag{7.31}$$

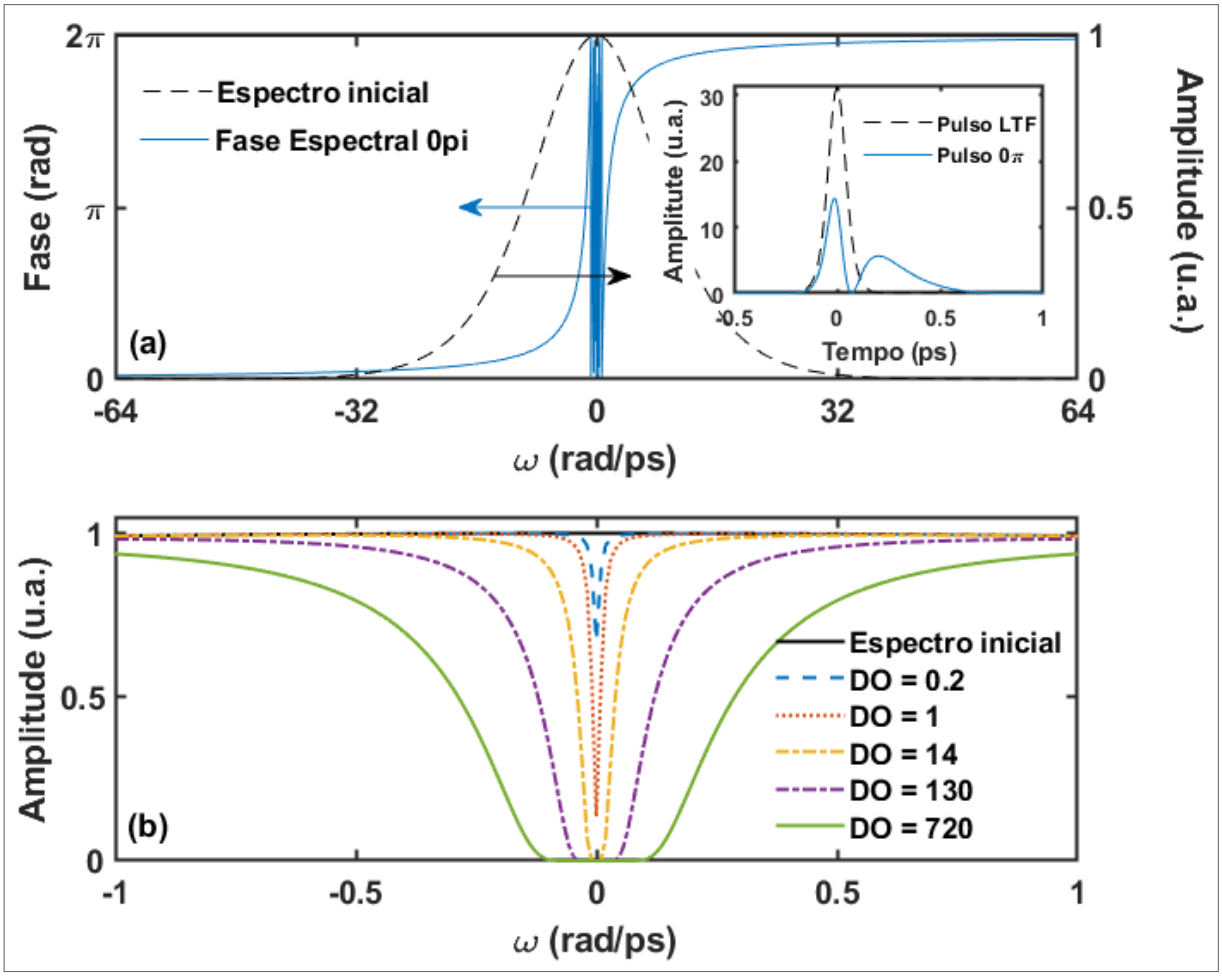
Separando a parte real e imaginária da exponencial, ficamos com

$$\tilde{E}(l,\omega)=\tilde{E}(0,\omega)e^{-\alpha_0 l/[1+(\omega-\omega_a)^2T_2^2]}e^{-i\alpha_0 l(\omega-\omega_a)T_2/[1+(\omega-\omega_a)^2T_2^2]}. \tag{7.32}$$

A Equação 7.32 revela que um pulso sofre grandes alterações após passar por um meio ressonante. A exponencial real descreve quanta energia será removida do espectro após a

passagem e a exponencial imaginária descreve como a fase espectral do pulso será modificada. A Figura 30 traz alguns exemplos do resultado dessas alterações. (a) Mostra como a fase é alterada em cada ponto do espectro para um meio com DO igual a 720 e $\omega_a = 0$. Essa curva é descrita de acordo com a exponencial imaginária da Equação 7.32. Note a variação rápida na região central, cujas oscilações excedem bastante a resolução espectral do MEL que iremos utilizar. A figura inserida mostra a forma do pulso para essas mesmas condições. Note como uma parte do pulso é atrasada e como o pico principal é bastante reduzido em comparação com o pulso LTF.

Figura 30 – (a) Fase espectral de um pulso 0π, formado após a passagem por um meio atômico ressonante com DO = 720, descrita pela parte exponencial imaginária da Equação 7.32. A figura inserida trás a forma temporal do mesmo pulso para ser comparada com um pulso LTF. (b) Perdas causadas por meios ressonantes, descritas pela exponencial real da Equação 7.32, com diferentes DO que podem ser comparadas com o pulso LTF inalterado. Para esses exemplos, foi utilizado $\omega_a = 0$



Fonte: Elaborado pelo autor (2022)

Em (b) vemos o efeito da exponencial real da Equação 7.32 no espectro do pulso para diferentes valores de DO. ω_a é mantido igual a zero pois ele produz apenas uma translação dos mesmos efeitos, que são melhores visualizados em torno do centro do espectro do pulso. Aqui

podemos ver como parte da energia é retirada do pulso após a passagem pelo meio ressonante. Note que a energia é removida principalmente da região em torno da ressonância, mas apenas em um intervalo muito curto de frequências pois o efeito exponencial decai com $1/\omega^2$, diferente da exponencial imaginária que decai com $1/\omega$, de acordo com a Equação 7.32. Essa extração de energia será importante para o AG pois essa é a região na qual as alternações de fase são capazes de oferecer maior ganho, como discutimos no Capítulo 4. Vemos então que o AG precisa lidar com três desafios simultaneamente ao tratar com pulsos 0π: mudanças rápidas na fase espectral, deformações temporais e perdas de energia na frequência de ressonância. Vamos agora ver como o AG se comporta diante dessas novas circunstâncias.

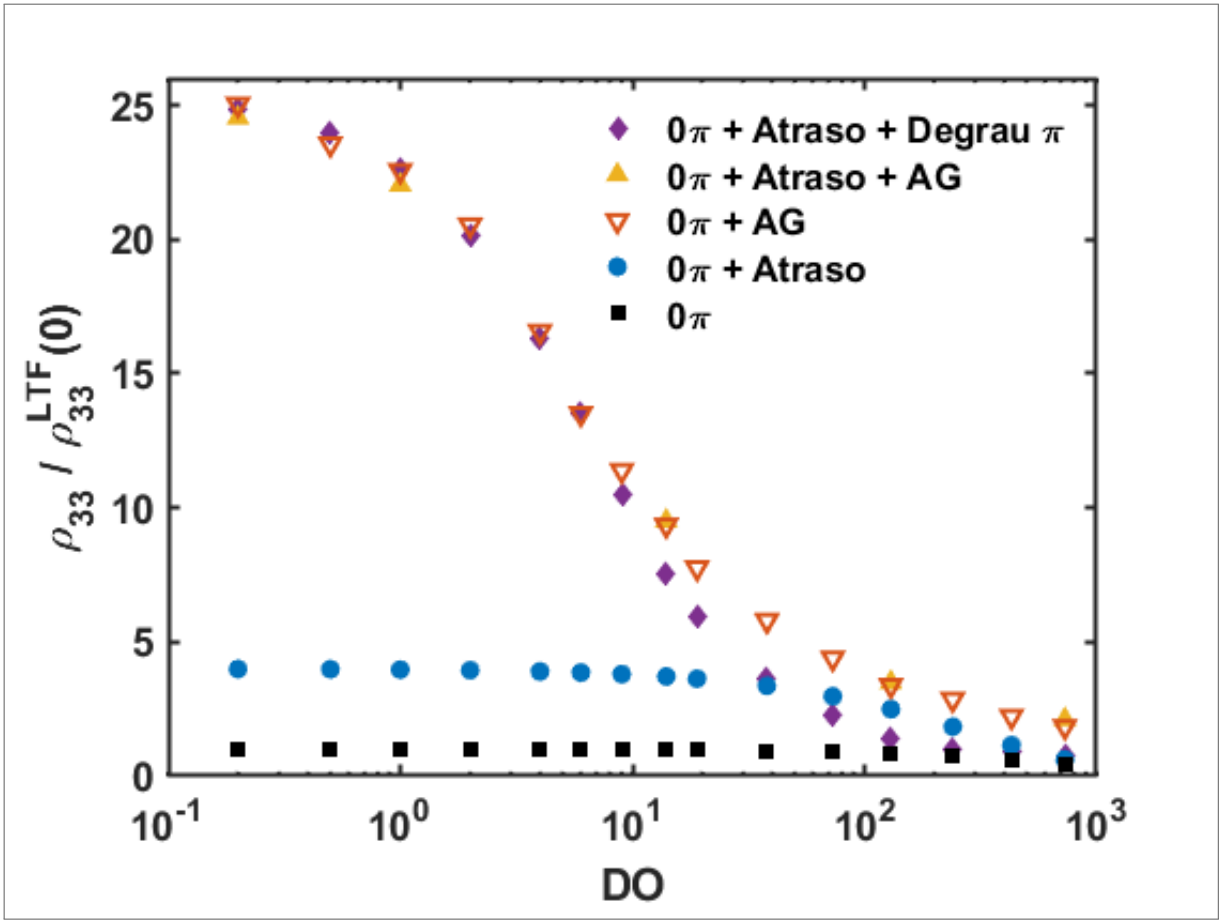
7.3.2 Otimização para Pulsos de Área Zero

No experimento realizado por Carvalho et al., o laser de controle sofria alterações desprezíveis (25). Entretanto, o pulso de sinal estava sujeito a interações proporcionalmente mais intensas. A formação de um pulso de sinal 0π era uma consequência nas equações que modelaram o experimento. Sendo assim, na nossa simulação iremos aplicar essas alterações apenas ao pulso de sinal. Trazemos à lembrança que nesta seção estamos trabalhando apenas uma versão simplificada do problema. Iremos aplicar as alterações de pulso 0π no pulso de sinal e então aplicaremos a MF gerada pelo AG neste mesmo pulso, deixando o pulso de controle inalterado. Só depois iremos considerar ambos os pulsos e realizar a integração que modela a ADF. Isso quer dizer que estamos simulando um único átomo sob condições que poderemos controlar. Uma das condições sob nosso controle, por exemplo, é o atraso entre os pulsos, que não poderá ser controlada para todos os átomos quando formos simular um meio denso, no qual teremos que lidar com a deriva (*walk-off*) temporal. Isso criará condições subótimas para a maioria dos átomos, mas só será abordado no próximo capítulo. Por ora, vamos estudar o caso mais simples.

A Figura 31 traz uma série de pontos gerados em circunstâncias diferentes, para comparação, todos normalizados pela absorção de dois lasers LTF. Os quadrados pretos mostram como a ADF é impactada quando mudamos o pulso de sinal para um pulso 0π. Note que a transição piora quanto maior for a DO que gerou o pulso 0π. Isso não é uma surpresa pois, como mostramos da seção anterior, o pulso 0π oriundo de um meio com maior DO tem cada vez menos energia a ceder para o átomo. Passando para os círculos azuis, temos uma melhora de 4 vezes devido ao ajuste do atraso (que também pode ser entendido como uma fase linear,

como vimos antes) no pulso 0π. Assim como o pulso LTF, o pulso 0π apresenta a mesma característica em relação ao atraso. Isso também pode ser compreendido espectralmente pois a fase linear adicionada pelo atraso alivia interferência destrutiva, melhorando a absorção. Porém, vemos que o atraso se torna menos eficiente quanto maior for a DO, chegando a ficar muito próximo do pulso 0π sem atraso. Isso mostra o comportamento puro de um pulso 0π.

Figura 31 – Dependência da população do estado excitado, ρ_{33}, com o pulso 0π gerado por diferentes DO. Os quadrados pretos foram gerados apenas com o pulso 0π não modificado e $\tau = 0$. Os círculos azuis utilizam o atraso para melhorar a absorção do pulso 0π. Os triângulos vermelhos vazados são o resultado da aplicação do AG ao pulso 0π, com $\tau = 0$. Os triângulos amarelos invertidos preenchidos são o resultado da aplicação do AG com a varredura no atraso entre o sinal 0π e o controle. Os diamantes roxos são o que se obtém ao aplicar uma MF com um degrau de π em torno da ressonância no sinal 0π e fazer uma varredura no atraso do controle. A normalização é feita em relação a ρ_{33} com $\tau = 0$ e dois pulsos LTF



Fonte: Elaborado pelo autor (2022)

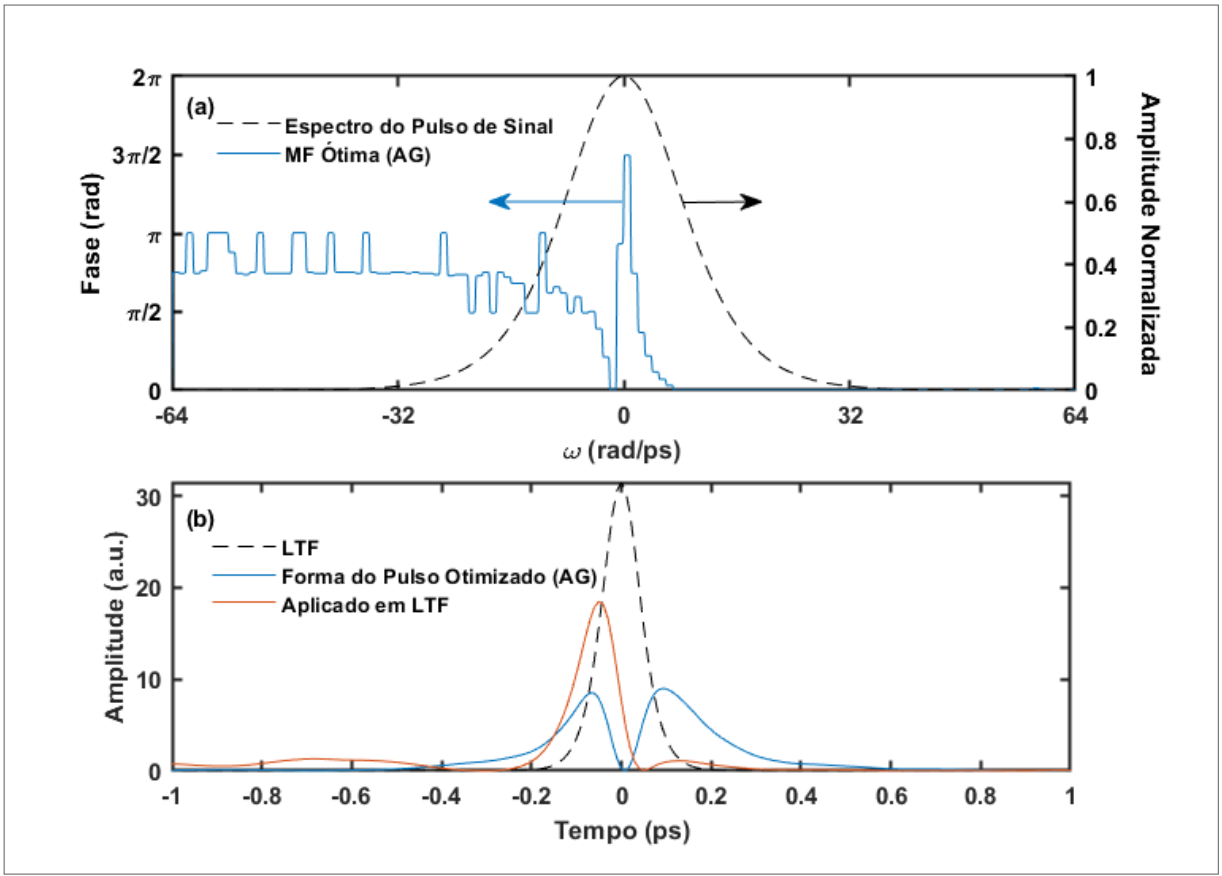
Ao adicionarmos o AG, triângulos vermelhos vazados, vemos um grande aumento na absorção. Para baixas DO, o aumento fica por volta de 25 vezes, como vimos antes (sem o pulso 0π). Com o aumento da DO, a eficiência do AG cai, mas mantém uma melhoria acima de 5 vezes até DO próximas de 100, quando a melhoria do AG se torna próxima do aumento dado puramente pelo atraso, em pontos azuis. Ainda assim os últimos pontos revelam que,

ao continuar aumentando a DO, o AG volta a ser consideravelmente mais eficiente do que o atraso. Isso ocorre pois o AG sempre está buscando a situação ótima, enquanto uma combinação eficiente da fase do pulso 0π com a fase linear do atraso é determinada puramente pela coincidência. Incluímos cinco pontos para reforçar a ideia de que o MEL com o AG dispensam a necessidade do atraso. Os triângulos amarelos invertidos coincidem com os triângulos vermelhos vazados, confirmando que mesmo o efeito de pulso 0π não afeta nossa conclusão. Dispensamos, então, qualquer investigação a mais com o atraso a fim de poupar tempo.

Ainda na Figura 31, incluímos os resultados com uma MF predefinida de salto de π em torno da ressonância, em diamantes roxos. Note como a resposta predefinida é equivalente ao resultado do AG para baixos valores de DO, onde as alterações de fase do pulso 0π são brandas. Entretanto, para valores mais altos de DO a resposta com o degrau de π perde eficiência rapidamente, chegando a se torna pior do que o atraso, em pontos azuis. A razão dessa queda brusca na eficiência é o fato de estarmos utilizando um degrau fixo, sem adaptar as particularidades do pulso 0π. Então, ao aumentar a DO, as alterações de fase do pulso 0π combinadas ao degrau de π acabam por contribuir com a interferência destrutiva, ao invés de mitigá-la. Isso mostra como o AG pode ser uma ferramenta importante ao tratar com esse tipo de sistema, sendo capaz de se adaptar a particularidades que são delicadas de serem tratadas manualmente.

Uma maneira alternativa de entender esse jogo entre a fase do pulso 0π com o AG é observando a Figura 32 (a), que traz a MF necessária para otimizar um pulso 0π gerado por uma DO de 720 (resultado mais à direita na Figura 31), em azul sólido. Note como a MF não mais é um simples degrau de π em torno da ressonância. O degrau está agora entre π e $\pi/2$ e não tem uma transição abrupta em torno da ressonância, indo de forma mais suave até atingir o zero. Essa MF pode ser comparada com a fase mostrada na Figura 30 (a), onde mostramos a fase imposta a um pulso que se propagou por um meio com DO de 720. Note como o AG buscou manter a soma da MF com a fase do pulso 0π equivalente a um degrau de π. Esse fato pode ser visualmente confirmado pelo resultado da forma temporal do pulso 0π modificado pelo AG, na Figura 32 (b), em azul sólido. Perceba que a forma final do pulso ficou semelhante ao que obtivemos na Figura 27, apesar da diferença entre as MF. Para deixar mais claro como a MF ótima para o pulso 0π é distinta da MF ótima para um pulso LTF, trouxemos a curva em laranja. Essa é uma aplicação da MF mostrada em (a) num pulso LTF, deixando clara a distinção entre os dois casos. Essa é a a razão pela qual um degrau de π não é um ajuste de fase suficiente para lidar com pulsos deformados por meios ressonantes.

Figura 32 – (a) MF gerada pelo AG para otimização da absorção do pulso 0π gerado por uma DO de 720, em azul sólido. A linha preta tracejada é o espectro do pulso do antes da formação do pulso 0π. (b) Em azul sólido, forma do pulso 0π após a modificação imposta pela MF na figura (a). Em laranja, a forma de um pulso LTF após a aplicação da mesma MF, para comparação.



Fonte: Elaborado pelo autor (2022)

Além disso, vemos novamente o AG tratando o problema como uma simples soma de fases, de maneira que agora podemos entender cada efeito de forma linear, nos dando uma grande vantagem na compreensão e controle do problema.

Isso encerra nossa avaliação sobre o efeito de pulsos 0π numa ASDF para um único átomo. Vimos que, mesmo considerando situações ótimas, os átomos que interagem com pulsos muito distorcidos pelo meio ressonante não são capazes de experimentar grandes aumentos na absorção. Além disso, vimos que a otimização de fase necessária para cada região de átomos é diferente. Átomos no início da amostra podem ser suficientemente otimizados com uma MF degrau de π, enquanto os átomos no final da amostra necessitam de ajustem mais elaborados. Por fim, até aqui fomos capazes tratar toda a problemática de fase espectral como uma simples superposição linear. No capítulo seguinte, iremos introduzir um meio atômico denso, no qual o AG precisará lidar com todos os átomos simultaneamente e compensar novos efeitos, como a deriva (*walk-off*) temporal imposto pelo meio.

8 OTIMIZAÇÃO DE FASE ESPECTRAL PARA ABSORÇÃO POR UM MEIO ATÔMICO DENSO

Até agora nós buscamos otimizar a ADF no sistema da Figura 26 sob diferentes circunstâncias, mas sempre considerando um único átomo. Neste capítulo iremos ver como o AG é capaz de melhorar a transição quando temos um conjunto desses mesmos átomos. Porém, para manter uma abordagem comparável com a que foi utilizada por Carvalho et al. (25), também iremos modificar a perspectiva do observável a ser otimizado. Ao invés de mantermos nosso olhar na população do estado excitado, vamos focar na energia que foi removida do pulso de sinal após a amostra de átomos. Essencialmente, estamos observando o mesmo efeito, apenas modificamos o ponto de observação. Pensando numa situação experimental, estamos apenas deixando de detectar a fluorescência, que tipicamente é medida perpendicularmente ao laser de sinal, para fazer uma medida direta do pulso de sinal após a célula de vapor de átomos. Essa mudança no observável foi relevante no trabalho de Carvalho et al. pois uma das motivações era demonstrar que a medida é possível de ser realizada, permitindo assim o uso do laser de sinal para armazenamento e leitura de informação (25). Visto que nossa motivação também é a aplicação em memórias quânticas, temos interesse em manter a mesma abordagem.

8.1 MODELO

Os detalhes da dedução do modelo que iremos utilizar neste capítulo podem ser encontrados em (25) e (47). Mas aqui iremos observar as equações brevemente, compreendendo o surgimento dos termos relevantes para nossa interpretação até chegarmos ao novo observável que utilizaremos como parâmetro de qualidade no AG.

Podemos descrever a propagação do nosso pulso de sinal (fraco), $E_s(z,t)$, por um meio ressonante e denso utilizando a equação de Maxwell

$$\left(\frac{\partial}{\partial z}+\frac{1}{c}\frac{\partial}{\partial t}\right)\Omega_s(z,t)=i\alpha_0\Gamma_{21}\int_{-\infty}^{\infty}d\Delta\sigma_{12}(\Delta)f(\Delta), \tag{8.1}$$

onde $\Omega=\mu E(z,t)/\hbar$ é a frequência de Rabi associada àquele campo, α_0 é o coeficiente de absorção do meio, Γ_{21} é a taxa de decaimento de $|2\rangle\rightarrow|1\rangle$, $\sigma_{12}(\Delta)$ é a coerência atômica entre $|2\rangle$ e $|1\rangle$, na representação lenta, Δ é a a dessintonia em relação à frequência central do campo e $f(\Delta)$ é a distribuição Doppler para o átomo.

É possível determinar σ_{12} utilizando a matriz densidade, ρ, na equação de Liouville

$$\frac{d\rho}{dt} = \frac{i}{\hbar}[\rho, \hat{H}], \tag{8.2}$$

com

$$\hat{H} = \hat{H}_0 + \hat{V}(z,t); \tag{8.3}$$

$$\hat{H}_0 = E_1|1\rangle\langle 1| + E_2|2\rangle\langle 2| + E_3|3\rangle\langle 3|; \tag{8.4}$$

$$\hat{V}(z,t) = -\mu_{12}E_s(z,t)|1\rangle\langle 2| - \mu_{23}E_c(z,t-\tau)|2\rangle\langle 3| + c.c.; \tag{8.5}$$

$$E(z,t) = \mathcal{E}(t-z/c)cos[\omega(t-z/c)]. \tag{8.6}$$

$\hat{H}_0$ é o Hamiltoniano do átomo livre com energias de E_1, E_2 e E_3, associados aos níveis $|1\rangle$, $|2\rangle$ e $|3\rangle$, respectivamente. $\hat{V}(z,t)$ é o potencial induzido pelos dois lasers, $E_s(\,t)$ e $E_c(z,t-\tau)$, que são escritos como uma envoltória $\mathcal{E}(t-z/c)$ e uma onda portadora de informação $cos[\omega(t-z/c)]$, são os campos de sinal e de controle, respectivamente. τ representa o atraso entre os pulsos. Utilizando a aproximação de onda girante, é possível escrever as equações de Bloch na forma

$$\begin{aligned}
\frac{\partial\rho_{11}}{\partial t} &= i\Omega_s^*\sigma_{12} - i\Omega_s\sigma_{21} + \Gamma_{21}\rho_{22},\\
\frac{\partial\rho_{22}}{\partial t} &= i\Omega_s\sigma_{21} - i\Omega_s\sigma_{12} - i\Omega_c\sigma_{32} + i\Omega_c^*\sigma_{23} + \Gamma_{32}\rho_{33} - \Gamma_{21}\rho_{22},\\
\frac{\partial\rho_{33}}{\partial t} &= i\Omega_c\sigma_{32} - i\Omega_c^*\sigma_{23} + \Gamma_{32}\rho_{33},\\
\frac{\partial\sigma_{12}}{\partial t} &= i(\omega_{21}-\omega_s)\sigma_{12} + i\Omega_s(\rho_{11}-\rho_{22}) + i\Omega_c^*\sigma_{13} - \frac{\Gamma_{21}}{2}\sigma_{12},\\
\frac{\partial\sigma_{13}}{\partial t} &= i[\omega_{31}-(\omega_c+\omega_s)]\sigma_{13} - i\Omega_s\sigma_{23} + i\Omega_c\sigma_{12} - \frac{\Gamma_{32}+\Gamma_{21}}{2}\sigma_{13},\\
\frac{\partial\sigma_{23}}{\partial t} &= i(\omega_{32}-\omega_c)\sigma_{23} + i\Omega_c(\rho_{22}-\rho_{33}) - i\Omega_s^*\sigma_{13} - \frac{\Gamma_{32}}{2}\sigma_{23}.
\end{aligned} \tag{8.7}$$

Com isso, podemos resolver as equações de Bloch para obter uma expressão de σ_{12} usando a condição de pulso fraco e área do pulso de controle constante. Substituindo o resultado na equação 8.1, ficamos com

$$\left(\frac{\partial}{\partial z} + \frac{1}{c}\frac{\partial}{\partial t}\right)\Omega_s(z,t) = -\alpha_0\int_{-\infty}^{t} dt' G(t-t')\left[\Omega_s(z,t') - \Omega_c^*(z,t')\theta_s(z,\tau+z/c)\frac{\Theta_c}{8}\right]; \tag{8.8}$$

$$G(t-t') = \frac{\Gamma_{21}}{2}\int_{-\infty}^{\infty} d\Delta f(\Delta) e^{(\Gamma_{21}/2+i\Delta)(t-t')}, \tag{8.9}$$

sendo Θ_c a área do pulso de controle, que assumimos constante, e $\theta_s(\,\tau + z/c)$ a área do pulso de sinal. Utilizando a transformada de Fourier chegamos a

$$\left(\frac{\partial}{\partial z} - \frac{1}{c}\frac{i\omega}{c} + \alpha_0 A(\omega)\right)\tilde{\Omega}_s(z,\omega) = \alpha_0 A(\omega)\tilde{\Omega}_c^*(z,-\omega)\theta_s\left(z, \tau + \frac{z}{c}\right)\frac{\Theta_c}{8}; \tag{8.10}$$

$$A(\omega) = \frac{\Gamma_{10}}{2}\int_{-\infty}^{\infty} d\Delta \frac{f(\Delta)}{\frac{\Gamma_{12}}{2} - i(\omega-\Delta)}. \tag{8.11}$$

$\tilde{\Omega}$ representa a transformada de Fourier do respectivo pulso e ω é a frequência em relação à frequência central do pulso.

Aqui é interessante realizar uma breve pausa para observar uma das consequências da Equação 8.10, que já é capaz de demonstrar uma relação com o que apresentamos no capítulo anterior. Note que na ausência o campo de controle ($\Omega_c = 0$), temos a equação que descreve a formação do pulso 0π, apresentada no capítulo anterior. Usando o modelo deste capítulo, podemos entender que o pulso 0π é uma aproximação de ordem zero no pulso de controle para a propagação do pulso de sinal. Esse entendimento será útil mais adiante pois a área do pulso de controle se tornará um parâmetro de aproximação. Empregando a notação deste capítulo podemos escrever o pulso 0π como

$$\tilde{\Omega}_s^{(0)}(z,\omega) = \tilde{\Omega}_s(0,\omega)e^{\left[\frac{i\omega}{c} - \alpha_0 A(\omega)\right]z}. \tag{8.12}$$

Aplicaremos essa descrição nas equações finais. Perceba que essa equação está nos dizendo que quando a ação do pulso de controle for desprezível o pulso de sinal ficará sujeito apenas a deformações e perdas energéticas de um pulso 0π. Note também que é difícil evitar, dentro de um meio denso com pulsos de frequências diferentes, que o pulso de sinal viaje, mesmo que apenas em parte da amostra, separado do pulso de controle (devido à deriva temporal). Sendo assim, é esperado que em algum ponto da amostra os pulsos não estejam superpostos, o que causará mudanças de pulso 0π no pulso de sinal.

O pulso 0π também será importante para definir a normalização do novo observável. Temos que a energia de um pulso é dada por,

$$U(z) = \epsilon\int_{-\infty}^{\infty} d\omega |\tilde{\Omega}(z,\omega)|^2. \tag{8.13}$$

Então, com o pulso 0π fica

$$U^0(z) = \epsilon \int_{-\infty}^{\infty} d\omega |\tilde{\Omega}_s^{(0)}(z,\omega)|^2, \tag{8.14}$$

$$U^0(z) = \epsilon \int_{-\infty}^{\infty} d\omega |\tilde{\Omega}_s^{(0)}(0,\omega) e^{-\alpha_0 z Re(A(\omega))}|^2. \tag{8.15}$$

Essa é a energia do pulso de sinal após a interação mínima com o meio (ou seja, sem influência do pulso de controle). ϵ é uma constante de proporcionalidade. Como vimos anteriormente, a exponencial real do pulso 0π descreve a energia removida do pulso original. Aqui temos essa mesma característica evidenciada de outra maneira.

Em seu trabalho, Carvalho et al. também estava interessado em saber a energia total do pulso após deixar a amostra. Mas, na ocasião do seu experimento, a intenção era demonstrar que o pulso de controle altera a quantidade de energia absorvida pelo meio (25). Por este motivo a Equação 8.15 foi utilizada para normalizar os resultados, apontando o quanto mais absorvido era o pulso de sinal por influência do pulso de controle. Para nós, a Equação 8.15 ainda é interessante para normalizar os resultados pois o pulso de sinal não foi alterado por nenhum MEL. Então usaremos essa equação para normalizar o quanto mais o pulso de sinal é absorvido por influência das alterações de fase realizdas pelo AG em conjunto com o MEL. Dessa forma, seremos capazes de comparar diretamente os nossos resultados com os resultados obtidos por Carvalho et al. (25).

Retornando ao desenvolvimento da Equação 8.10, podemos colocá-la na forma integral após alguma manipulação (detalhes em (47)) e obter:

$$f(z,\omega) = f(0,\omega) + \int_0^z dz' \frac{\beta A(\omega)}{\sqrt{2\pi}} e^{i\omega\tau + \alpha_0 A(\omega) z'} \tilde{\Omega}_c^*(0,-\omega) \times \int_{-\infty}^{\tau} dt \int_{-\infty}^{\infty} d\omega' e^{-\alpha_0 A(\omega') z'} f(z',\omega') e^{-i\omega' t}; \tag{8.16}$$

onde definimos

$$f(z,\omega) = \tilde{\Omega}_s(\,\omega) e^{\left(-\frac{i\omega}{c} + \alpha_0 A(\omega)\right)}, \tag{8.17}$$

$$\beta = \frac{\alpha_0 \Theta_c}{8}. \tag{8.18}$$

Podemos expandir em Θ_c e a primeira ordem nos dá

$$f^{(1)}(z,\omega) = f(0,\omega) + \int_0^z dz' \frac{\alpha_0 \Theta_c A(\omega)}{8\sqrt{2\pi}} e^{i\omega\tau + \alpha_0 A(\omega) z'} \times \tilde{\Omega}_c^*(0,-\omega) \int_{-\infty}^{\tau} dt \int_{-\infty}^{\infty} d\omega' e^{-\alpha_0 A(\omega') z'} \tilde{\Omega}_s(0,\omega') e^{-i\omega' t}, \quad (8.19)$$

que nos dá a forma do pulso após a amostra.

Entretanto, no experimento, o laser de controle é focalizado no centro da amostra. Para contabilizar esse fato, a área do pulso de controle é considerada nula fora da região central da célula de vapor. Apenas dentro da região definida pelo comprimento Rayleigh em torno do centro, onde está a cintura do feixe de controle, consideramos a área do pulso de controle não nula (mas ainda constante). Note que isso tem uma implicação para o pulso de sinal. Se apenas em uma determinada sub região da amostra de átomos a área do pulso de controle é não nula, isso garante que o pulso de sinal viaja por parte da amostra de átomos sem interagir com o pulso de controle. Como discutimos alguns parágrafos atrás, isso quer dizer que o pulso de sinal irá evoluir de acordo com a Equação 8.12 em parte da amostra, ou seja, nosso modelo está nos dizendo que o pulso de sinal obrigatoriamente assume características de pulso 0π ao longo da propagação. Introduzindo essa condição experimental no modelo nos permite finalmente escrever a forma do pulso após a célula de vapor

$$\tilde{\Omega}_s^{(1)}(l,\omega) = \tilde{\Omega}_s^{(0)}(l,\omega) + \frac{\Theta_c}{8} e^{i\omega\left(\tau + \frac{l}{c}\right)} \int_{-\infty}^{\tau} dt F(\omega,t), \quad (8.20)$$

com

$$F(\omega,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} d\omega' e^{-i\omega' t} \tilde{\Omega}_s(0,\omega) \tilde{\Omega}_c^*(0,-\omega) T(\omega,\omega'), \quad (8.21)$$

e

$$T(\omega,\omega') = \frac{A(\omega) e^{D[A(\omega)+A(\omega')]/2}}{A(\omega) - A(\omega')} \left\{ e^{D_c[A(\omega)-A(\omega')]/2} - e^{-D_c[A(\omega)+A(\omega')]/2} \right\}. \quad (8.22)$$

D é a profundidade ótica do meio e Dc é a profundidade ótica da região definida pelo comprimento de Rayleigh.

Ao longo da análise de Carvalho et al., a forma de linha $A(\omega)$ também foi aproximada para

$$A(\omega) = \frac{1}{1 - i\omega T_2}. \quad (8.23)$$

Isso permitiu fazer

$$\frac{A(\omega)}{A(\omega) - A(\omega')} = -\frac{\omega' T_2 + i}{(\omega - \omega')T_2}. \tag{8.24}$$

Note que isso deixa evidente a singularidade da ADF, dentro de $F(\omega, t)$. Mas aqui temos a ressonância escrita de forma um pouco distinta do que apresentamos anteriormente. Ao invés de focar no caso RFI, como fizemos, todas as ressonâncias de dois fótons estão sendo consideradas aqui, porém moduladas pelas exponenciais da forma de linha em $T(\omega, \omega')$.

A Equação 8.20 será utilizada nas nossas simulações, com a aproximação da Equação 8.23, para descrever o pulso de sinal. Já o nosso observável será dado pela diferença de energia do pulso 0π previsto para aquela amostra e o pulso otimizado, ao final da amostra. Além disso, o resultado será normalizado pela energia do pulso 0π. Em outras palavras, nosso observável é

$$\Delta\alpha_0 = \frac{\Delta U}{U^{(0)}(l)}, \tag{8.25}$$

onde

$$\Delta U = U^{(0)}(l) - U^{(1)}(l). \tag{8.26}$$

Vemos então que problema proposto por Carvalho et al. (25) apresenta desafios semelhantes aos que trabalhamos nos capítulos anteriores. Temos o problema de interferência destrutiva e formação de pulso 0π com os quais o AG demonstrou ser capaz de lidar com sucesso, apesar de ter limitações sob certas circunstâncias. Porém, agora o AG precisará lidar a propagação de pulsos com frequências diferentes no meio atômico, o que tratá o problema do atraso entre os pulsos de uma uma maneira distinta.

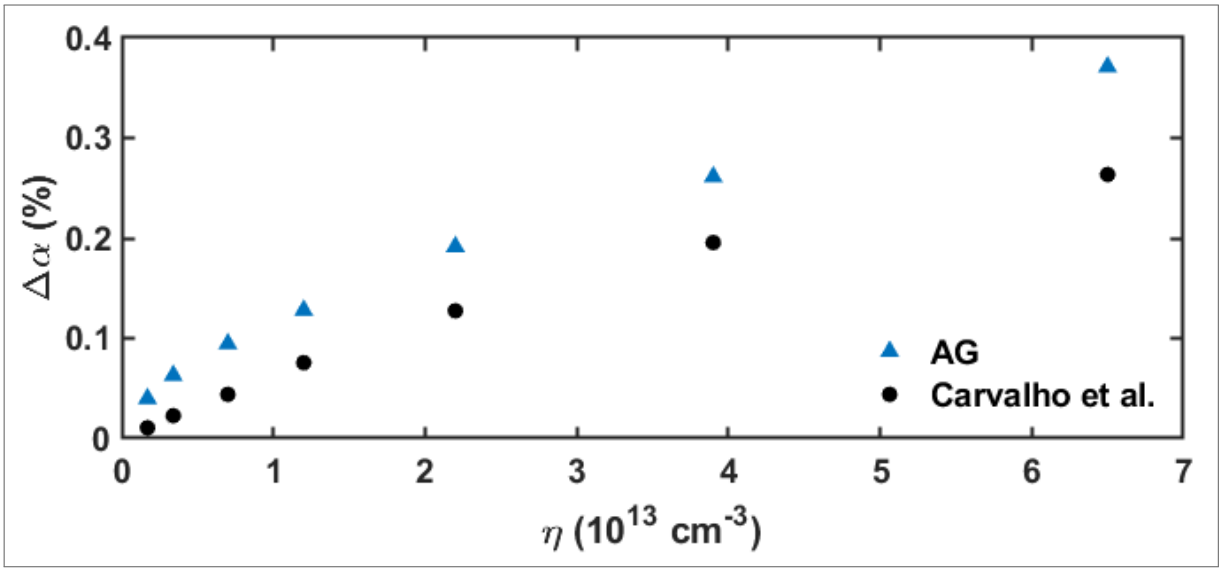
Vimos antes que para o AG o atraso entre os pulsos era relevante, mas que bastava ser compensado para atingir altos níveis de absorção (Figura 29). Vimos também que, para aquela determinada MF que otimizava a transição atômica, mudar o atraso entre os pulsos poderia alterar consideravelmente a resposta do sistema (Figura 28). Agora, com o novo modelo, a otimização de fase espectral em uma região da amostra poderá implicar numa situação desfavorável em outra região, pois a separação temporal não é fixa dentro da amostra. Vejamos como o AG lida com essas circunstâncias.

8.2 RESULTADOS

O trabalho de Carvalho et al. se concentrava em demonstrar que era possível medir como a área do pulso de controle, utilizada como parâmetro, influencia a absorção do pulso de sinal em um meio atômico denso (25). Para isso, o experimento foi realizado em diferentes condições de DO a fim de demonstrar a melhoria promovida pelo ajuste do laser de controle. Então os pontos de aumento de absorção máximos foram capturados para cada DO. Partiremos dessa condição ótima, promovida pela área do pulso de controle, e aplicaremos o nosso método para verificar se é possível realizar alterações de fase espectral no pulso de sinal para aumentar ainda mais a absorção do pulso de sinal pelo meio. O código para simulação desde capítulo pode ser encontrado no Apêndice B.

A Figura 33 traz os resultados obtidos pelo AG, em triângulos azuis, e compara com os resultados obtidos por Carvalho et al. (25), em círculos pretos. Imediatamente percebemos que o AG foi capaz de fazer o pulso de sinal ser mais absorvido pelo meio atômico. Fica evidente também que aumentar a profundidade ótica aumenta a absorção. Isso é esperado pois mais átomos implica em mais energia sendo retirada do pulso de sinal, mesmo se consideramos o caso sem o pulso de controle (Figura 30).

Figura 33 – Comparação do aumento de absorção do pulso de sinal obtido por Carvalho et al., em círculos pretos, e pelo nosso método com AG, em triângulos azuis. O eixo vertical é determinado pela Equação 8.25. Os valores de η utilizados estão relacionados com a DO da seguinte maneira: $\eta = 0.17 \times 10^{-3}$ é igual a $OD = 19$, $\eta = 0.34 \times 10^{-3}$ é igual a $OD = 38$, $\eta = 0.7 \times 10^{-3}$ é igual a $OD = 73$, $\eta = 1.2 \times 10^{-3}$ é igual a $OD = 130$, $\eta = 2.2 \times 10^{-3}$ é igual a $OD = 230$, $\eta = 3.9 \times 10^{-3}$ é igual a $OD = 430$, $\eta = 6.5 \times 10^{-3}$ é igual a $OD = 720$.



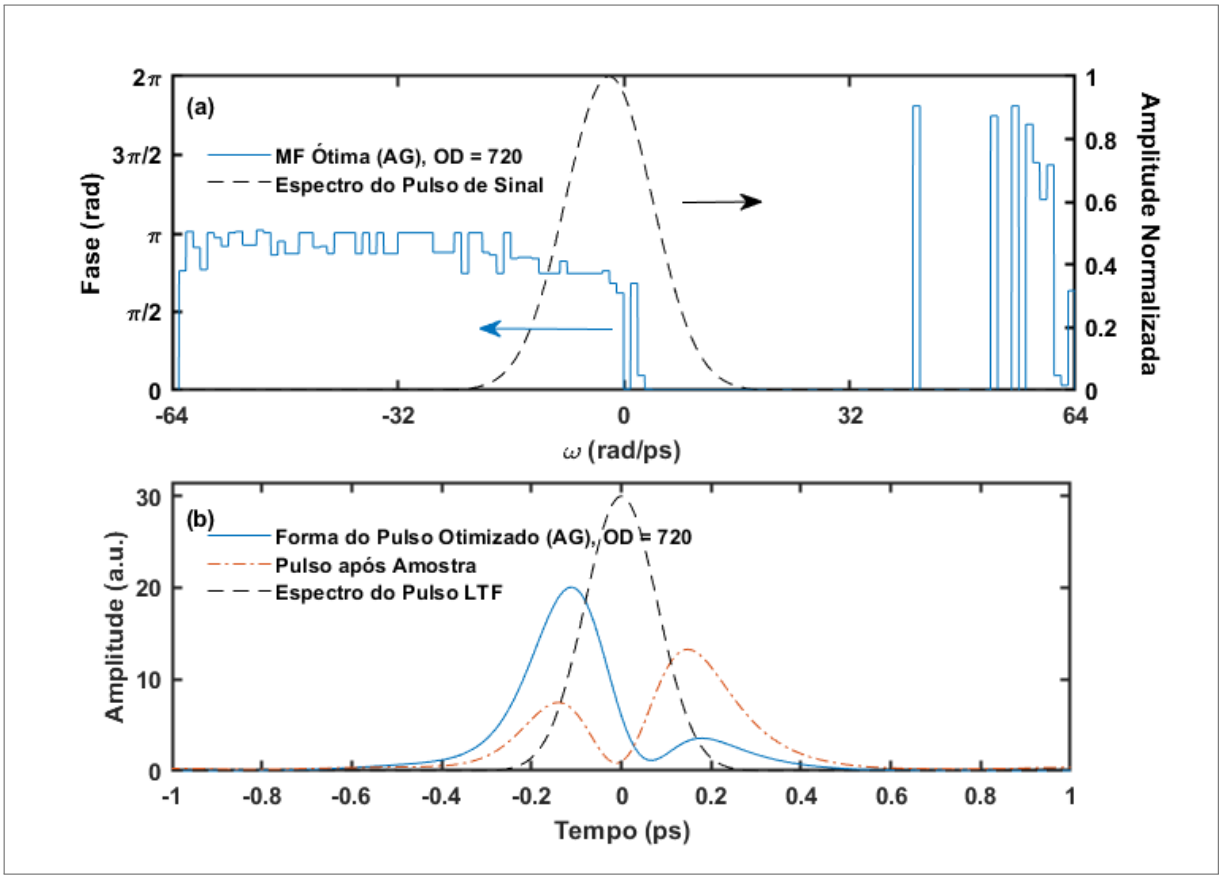
Fonte: Elaborado pelo autor (2022)

Olhando com mais atenção, também podemos perceber que, assim como no capítulo anterior, a melhoria relativa provida pelo AG foi maior para os valores de DO mais baixos. Novamente, isso está relacionado ao fato de que pulsos menos distorcidos tem maior potencial de otimização por conterem mais energia nas frequências ressonantes. O aumento do ponto com DO $=$ 19 foi de ~ 4, enquanto o do ponto com DO $=$ 720 foi de ~ 1.5. E reforçamos que, apesar da quantidade de energia total absorvida ser pequena, a diferença está dentro da ordem que o experimento executado por Carvalho et al. foi capaz de detectar (25). Ou seja, ter o pulso otimizado dá ao experimento uma maior amplitude de valores diante das alterações aplicadas ao pulso de controle, permitindo assim uma manipulação mais eficiente do experimento.

Com isso, vemos que o desenvolvimento feito nos capítulos anteriores é capaz de nos ajudar a compreender os resultados obtidos aqui. Ainda assim, é interessante ver a solução de fase espectral proposta pelo AG. A Figura 34 mostra como a solução para esse novo modelo é semelhante ao visto na Figura 32. Em (a) temos a MF que otimizou o processo para DO $=$ 730, em azul contínuo. Em preto tracejado temos o espectro do pulso de sinal. Note como a MF tem uma transição mais suave em torno da ressonância ($\omega = 0$), como na Figura 32, assim como não tem uma transição de π (apenas próximo de π). O espectro do pulso de sinal está levemente deslocado de $\omega = 0$, a frequência da transição, pois essa dessintonia estava presente no experimento de Carvalho et al. (25). Decidimos manter a dessintonia para sermos capazes de comparar os nossos resultado diretamente com o apresentados por Carvalho et al.. E, por ser pequena, a dessintonia não altera o modelo nem as soluções propostas pelo AG, apenas causando pequenas alterações na energia total disponível para absorção.

Em (b) vemos a forma do pulso de sinal após a alteração provida pelo MEL (e antes do meio denso), em azul contínuo, comparada com a forma inicial, em preto tracejado. Note que temos uma forma muito semelhante ao obtido na Figura 32 (b). Entretanto, a similaridade é com ao caso da MF aplicada ao pulso LTF (curva laranja) do capítulo anterior, pois os efeitos do meio denso ainda não foram aplicados ao pulso. O que chama a atenção é que agora, na Figura 34, nós invertemos a ordem das aplicações (primeiro MF e depois meio denso), mas isso não mudou a forma que o AG lidou com o problema. Esse fato reforça que estamos lidando com uma situação de superposição de efeitos. A ordem dos elementos não afetou a MF resultante. Além disso, não estamos lidando apenas com a formação de pulso 0π, mas outros efeitos descritos pela equação 8.20 e ainda assim o AG está nos dando resultados similares ao do modelo simplificado. Falaremos mais sobre isso adiante.

Figura 34 – (a) MF resultante da aplicação do AG para o modelo 8.20 com DO = 720, em azum contínuo. Em preto tracejado, o espectro do pulso de sinal. (b) Forma do pulso otimizado, antes da amostra, em azul contínuo. Em preto tracejado, o pulso antes da modificação pelo MEL. Em vermelho alternado, o pulso após a amostra.



Fonte: Elaborado pelo autor (2022)

Uma última observação é a forma do pulso após a amostra, em vermelho traço-pontilhado. Podemos perceber que ela lembra a forma do pulso azul contínuo da Figura 32 (b), porém um pouco mais deformado. Novamente, como temos uma sequência de MEL e meio atômico, temos a forma do pulso com dois picos, dando o resultado final do ajuste do AG para o meio, como vimos anteriormente. A deformação que leva o pulso a ter um pico mais alto em tempos positivos é causada pois as alterações de fase no pulso não estão sendo ajustadas pelo AG para toda a amostra, mas sim para otimizar uma determinada região na qual o pulso de sinal interage com o pulso de controle. Assim, após a região de interação temos um pulso que apenas é deformado pelo meio, sem compensação de fase, deixando a forma do pulso de sinal após a amostra, neste capítulo, levemente distinta do que encontramos na Figura 32 (b).

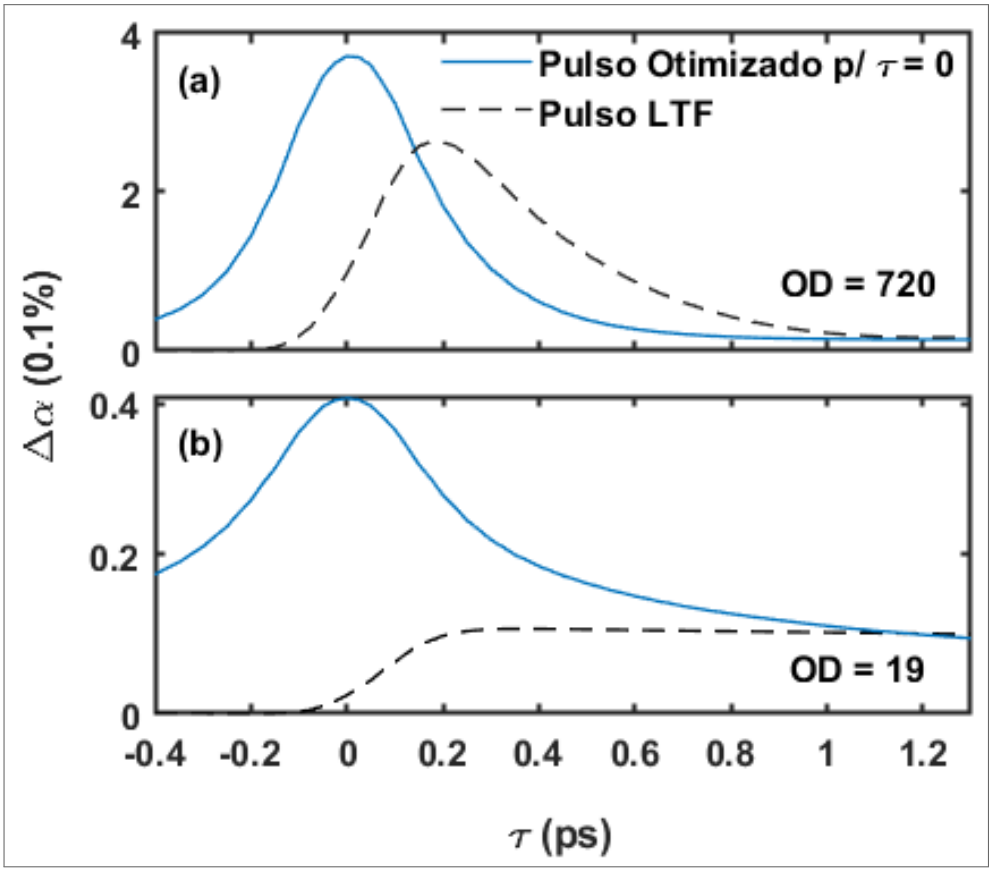
Outro traço de semelhança entre o que observamos neste capítulo e os anteriores é o desempenho da MF gerada pelo AG sob variação do atraso entre os pulsos. A Figura 35 mostra este comportamento para as duas extremidades da Figura 33. Inicialmente percebemos que o

comportamento mostrado é semelhante ao mostrado na Figura 28, especialmente em (b). A MF promove um aumento na absorção, mas também altera a relação com o atraso. Entretanto, vemos que para DO $= 720$, em (a), mesmo a curva do pulso LTF já apresenta um máximo. Visto que a amostra de átomos impõe mudanças de fase no pulso de sinal, é natural que exista uma situação ótima de atraso. Mas o aspecto mais interessante é notar que, independente da profundidade ótica, é mais eficiente que o pulso de controle chegue depois do pulso de atraso quando utilizamos pulsos LTF, pois isso é alterado quando utilizamos o AG. As curvas azuis mostram que o AG não apenas otimiza a absorção para $\tau = 0$ como ele também força que essa seja a melhor situação de atraso entre os pulsos. Isso reforça como o AG combinado com o MEL dispensam qualquer intervenção extra de atraso temporal, como temos mostrado recorrentemente. Também forçamos o AG a trabalhar com diferentes situações fixas de atraso, mas a otimização de absorção não foi alterada. Apenas temos a alteração na MF mostrada na Figura 29 (a), para compensar o atraso, e as curvas azuis são transladadas de forma que o pico esteja localizado sobre o valor escolhido de atraso. Isso define o atraso como um parâmetro irrelevante ao utilizarmos o AG e MEL, sendo uma grande vantagem experimental.

A última observação sobre a Figura 35 é sobre o que já foi comentado da melhora relativa que o AG pode proporcionar. Note que a diferença entre o pico da curva preta tracejada e a curva azul contínua é diferente para cada valor de DO. Embora a visualização fosse mais difícil na Figura 33, aqui vemos claramente que a melhoria relativa provida pelo AG para DO $= 19$ foi maior do que para DO $= 720$. Isso reafirma o comportamento identificado na Figura 31, que as otimizações de fase espectral realizadas pelo AG se tornam menos relevantes quanto mais o pulso otimizado foi alterado por um meio ressonante. Mas aqui vemos um aspecto ambivalente, pois ao mesmo tempo que a otimização perde relevância, o aumento na DO torna a absorção mais eficiente. Dessa forma, uma pessoa interessada em aplicar o experimento poderá escolher entre ter um pulso de sinal que é melhor absorvido por ter um meio atômico mais denso, colocando a otimização em segundo plano, ou um meio atômico com menor capacidade de absorção, porém mais susceptível às alterações de fase espectrais.

Podemos usar nossa suposição de que as alterações de fase espectral no pulso de sinal são uma superposição linear para tentar prever como seria a fase do pulso de sinal na saída da amostra atômica. A Figura 36 (a) trás a soma da MF gerada pelo AG, mostrada na Figura 34, com a fase de um pulso 0π gerado por um meio com DO $= 720$, mostrada na Figura 32. A soma está em azul contínuo e o espectro do pulso de sinal está em preto tracejado, para referência. Na Figura 36 (b) temos a fase do pulso de sinal otimizado após a amostra

Figura 35 – Resposta da MF gerada pelo AG na Figura 33 sob diferentes condições de atraso entre os pulsos, em azul sólido, comparado com os resultados obtidos por Carvalho et al., em preto tracejado, que usou pulsos LTF antes da amostra. (a) DO = 720. (b) DO = 19.



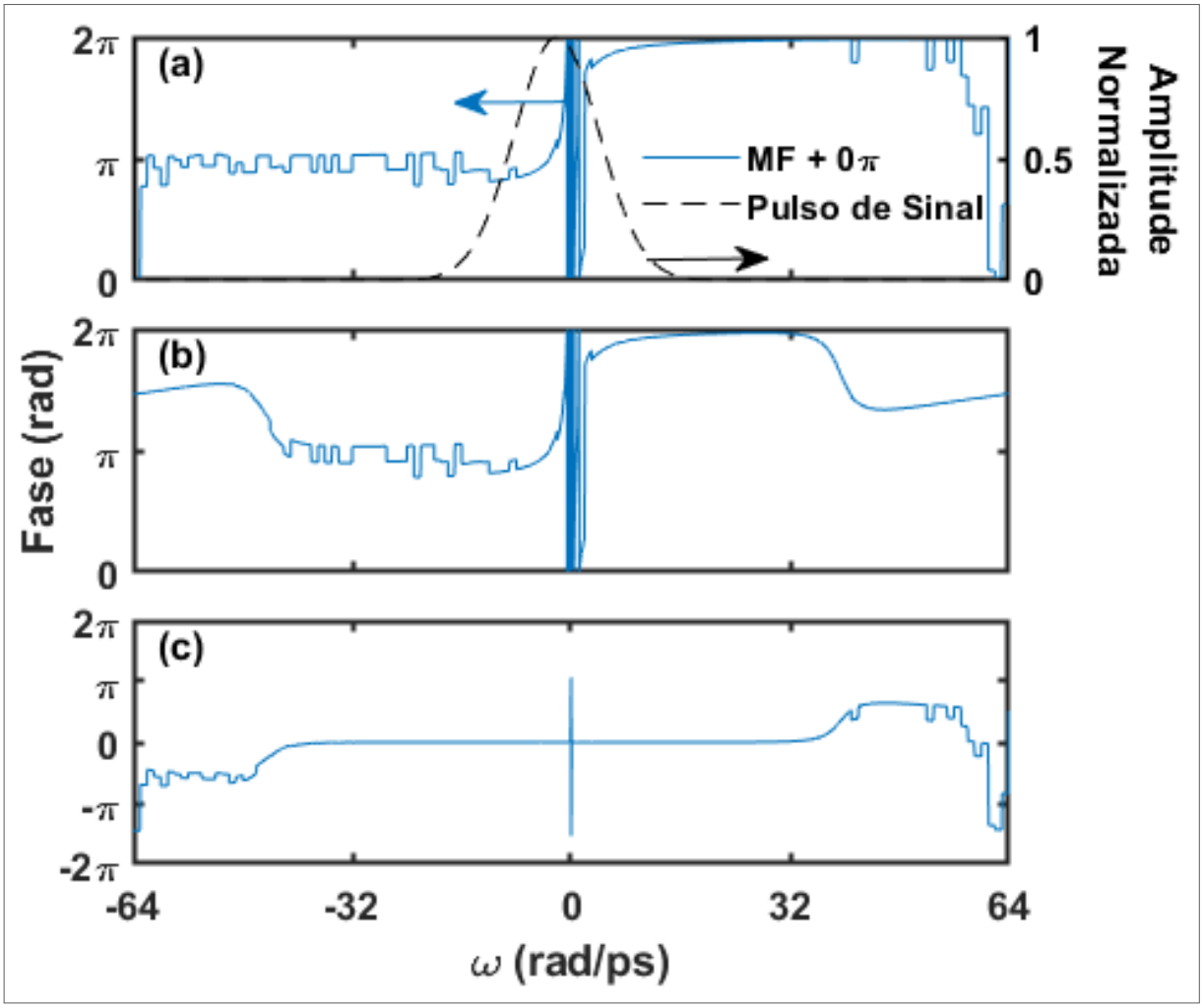
Fonte: Elaborado pelo autor (2022)

amostra atômica, utilizando o modelo da Equação 8.20. A semelhança entre o gráfico (a) e (b) na região central é visível. Embora existam diferenças longe do centro, elas são irrelevantes devido a baixa amplitude do campo.

Para deixar mais evidente o nível de similaridade entre a Figura 36 (a) e (b), temos o gráfico mostrado em (c), que mostra do resultado da subtração (a) - (b). Note que o resultado é nulo em toda a região relevante do pulso, ou seja, as únicas alterações de fase espectral foram, de fato, a modificação promovida pelo AG e o resultado da propagação pelo meio denso, que gera as características de pulso 0π. A variação que vemos em $\omega = 0$ é causada pela leitura do campo naquela região. Como as amplitudes são muito baixas, a leitura da fase se torna difícil. De qualquer maneira, a amplitude do campo em $\omega = 0$ pode ser considerada nula, como vemos na Figura 30 (b), tornando qualquer característica de fase naquela região irrelevante.

Dessa forma, vemos claramente que as mudanças de fase espectrais podem ser entendidas como uma superposição linear de efeitos. Isso pode ser de grande utilidade, pois partimos de um modelo físico complexo (Equação 8.20) e estamos sendo capazes de explicar a otimização

Figura 36 – (a) Soma da MF gerada pelo AG com a fase de um pulso 0π gerado por um meio com DO $=$ 720, em azul contínuo, e o espectro do pulso de sinal, em preto tracejado. (b) Fase do pulso de sinal após a passagem pela amostra atômica usando o modelo da Equação 8.20. (c) subtração de (a) com (b), revelando a semelhança dentro da região de interesse.



Fonte: Elaborado pelo autor (2022)

como uma simples soma de fases espectrais, abrindo a possibilidade de compreender a ADF por uma perspectiva mais simples.

Com isso, concluímos este capítulo mostrando que nosso método com o AG não apenas é capaz de proporcionar uma otimização espectral para a ASDF com um meio denso, mas que também podemos quebrar o problema em etapas de entendimento mais acessível, tornando possível compreender a otimização de fase espectral da ADF como uma simples superposição linear de efeitos.

9 CONCLUSÃO

Apresentamos uma caracterização da rota para o caos no laser de modos travados em fibra de Yb com diversas assinaturas que distinguem o caos do ruído gaussiano, típico dos sistemas experimentais. Montando as representações básicas para estudos de caos, como mapas de bifurcação e mapas de retorno, fomos capazes de visualizar experimentalmente a rota para o caos de forma completa, que nos surpreendeu com caos unimodal para o laser de modos travados em fibra de Yb. Indo além, vimos que é possível calcular o coeficiente de Lyapunov do regime caótico com os dados obtidos a partir do laser. E não apenas isso, o ajuste utilizado para calcular o coeficiente de Lyapunov dos dados experimentais também foi capaz de reproduzir um mapa de retorno e coeficiente de Lyapunov muito próximos dos nossos dados.

Vimos também que é possível observar a rota para o caos diretamente partir dos instrumentos. No osciloscópio, podemos ver a variação dos máximos da série temporal. No analisador RF, fizemos uma varredura que se mostrou análoga ao diagrama de bifurcação. Além dessas medidas análogas, observamos que mesmo a potência média do laser revela as transições dinâmicas da rota para o caos, quando avaliada da perspectiva da transmissão não linear, devido à dinâmica de rotação não linear da polarização. Com isso, a rota para o caos no laser de modos travados em fibra de Yb pode ser identificada utilizando diferentes instrumentos e técnicas.

Embora a curva de ajuste tenha obtido sucesso em quantificar e prever o comportamento do laser, não foi possível estudar a fundo a relação entre o nosso sistema e o ajuste. Visto que esse ajuste é encontrado também no sistema de Rössler, um sistema dissipativo com as condições mínimas para o caos, investigar se o sistema do laser de Yb pode ser reduzido a um formato tão simples pode ser promissor. Deixamos essa sugestão para futuros trabalhos. Adicionalmente, nossa análise sobre a potência média nos levou a um segundo trabalho, onde estudamos transições discretas na separação de sólitons nesse mesmo laser (42). Isso concluiu nossos estudos de dinâmica não linear no laser de modos travados em fibra de Yb.

Mudando o foco para nosso trabalho mais recente, trouxemos uma análise numérica sobre como é possível otimizar a fase espectral de um pulso para absorção em uma transição de dois fótons por uma amostra densa de átomos. Para isso, utilizamos o auxílio do algoritmo genético, o qual nos ajudou a contornar a dificuldade do grande número de parâmetros a serem otimizados. Após nossas simulações, que utilizaram um modelo de boa compatibilidade com o experimento (25), apresentamos alterações de fase que otimizam o aumento da absorção,

com uma melhoria média de 200%.

Ao longo da nossa análise, observamos as mudanças de fase espectral necessárias para otimizar a absorção de pulsos por átomos individuais. Isso rendeu aumentos de absorção muito maiores, chegando a 2500%. Adicionado a isso, vimos também que a fase ótima é o resultado de uma superposição de efeitos, independente do sistema de transição de dois fótons, sendo os principais deles o ajuste para compensação do atraso entre os pulsos e a correção da interferência destrutiva. Isso permite simplificar a discussão, quanto a abordagem utilizando máscaras de fase para ajuste espectral.

No futuro, serão realizados experimentos de transição de dois fótons baseado nos resultados e abordagem deste trabalho. Também pretende-se explorar outras formas de otimizar a absorção, além utilizar o pulso de controle (25) e fase espectral. Alteração do foco do feixe e cortes de frequências são alguns exemplos de abordagem alternativa.

REFERÊNCIAS

1 LORENZ, E. N. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, v. 20, n. 2, p. 130–141, 1963.

2 STROGATZ, S. H. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. [S.l.]: CRC press, 2018.

3 MÉLO, L. B.; PALACIOS, G. F.; CARELLI, P. V.; ACIOLI, L. H.; LEITE, J. R. R.; MIRANDA, M. H. de. Deterministic chaos in an ytterbium-doped mode-locked fiber laser. *Optics Express*, Optica Publishing Group, v. 26, n. 10, p. 13686–13692, 2018.

4 LUO, L.; TEE, T.; CHU, P. Chaotic behavior in erbium-doped fiber-ring lasers. *JOSA B*, Optica Publishing Group, v. 15, n. 3, p. 972–978, 1998.

5 KOVALSKY, M. G.; HNILO, A. A. Stability and bifurcations in kerr-lens mode-locked ti: sapphire lasers. *Optics communications*, Elsevier, v. 186, n. 1-3, p. 155–166, 2000.

6 SÁNCHEZ, L. M.; HNILO, A. A. Description of kerr lens mode-locked lasers with poincaré maps in the complex plane. *Optics communications*, Elsevier, v. 199, n. 1-4, p. 189–199, 2001.

7 KOVALSKY, M. G.; HNILO, A. A. Different routes to chaos in the ti: sapphire laser. *Physical Review A*, APS, v. 70, n. 4, p. 043813, 2004.

8 ZHAO, L.; TANG, D.; LIU, A. Chaotic dynamics of a passively mode-locked soliton fiber ring laser. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, American Institute of Physics, v. 16, n. 1, p. 013128, 2006.

9 MÉLO, L. B. d. A. *Laser de modos travados em fibra de Itérbio construção e estudos sobre a dinâmica não linear*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2017.

10 RUNGE, A. F.; AGUERGARAY, C.; BRODERICK, N. G.; ERKINTALO, M. Raman rogue waves in a partially mode-locked fiber laser. *Optics letters*, Optica Publishing Group, v. 39, n. 2, p. 319–322, 2014.

11 LIU, Z.; ZHANG, S.; WISE, F. W. Rogue waves in a normal-dispersion fiber laser. *Optics letters*, Optica Publishing Group, v. 40, n. 7, p. 1366–1369, 2015.

12 KOBTSEV, S.; KUKARIN, S.; SMIRNOV, S.; TURITSYN, S.; LATKIN, A. Generation of double-scale femto/pico-second optical lumps in mode-locked fiber lasers. *Optics Express*, Optical Society of America, v. 17, n. 23, p. 20707–20713, 2009.

13 ZAYTSEV, A.; LIN, C.; YOU, Y.; TSAI, F.; WANG, C.; PAN, C. A controllable noise-like operation regime in a yb-doped dispersion-mapped fiber ring laser. *Laser Physics Letters*, IOP Publishing, v. 10, n. 4, p. 045104, 2013.

14 SUZUKI, M.; GANEEV, R. A.; YONEYA, S.; KURODA, H. Generation of broadband noise-like pulse from yb-doped fiber laser ring cavity. *Optics letters*, Optica Publishing Group, v. 40, n. 5, p. 804–807, 2015.

15 RUNGE, A. F.; BRODERICK, N. G.; ERKINTALO, M. Dynamics of soliton explosions in passively mode-locked fiber lasers. *JOSA B*, Optica Publishing Group, v. 33, n. 1, p. 46–53, 2016.

16 GRELU, P.; AKHMEDIEV, N. Dissipative solitons for mode-locked lasers. *Nature photonics*, Nature Publishing Group, v. 6, n. 2, p. 84–92, 2012.

17 MAY, R. M.; OSTER, G. F. Bifurcations and dynamic complexity in simple ecological models. *The American Naturalist*, University of Chicago Press, v. 110, n. 974, p. 573–599, 1976.

18 MANDEL, L.; WOLF, E. *Optical coherence and quantum optics*. [S.l.]: Cambridge university press, 1995.

19 COUTEAU, C. Spontaneous parametric down-conversion. *Contemporary Physics*, Taylor & Francis, v. 59, n. 3, p. 291–304, 2018.

20 BOUWMEESTER, D.; EKERT, A.; ZEILINGER, A. *The Physics of Quantum Information: Quantum Cryptography, Quantum Teleportation, Quantum Computation*. Springer Berlin Heidelberg, 2013. ISBN 9783662042106. Disponível em: <https://books.google.ie/books?id=y9kbswEACAAJ>.

21 BUSTARD, P. J.; LAUSTEN, R.; ENGLAND, D. G.; SUSSMAN, B. J. Toward quantum processing in molecules: A thz-bandwidth coherent memory for light. *Physical review letters*, APS, v. 111, n. 8, p. 083901, 2013.

22 BUSTARD, P. J.; ERSKINE, J.; ENGLAND, D. G.; NUNN, J.; HOCKETT, P.; LAUSTEN, R.; SPANNER, M.; SUSSMAN, B. J. Nonclassical correlations between terahertz-bandwidth photons mediated by rotational quanta in hydrogen molecules. *Optics letters*, Optica Publishing Group, v. 40, n. 6, p. 922–925, 2015.

23 ENGLAND, D. G.; FISHER, K. A.; MACLEAN, J.-P. W.; BUSTARD, P. J.; LAUSTEN, R.; RESCH, K. J.; SUSSMAN, B. J. Storage and retrieval of thz-bandwidth single photons using a room-temperature diamond quantum memory. *Physical review letters*, APS, v. 114, n. 5, p. 053602, 2015.

24 FANG, B.; DONG, S.; MEISELMAN, S.; COHEN, O.; LORENZ, V. O. Storage of ultra-broadband pulses in hot atomic barium vapor. In: OPTICA PUBLISHING GROUP. *CLEO: QELS_Fundamental Science*. [S.l.], 2017. p. FM2E–3.

25 CARVALHO, A.; MOREIRA, R.; FERRAZ, J.; VIANNA, S.; ACIOLI, L.; FELINTO, D. Enhanced absorption of weak ultrashort light pulses by a narrowband atomic medium. *Physical Review A*, APS, v. 101, n. 5, p. 053426, 2020.

26 LOZOVOY, V. V.; PASTIRK, I.; WALOWICZ, K. A.; DANTUS, M. Multiphoton intrapulse interference. ii. control of two-and three-photon laser induced fluorescence with shaped pulses. *The Journal of chemical physics*, American Institute of Physics, v. 118, n. 7, p. 3187–3196, 2003.

27 BALLARD, J. B.; STAUFFER, H. U.; MIROWSKI, E.; LEONE, S. R. Simultaneous control of time-dependent population transfer dynamics and wave-packet quantum interferences in li 2 by shaped ultrafast pulses. *Physical Review A*, APS, v. 66, n. 4, p. 043402, 2002.

28 COMSTOCK, M.; LOZOVOY, V. V.; PASTIRK, I.; DANTUS, M. Multiphoton intrapulse interference 6; binary phase shaping. *Optics express*, Optica Publishing Group, v. 12, n. 6, p. 1061–1066, 2004.

29 BALLARD, J. B.; STAUFFER, H. U.; AMITAY, Z.; LEONE, S. R. Optimization of wave packet coefficients in li 2 using an evolutionary algorithm: The role of resonant and nonresonant wavelengths. *The Journal of chemical physics*, American Institute of Physics, v. 116, n. 4, p. 1350–1360, 2002.

30 ANDO, T.; URAKAMI, T.; ITOH, H.; TSUCHIYA, Y. Optimization of resonant two-photon absorption with adaptive quantum control. *Applied physics letters*, American Institute of Physics, v. 80, n. 22, p. 4265–4267, 2002.

31 BAJPAI, P.; KUMAR, M. Genetic algorithm–an approach to solve global optimization problems. *Indian Journal of computer science and engineering*, Citeseer, v. 1, n. 3, p. 199–206, 2010.

32 DENG, L.-Z.; YAO, Y.-H.; DENG, L.; JIA, H.-Y.; ZHENG, Y.; XU, C.; LI, J.-P.; JIA, T.-Q.; QIU, J.-R.; SUN, Z.-R. et al. Tuning up-conversion luminescence in er3+-doped glass ceramic by phase-shaped femtosecond laser field with optimal feedback control. *Frontiers of Physics*, Springer, v. 14, n. 1, p. 1–7, 2019.

33 HUANG, Y.; XU, S.; JI, X.; YANG, X. Optimizing femtosecond laser-induced upconversion luminescence enhancement and suppression of dy3+-doped glass. *Laser Physics Letters*, IOP Publishing, v. 17, n. 1, p. 016001, 2019.

34 DUDOVICH, N.; DAYAN, B.; FAEDER, S. M. G.; SILBERBERG, Y. Transform-limited pulses are not optimal for resonant multiphoton transitions. *Physical Review Letters*, APS, v. 86, n. 1, p. 47, 2001.

35 ILDAY, F.; BUCKLEY, J.; KUZNETSOVA, L.; WISE, F. Generation of 36-femtosecond pulses from a ytterbium fiber laser. *Optics Express*, Optica Publishing Group, v. 11, n. 26, p. 3550–3554, 2003.

36 WANG, L.; XU, P.; LI, Y.; HAN, J.; GUO, X.; CUI, Y.; LIU, X.; TONG, L. Femtosecond mode-locked fiber laser at 1 μm via optical microfiber dispersion management. *Scientific Reports*, Nature Publishing Group, v. 8, n. 1, p. 1–7, 2018.

37 BAUMGARTL, M.; ORTAÇ, B.; SCHREIBER, T.; LIMPERT, J.; TÜNNERMANN, A. Ultrashort pulse formation and evolution in mode-locked fiber lasers. *Applied Physics B*, Springer, v. 104, n. 3, p. 523–536, 2011.

38 ILDAY, F. Ö.; CHEN, J.; KÄRTNER, F. Generation of sub-100-fs pulses at up to 200 mhz repetition rate from a passively mode-locked yb-doped fiber laser. *Optics Express*, Optica Publishing Group, v. 13, n. 7, p. 2716–2721, 2005.

39 RICHARDSON, D.; LAMING, R.; PAYNE, D.; PHILLIPS, M.; MATSAS, V. 320 fs soliton generation with passively mode-locked erbium fibre laser. *Electronics Letters*, v. 27, n. 9, p. 730–732, 1991.

40 OLSEN, L. F.; DEGN, H. Chaos in biological systems. *Quarterly reviews of biophysics*, Cambridge University Press, v. 18, n. 2, p. 165–225, 1985.

41 CAMPOS, C. L.; MÉLO, L. B.; CAVALCANTE, H. L. d. S.; ACIOLI, L. H.; MIRANDA, M. H. de. Internal nonlinear transmission in an yb mode-locked fiber laser through bifurcations. *Optics Communications*, Elsevier, v. 461, p. 125154, 2020.

42 CAMPOS, C. L.; MÉLO, L. B.; ACIOLI, L. H.; MIRANDA, M. H. de. Discrete evolution of temporal separation and relative phase for bound solitons in a mode-locked yb fiber laser. *OSA Continuum*, Optica Publishing Group, v. 3, n. 6, p. 1490–1497, 2020.

43 POLYCARPOU, C.; CASSEMIRO, K.; VENTURI, G.; ZAVATTA, A.; BELLINI, M. Adaptive detection of arbitrarily shaped ultrashort quantum light states. *Physical review letters*, APS, v. 109, n. 5, p. 053602, 2012.

44 COSTANZO, L. S.; COELHO, A. S.; PELLEGRINO, D.; MENDES, M. S.; ACIOLI, L.; CASSEMIRO, K. N.; FELINTO, D.; ZAVATTA, A.; BELLINI, M. Zero-area single-photon pulses. *Physical review letters*, APS, v. 116, n. 2, p. 023602, 2016.

45 MOORE, J. H. Artificial intelligence programming with labview: genetic algorithms for instrumentation control and optimization. *Computer Methods and Programs in Biomedicine*, Elsevier, v. 47, n. 1, p. 73–79, 1995.

46 CRISP, M. Propagation of small-area pulses of coherent light through a resonant medium. *Physical Review A*, APS, v. 1, n. 6, p. 1604, 1970.

47 CARVALHO, A. J. A. Rota para absorção de fótons individuais ultracurtos por um meio atômico. Universidade Federal de Pernambuco, 2020.

APÊNDICE A – CÓDIGO PARA ENCONTRAR A MÁSCARA DE FASE ÓTIMA EM UMA ABSORÇÃO DE DOIS FÓTONS COM UM ÚNICO PULSO

```
/*  This C++ code simulates an Rb atom transition carried out by a light pulse
    which is modified by a SLM with the purpose of absorption optimization. An GA
     is used to optimize the phase mask applied by the SLM. The atomic system and
     laser is mimics the system proposed by Dudovich et al. in "Transform-Limited
     Pulses Are Not Optimal for Resonant Multiphoton Transitions". The core of
    the atomic simulation code was written by Daniel Felinto and modified by
    Lucas B. A. Melo. The GA was written by Lucas B. A. Melo. The code was
    assembled by Lucas B. A. Melo */

#define _CRT_RAND_S         //MUST BE ON TOP! Used for rand_s function.

#include    <iostream>
#include    <limits.h>
#include    <time.h>
#include    <stdio.h>
#include    <stdlib.h>
#include    <math.h>

/*Data for the Genetic Algorithm*/
#define     ps  20          //Population Size.
#define     al  128         //Array Lenght.
#define     rf  0.6         //Recombination Frequency.
#define     mf  0.00015     //Mutation Frequency.
#define     it  5000            //Iterations.
#define     rn  1           //Number of runs.
#define     bt  8           //Number of Genes.

int random_integer(int floor, int roof);    //For random integers generation,
    going from 'floor' to 'roof' value.

double random_double(double roof, double floor);            //For random double
    generation.
double response(double input[], int index);                 //To simulate a
    system response.

/*GA functions*/
void random_population();                   //To create the initial population.
void fitness_function();                    //The selection process.
void keep_most_fit();                       //Replaces worse fits for the best
    ones.
void recombination();                       //Mixes individual properties.
void mutation();                            //Random changes on individuals.
```

```
33 double avg_position, avg_value;              //To store the population's avarage
       position and response value.
   double max_value, min_value;                 //To store the min and max values of
       the response.
35 double max0, max_mem, max_binary;            //To store local response results.

37 int max_index, min_index;                    //and the respective indexes.
   int max_mask;
39 int r, ite;                                      //To count the number of runs.

41 int             BINARY[ps][al][bt];     //All the Genes, organized by Population
       Size, Array Lenght and BiT.

43 double          GENE[ps][al];   //Each element in this vector represents a gene
       of an individual inside the population.

45 FILE* arq1;                 //For the files.
   char fname[_MAX_PATH];      //For the files names.
47
   unsigned int    number; //To store the rand_s result
49 errno_t          err;    //and track for errors when called.


51
   /*  unit of time: ps  */
53 /*  unit of length: cm  */

55 /*  constants  */
   #define     doispi  6.28318530717959
57 #define     c   0.03    /* light velocity */

59 /*  parameters of the sample  */
   #define     wl      (doispi*385.3) /* laser frequency in rad*(ps)^-1 */
61 #define     wa      (doispi*386.3) /* atomic frequency in rad*(ps)^-1 */
   #define     w12     (doispi*384.2) /* second atomic frequency in rad*(ps)^-1 */
63 #define     T2      (160)          /* T2 in ps */

65 /*  parameters of the laser */
   #define     Tp  0.10               /* initial pulse duration */
67
   /*  parameters of the time grating  */
69 #define     dt  0.01*16            /* discrete time step */
   #define     N   1024*32/16
71
   #define     lk  40                   /* logistic function inclination factor, for
       phase mask smoothing */
73 #define     gratres 0.5                /* [rad/ps] grating's angular frequency
       resolution */
```

```
75 #define      SWAP(a,b)    tempr=(a);(a)=(b);(b)=tempr

77 /*Functions for FFT and large vectors*/
   void         nrerror(char error_text[]);
79 double*      dvector(int nl, int nh);
   void         free_dvector(double* v, int nl, int nh);
81
   void         FFT(double coord[], double data[], int nn, double dd);
83 void         IFFT(double coord[], double data[], int nn, double dd);
   void         four1(double data[], int nn, int isign);
85
   double       pulse(double in[], int indx);
87
   void         prop_lin(double coord[], double data[], int nn, double dd, double ipt
       [], int ind);
89
   double       D;
91 double       MASK[3][128];

93 double* x, * fx, * fxu, * Area;

95 int main()
   {
97     int i, c1, c2;           //Loop counter.
       clock_t t;      //The clock-type variable to measure the run time.
99     double  a, b, a2, a3, a4, S1;
       double  Re, Im, ReEE, ImEE;         //For operations with complex quantities.
101
       //printf("Hello World!\n"); //Start checker.
103    t = clock();                //Start time

105    /*Making invalid start values to make debugging easier*/
       max_mask = -1;
107    max_mem = -1;

109    /*Stores the maximum possible value possible, based on the number of bt. Will
           be used to normalize future values.*/
       max_binary = 0;
111    for (i = 0; i < bt; ++i)
       {
113        max_binary += pow(2, i);
       }
115
       S1 = (doispi * Tp) / (2 * 1.763);   /* initial pulse area with normalized
           maximum */
117
```

```
    /*Creating special large vectors to facilitate the operations*/
119 x = dvector(0, N);
    fx = dvector(0, 2 * N);
121 fxu = dvector(0, 2 * N);
    Area = dvector(0, 2 * N);
123
    for (c1 = 0; c1 < N; c1 += 1)
125 {
        x[c1] = ((double)c1 - (double)N / 2) * (double)dt;                    /*
            time scale */
127     fx[2 * c1] = 1 / (S1 * cosh(1.763 * x[c1] / Tp)); /* real part of initial
             field */
        fx[2 * c1 + 1] = 0.0;                          /* imaginary part of initial
             field */
129
        a2 = (wa - wl) * x[c1];                            // Store the angular
            phase diference,
131     a3 = fx[2 * c1];                                   // the field real part
        a4 = fx[2 * c1 + 1];                               // and also the field
            imaginary part.
133     fxu[2 * c1] = a3 * cos(a2) - a4 * sin(a2);      // Then, apply a complex
            rotation using the angular phase diference to obtain the resulting
            field
        fxu[2 * c1 + 1] = a3 * sin(a2) + a4 * cos(a2);
135
        a2 = (w12 - wl) * x[c1];                           // Store the angular
            phase diference,
137     a3 = fx[2 * c1];                                   // the field real part
        a4 = fx[2 * c1 + 1];                               // and also the field
            imaginary part.
139     fx[2 * c1] = a3 * cos(a2) - a4 * sin(a2);       // Then, apply a complex
            rotation using the angular phase diference to obtain the resulting
            field
        fx[2 * c1 + 1] = a3 * sin(a2) + a4 * cos(a2);
141
        Area[2 * c1] = 0.0;                                //Sets pulse area to zero
            , to avoid problems.
143     Area[2 * c1 + 1] = 0.0;
    }
145
    /*These variables will be self-summed. Making it start as 0 prevents
        summation problems.*/
147 Re = 0;
    Im = 0;
149 max0 = 0;

151 Area[0] = fx[0] / 2;
```

```
    Area[1] = fx[1] / 2;

153

    /*Calculating the pulse area*/
155 for (c2 = 1; c2 < N; c2 += 1)
    {
157     Area[2 * c2] = (fx[2 * c2] + fx[2 * (c2 - 1)]) / 2 + Area[2 * (c2 - 1)];
        Area[2 * c2 + 1] = (fx[2 * c2 + 1] + fx[2 * (c2 - 1) + 1]) / 2 + Area[2 *
            (c2 - 1) + 1];
159 }

161 /*Intergral for normalization factor*/
    for (c1 = 1; c1 < N; c1 += 1)
163 {
        a = (fxu[2 * c1] + fxu[2 * (c1 - 1)]) / 2;
165     b = (fxu[2 * c1 + 1] + fxu[2 * (c1 - 1) + 1]) / 2;
        ReEE = a * Area[2 * c1] - b * Area[2 * c1 + 1];
167     Re += ReEE;
        ImEE = a * Area[2 * c1 + 1] + b * Area[2 * c1];
169     Im += ImEE;
    }
171

    max0 = Re * Re + Im * Im;
173 printf("Max0 = %g\n", max0);

175


177 snprintf(fname, _MAX_PATH, "InitialMacro.dat");
    fopen_s(&arq1, fname, "w");
179 if (arq1 == 0)
    {
181     printf("ERRO EM 'arq1' !\n");
        system("pause");
183     exit(0);
    }
185 else
    {
187     fprintf(arq1, "AvgVal\tMinVal\tMaxVal\n");
        fclose(arq1);
189 }

191 snprintf(fname, _MAX_PATH, "FinalMacro.dat");
    fopen_s(&arq1, fname, "w");
193 if (arq1 == 0)
    {
195     printf("ERRO EM 'arq1' !\n");
        system("pause");
197     exit(0);
```

```
    }
199 else
    {
201     fprintf(arq1, "AvgVal\tMinVal\tMaxVal\n");
        fclose(arq1);
203 }                                   //and closes it.

205 /*Loop for the number of program runs*/
    for (r = 0; r < rn; ++r)
207 {
        printf("Run #%d\n", r);
209
        snprintf(fname, _MAX_PATH, "Statistics_%d.dat", r);
211     fopen_s(&arq1, fname, "w");
        if (arq1 == 0)
213     {
            printf("ERRO EM 'arq1' !\n");
215         system("pause");
            exit(0);
217     }
        else
219     {
            fprintf(arq1, "AvgVal\tMinInd\tMinVal\tMaxInd\tMaxVal\n");
221         fclose(arq1);
        }                                   //and closes it.
223
        /*GA start*/
225     random_population();        //Creates initial population

227     /*runs the process 'it' times*/
        for (i = 0; i < it; ++i)
229     {
            ite = i;
231         printf("Iteration #%d\n", i);   //Prints the run number (for
                verification).
            fitness_function(); //checking population fitness
233         //printf("### Fitness ###\n");
            keep_most_fit();    //keeping only the bests individuals
235         //printf("### Discard ###\n");
            recombination();    //making them exchange information
237         //printf("### Recombine ###\n");
            mutation();         //and suffer random alterations.
239         //printf("### Mutate ###\n");

241         /*In the first iteration*/
            if (i == 0)
243         {
```

```
                snprintf(fname, _MAX_PATH, "InitialMacro.dat");
245             fopen_s(&arq1, fname, "a+");
                if (arq1 == 0)
247             {
                    printf("ERRO EM 'arq1' !\n");
249                 system("pause");
                    exit(0);
251             }
                else
253             {
                    fprintf(arq1, "%g\t%g\t%g\n", avg_value, min_value, max_value
                        );
255                 fclose(arq1);
                }                                   //and closes it.
257         }
        }
259
        snprintf(fname, _MAX_PATH, "FinalMacro.dat");
261     fopen_s(&arq1, fname, "a+");
        if (arq1 == 0)
263     {
            printf("ERRO EM 'arq1' !\n");
265         system("pause");
            exit(0);
267     }
        else
269     {
            fprintf(arq1, "%g\t%g\t%g\n", avg_value, min_value, max_value);
271         fclose(arq1);
        }                               //and closes it.
273 }

275 t = clock() - t;    //Stores the elapsed time
printf("Elapsed time: %g seconds\n", (double)t / (double)CLOCKS_PER_SEC);
                            //and prints it.
277
snprintf(fname, _MAX_PATH, "Data.dat");
279 fopen_s(&arq1, fname, "w");
if (arq1 == 0)
281 {
    printf("ERRO EM 'arq1' !\n");
283 system("pause");
    exit(0);
285 }
else
287 {
    fprintf(arq1, "Elapsed Time: %g seconds", (double)t / (double)
```

```
            CLOCKS_PER_SEC);
289         fclose(arq1);
    }                           //and closes the file.
291
    /*Releasing the memory used by the large vectors*/
293 free_dvector(x, 0, N);
    free_dvector(fx, 0, 2 * N);
295 free_dvector(fxu, 0, 2 * N);
    free_dvector(Area, 0, 2 * N);
297 _fcloseall();                           //Closes any remaining file.
    system("pause");                        //Waits for the user to press a key
        before closing the program.
299 return(0);                             //Retunrs an integer, as required by
        the 'main()' type.
}
301

303 void nrerror(char error_text[])
{
305 fprintf(stderr, "Numerical Recipes run-time error...\n");
    fprintf(stderr, "%s\n", error_text);
307 fprintf(stderr, "...now exiting to system...\n");
    exit(1);
309 }

311
double* dvector(int nl, int nh)
313 /* Allocates a double vector with range [nl..nh] */
{
315 double* v;
    v = (double*)malloc(static_cast<int64_t>((unsigned)(nh - nl + 1)) * sizeof(
        double));
317 if (!v) nrerror((char*)"allocation failure in dvector()");
    return v - nl;
319 }

321
void free_dvector(double* v, int nl, int nh)
323 /* Frees a double vector allocated by dvector() */
{
325 free((char*)(v + nl));
}
327

329 void four1(double data[], int nn, int isign)
{
331 int n, mmax, m, j, istep, i;
```

```
    double wtemp, wr, wpr, wpi, wi, theta;
    double tempr, tempi;

    n = nn << 1;
    j = 1;
    for (i = 1; i < n; i += 2) {
        if (j > i) {
            SWAP(data[j], data[i]);
            SWAP(data[j + 1], data[i + 1]);
        }
        m = n >> 1;
        while (m >= 2 && j > m) {
            j -= m;
            m >>= 1;
        }
        j += m;
    }
    mmax = 2;
    while (n > mmax) {
        istep = mmax << 1;
        theta = isign * (6.28318530717959 / mmax);
        wtemp = sin(0.5 * theta);
        wpr = -2.0 * wtemp * wtemp;
        wpi = sin(theta);
        wr = 1.0;
        wi = 0.0;
        for (m = 1; m < mmax; m += 2) {
            for (i = m; i <= n; i += istep) {
                j = i + mmax;
                tempr = wr * data[j] - wi * data[j + 1];
                tempi = wr * data[j + 1] + wi * data[j];
                data[j] = data[i] - tempr;
                data[j + 1] = data[i + 1] - tempi;
                data[i] += tempr;
                data[i + 1] += tempi;
            }
            wr = (wtemp = wr) * wpr - wi * wpi + wr;
            wi = wi * wpr + wtemp * wpi + wi;
        }
        mmax = istep;
    }
}


void FFT(double coord[], double data[], int nn, double dd)
{
    int c1;
```

```
379     double  tempr;

381     for (c1 = 0; c1 < nn; c1 += 2) {
            SWAP(data[c1], data[c1 + nn]);
383         SWAP(data[c1 + 1], data[c1 + 1 + nn]);
        }
385
        four1(data - 1, nn, 1);
387
        for (c1 = 0; c1 < nn; c1 += 2) {
389         SWAP(data[c1], data[c1 + nn]);
            SWAP(data[c1 + 1], data[c1 + 1 + nn]);
391     }
        for (c1 = 0; c1 < nn; c1 += 1) {
393         coord[c1] = 2 * 3.14159265 * ((double)c1 - (double)nn / 2) / (nn * dd);
            data[2 * c1] *= dd;
395         data[2 * c1 + 1] *= dd;
        }
397 }

399
    void IFFT(double coord[], double data[], int nn, double dd)
401 {
        int c1;
403     double  tempr;

405     for (c1 = 0; c1 < nn; c1 += 2) {
            SWAP(data[c1], data[c1 + nn]);
407         SWAP(data[c1 + 1], data[c1 + 1 + nn]);
        }
409
        four1(data - 1, nn, -1);
411
        for (c1 = 0; c1 < nn; c1 += 2) {
413         SWAP(data[c1], data[c1 + nn]);
            SWAP(data[c1 + 1], data[c1 + 1 + nn]);
415     }

417     for (c1 = 0; c1 < nn; c1 += 1) {
            coord[c1] = ((double)c1 - (double)nn / 2) / (nn * dd);
419         data[2 * c1] *= dd;
            data[2 * c1 + 1] *= dd;
421     }
    }
423

425 void random_population()
```

```
{
427     int i, j, k;    //Loop counters.

429     /*Runs for each individual in population*/
        for (i = 0; i < ps; ++i)
431     {
            /*each chromossome of the individual*/
433         for (j = 0; j < al; ++j)
            {
435             /*and each gene of the chromosome*/
                for (k = 0; k < bt; ++k)
437             {
                    /*Diferent types of initial population*/
439
                    /*Silberberg Start Mask
441                 if (j >= 57 && j <= 69)
                    {
443                     BINARY[i][j][k] = (int)(((int)(max_binary / 4)) / pow(2, k));
                            // This builds the equivalent of pi/2 with the available
                            binaries
                    }
445
                    else
447                 {
                        BINARY[i][j][k] = 0;
449                 }
                    */
451                 BINARY[i][j][k] = 0;
                    //BINARY[i][j][k] = random_integer(0, 1);   //gives it a random
                        integer number 0 or 1
453                 //printf_s("%g - %d - %d\n", GENE[i][j], i, j);     //and prints
                        on screen (for verification).
                }
455         }
        }
457 }

459
    void fitness_function()
461 {
        int i, j, k;        //Loop couters. 'i' for individual and 'j' for gene.
463
        double outcome; //To store the system response.
465
        /*To check for problems in the GENE matrix
467     for (i = 0; i < ps; ++i)                    //Runs for each individual in
            population
```

```
    {
469         for (j = 0; j < al; ++j)                //and each gene of the individual
            {
471             //GENE[i][j] = random_integer(0, 1);    //gives it a random integer
                    number 0 or 1
                if (GENE[i][j] > 1 || GENE[i][j] < 0)
473             {
                    printf_s("%d - %d - %d\n", GENE[i][j], i, j);       //and prints
                        on screen (for verification).
475                 getc(stdin);
                }
477         }
    }
479 */

481 avg_value = 0;      //Making it 0 to prevent alocation problems

483 /*Runs through each individual.*/
    for (i = 0; i < ps; ++i)
485 {
        for (j = 0; j < al; ++j)
487     {
            GENE[i][j] = 0;
489         for (k = 0; k < bt; ++k)
            {
491             GENE[i][j] += ((double)BINARY[i][j][k]) * (pow(2, k));
            }
493         GENE[i][j] = doispi * (GENE[i][j]) / max_binary;
        }
495

        outcome = pulse(GENE[i], i);    //Calls the 'response' function for the '
            sum' value and stores the 'outcome'.
497     //printf("Outcome = %g\n", outcome);

499     /*If it is the first run*/
        if (i == 0)
501     {
            max_value = outcome;    //stores the sum as the max outcome value
503         //max_position = sum[i];    //its position
            max_index = i;          //and index. Since it is the only sample for
                now
505

            min_value = outcome;    //do the same to min value
507         //min_position = sum[i];    //position
            min_index = i;          //and index
509     }
```

```
511         /*Otherwise*/
            else
513         {
                /*If the new outcome value is bigger than the last maximum outcome
                    value*/
515             if (outcome > max_value)
                {
517                 max_value = outcome;    //keep it as the maximum value
                    //max_position = sum[i];    //its position
519                 max_index = i;          //and index.
                }
521
                /*If the new outome value is smaller than the last minimum outcome
                    value*/
523             if (outcome < min_value)
                {
525                 min_value = outcome;    //keep it as the minimum value
                    //min_position = sum[i];    //its position
527                 min_index = i;          //and index.
                }
529         }

531         avg_value += outcome;   //and system response.
            //printf_s("Individual %d Sums %d\n", i, sum[i]);   //Prints the decimal
                value of the respective individual (for verification).
533     }

535     avg_value /= ps;    //and do the same for the system response.
        //printf_s("AvgVal %g MaxVal %d MaxInd %d MinVal %d MinInd %d\n", avg_value,
            max_value, max_index, min_value, min_index);//Prints everything on screen
             (for verification).
537
        /*
539     snprintf(fname, _MAX_PATH, "Population_%d.dat", r);//Name the population file
             for the r-th run.
        fopen_s(&arq1, fname, "a+");                        //Opens 'Population' file
541     runs a loop through all the individuals
        for (i = 0; i < ps; ++i)
543     {
            fprintf(arq1, "%d ", sum[i]);       //writing the decimal value for each
                one of them
545     }

547
        fprintf(arq1, "\n");                    //goes to the next line
549     fclose(arq1);                           //and closes the file.
        */
```

```
551
    snprintf(fname, _MAX_PATH, "Statistics_%d.dat", r);
553 fopen_s(&arq1, fname, "a+");
    if (arq1 == 0)
555 {
        printf("ERRO EM 'arq1' !\n");
557     system("pause");
        exit(0);
559 }
    else
561 {
        fprintf(arq1, "%g\t%d\t%g\t%d\t%g\n", avg_value, min_index, min_value,
            max_index, max_value);
563     fclose(arq1);
    }                                           //and closes it.
565 /*
    snprintf(fname, _MAX_PATH, "ArbitraryPhase_%d.dat", r);
567 fopen_s(&arq1, fname, "a+");
    for (j = 0; j < al; ++j)
569 {
        if (j == al - 1)
571     {
            fprintf(arq1, "%g\n", GENE[max_index][j]);
573     }

575     else
        {
577         fprintf(arq1, "%g ", GENE[max_index][j]);
        }
579 }
    fclose(arq1);
581 */
}
583


585 double pulse(double in[], int indx)
{
587 int     c1, c2;
    double  S1, max, Re, Im, a, b, a2, a3, a4;
589 double  ReEE, ImEE;
    FILE* arq1;
591

    D = wl - wa; /* detuning */
593 S1 = (doispi * Tp) / (2 * 1.763);   /* initial pulse area with normalized
        maximum */


595 for (c1 = 0; c1 < N; c1 += 1)
```

```
    {
597         x[c1] = ((double)c1 - (double)N / 2) * (double)dt;                    /*
                time scale */
            fx[2 * c1] = 1 / (S1 * cosh(1.763 * x[c1] / Tp));   /* real part of
                initial field */
599         fx[2 * c1 + 1] = 0.0;                               /* imaginary part of
                initial field */

601         Area[2 * c1] = 0;
            Area[2 * c1 + 1] = 0;
603     }

605     prop_lin(x, fx, (int)N, dt, in, indx);  /* Application of the mask */

607     /*These variables will be self-summed. Making it start as 0 prevents
            summation problems.*/
        max = 0;
609     Re = 0;
        Im = 0;
611
        for (c1 = 0; c1 < N; c1 += 1)
613     {
            a2 = (wa - wl) * x[c1];                         // Store the angular
                phase diference,
615         a3 = fx[2 * c1];                                // the field real part
            a4 = fx[2 * c1 + 1];                            // and also the field
                imaginary part.
617         fxu[2 * c1] = a3 * cos(a2) - a4 * sin(a2);      // Then, apply a complex
                rotation using the angular phase diference to obtain the resulting
                field
            fxu[2 * c1 + 1] = a3 * sin(a2) + a4 * cos(a2);
619
            a2 = (w12 - wl) * x[c1];                        // Store the angular
                phase diference,
621         a3 = fx[2 * c1];                                // the field real part
            a4 = fx[2 * c1 + 1];                            // and also the field
                imaginary part.
623         fx[2 * c1] = a3 * cos(a2) - a4 * sin(a2);       // Then, apply a complex
                rotation using the angular phase diference to obtain the resulting
                field
            fx[2 * c1 + 1] = a3 * sin(a2) + a4 * cos(a2);
625     }

627     /* |G(tau)|^2 */
        Area[0] = fx[0] / 2;
629     Area[1] = fx[1] / 2;
```

```
631     for (c2 = 1; c2 < N; c2 += 1)
        {
633         Area[2 * c2] = (fx[2 * c2] + fx[2 * (c2 - 1)]) / 2 + Area[2 * (c2 - 1)];
            Area[2 * c2 + 1] = (fx[2 * c2 + 1] + fx[2 * (c2 - 1) + 1]) / 2 + Area[2 *
                 (c2 - 1) + 1];
635     }

637     for (c1 = 1; c1 < N; c1 += 1)
        {
639         a = (fxu[2 * c1] + fxu[2 * (c1 - 1)]) / 2;
            b = (fxu[2 * c1 + 1] + fxu[2 * (c1 - 1) + 1]) / 2;
641         ReEE = a * Area[2 * c1] - b * Area[2 * c1 + 1];
            Re += ReEE;
643         ImEE = a * Area[2 * c1 + 1] + b * Area[2 * c1];
            Im += ImEE;
645     }

647     max = (Re * Re + Im * Im) / max0;
        //printf("Max = %g\n", max);
649
        /*When the algorithm produces a better output, stores the pulse information
            and mask */
651     if (ite == 0)
        {
653         max_mem = -1;
        }
655
        if (max > max_mem)
657     {
            max_mem = max;
659
            snprintf(fname, _MAX_PATH, "Campo_Max_%d.dat", r);
661         fopen_s(&arq1, fname, "w");
            if (arq1 == 0)
663         {
                printf("ERRO EM 'arq1' !\n");
665             system("pause");
                exit(0);
667         }
            else
669         {
                for (c1 = 0; c1 < N; c1 += 1)
671             {
                    fprintf(arq1, "%g %g\n", x[c1], fx[2 * c1] * fx[2 * c1] + fx[2 *
                        c1 + 1] * fx[2 * c1 + 1]);                    //saves the
                        header
673             }
```

```
            fclose(arq1);
675         }

677         FFT(x, fx, (int)N, dt);

679

            snprintf(fname, _MAX_PATH, "ArbitraryPhase_%d.dat", r);
681         fopen_s(&arq1, fname, "w");
            if (arq1 == 0)
683         {
                printf("ERRO EM 'arq1' !\n");
685             system("pause");
                exit(0);
687         }
            else
689         {
                for (c1 = 0; c1 < al; ++c1)
691             {
                    fprintf(arq1, "%g\n", GENE[max_index][c1]);
693             }
                fclose(arq1);
695         }
    }
697
    return(max);
699 }

701
void prop_lin(double coord[], double data[], int nn, double dd, double ipt[], int
     ind)
703 {
    int c1, c2, c3;
705     double  a2, a3, a4;

707     FFT(coord, data, nn, dd);

709     for (c1 = 0; c1 < 128; ++c1)
    {
711         MASK[0][c1] = c1 - 128.0 / 2.0;                 // Setting possible
                integer values ranging from -64 to 64
            MASK[1][c1] = MASK[2][c1] = 0.0;                // Setting all else to 0
                to prevent allocation problems
713         MASK[1][c1] = ipt[c1];
    }
715
    /*Using Logistic Funtion to make the Phase Mask soft*/
717     a2 = 0.0;
```

```
    c3 = -1;                                                    // Variable used to
        keep track of the bit section in the following loop:
719 for (c1 = 0; c1 < nn; ++c1)
    {
721     if (fabs(coord[c1]) <= 64)
        {
723         c2 = (int)floor(coord[c1]) + 64;
            if (c2 < 128 && c2 >= 0)
725         {
                /* Smooth Phase Mask */
727             if (fabs(coord[c1] - floor(coord[c1])) <= gratres / 2 || fabs(
                    coord[c1] - floor(coord[c1])) >= 1 - gratres / 2)
                {
729                 /* Left Side */
                    if (fabs(coord[c1] - floor(coord[c1])) <= gratres / 2)
731                 {
                        /* Border Exception */
733                     if (floor(coord[c1]) == -64)
                        {
735                         a2 = MASK[1][c2] / (1 + exp(-lk * (coord[c1] - floor(
                                coord[c1]))));

737                         data[2 * c1] = data[2 * c1] / (1 + exp(-lk * (coord[
                                c1] - floor(coord[c1]))));
                            data[2 * c1 + 1] = data[2 * c1 + 1] / (1 + exp(-lk *
                                (coord[c1] - floor(coord[c1]))));
739                     }
                        else
741                     {
                            a2 = MASK[1][c2 - 1] + (MASK[1][c2] - MASK[1][c2 -
                                1]) / (1 + exp(-lk * (coord[c1] - floor(coord[c1
                                ]))));
743                     }
                    }
745
                    /* Right Side */
747                 if (fabs(coord[c1] - floor(coord[c1])) >= 1 - gratres / 2)
                    {
749                     /* Border Exception */
                        if (floor(coord[c1]) == 63)
751                     {
                            a2 = MASK[1][c2] - MASK[1][c2] / (1 + exp(-lk * (
                                coord[c1] - (floor(coord[c1] + 1)))));
753
                            data[2 * c1] = data[2 * c1] - data[2 * c1] / (1 + exp
                                (-lk * (coord[c1] - (floor(coord[c1] + 1)))));
755                         data[2 * c1 + 1] = data[2 * c1 + 1] - data[2 * c1 +
```

```
                                1] / (1 + exp(-lk * (coord[c1] - (floor(coord[c1]
                                 + 1)))));
                        }
757                     else
                        {
759                         if (c2 < 127)
                            {
761                             a2 = MASK[1][c2] + (MASK[1][c2 + 1] - MASK[1][c2
                                    ]) / (1 + exp(-lk * (coord[c1] - (floor(coord
                                    [c1] + 1)))));
                            }
763                     }
                    }
765             }
                else
767             {
                    a2 = MASK[1][c2];                                   // Store the
                        angular phase diference,
769             }
            }
771         else
            {
773             a2 = 0.0;
            }
775     }
        else
777     {
            a2 = 0.0;
779         if (floor(coord[c1]) == -65 || floor(coord[c1]) == 64)
            {
781             if (floor(coord[c1]) == -65)
                {
783                 if (fabs(coord[c1] - floor(coord[c1])) >= 1 - gratres / 2)
                    {
785                     a2 = MASK[1][0] / (1 + exp(-lk * (coord[c1] - (floor(
                        coord[c1] + 1)))));

787                     data[2 * c1] = data[2 * c1] / (1 + exp(-lk * (coord[c1] -
                         (floor(coord[c1] + 1)))));
                        data[2 * c1 + 1] = data[2 * c1 + 1] / (1 + exp(-lk * (
                        coord[c1] - (floor(coord[c1] + 1)))));
789                 }
                    else
791                 {
                        data[2 * c1] = 0;
793                     data[2 * c1 + 1] = 0;
                    }
```

```
795                 }

797                 if (floor(coord[c1]) == 64)
                    {
799                     if (fabs(coord[c1] - floor(coord[c1])) <= gratres / 2)
                        {
801                         a2 = MASK[1][127] - MASK[1][127] / (1 + exp(-lk * (coord[
                                c1] - floor(coord[c1]))));

803                         data[2 * c1] = data[2 * c1] - data[2 * c1] / (1 + exp(-lk
                                 * (coord[c1] - floor(coord[c1]))));
                            data[2 * c1 + 1] = data[2 * c1 + 1] - data[2 * c1 + 1] /
                                (1 + exp(-lk * (coord[c1] - floor(coord[c1]))));
805                     }
                        else
807                     {
                            data[2 * c1] = 0;
809                         data[2 * c1 + 1] = 0;
                        }
811                 }
                }
813             else
                {
815                 data[2 * c1] = 0;
                    data[2 * c1 + 1] = 0;
817             }
            }
819
            a3 = data[2 * c1];                              // the field real part
821         a4 = data[2 * c1 + 1];                          // and also the field
                imaginary part.
            data[2 * c1] = a3 * cos(a2) - a4 * sin(a2);     // Then, apply a complex
                rotation using the angular phase diference to obtain the resulting
                field
823         data[2 * c1 + 1] = a3 * sin(a2) + a4 * cos(a2);

825     }

827     //printf_s("ind = %d; max_mask = %d\n", ind, max_mask);

829     /*
        if(ind == max_mask)
831     {
            fopen_s(&arq1, "ArbitraryMask_OD800b.dat", "w");
833         for (c1 = 0; c1 < nn; ++c1)
            {
835             if (fabs(coord[c1]) < 64)
```

```
            {
                c2 = floor(coord[c1]) + 64;
                a2 = MASK[1][c2];
            }
            else a2 = 0.0;
            fprintf(arq1, "%g\t%g\n", coord[c1], a2);
        }
        fclose(arq1);
    }
    */

    IFFT(coord, data, nn, 1 / (nn * dd));
}


void recombination()
{
    int i, j, k;                //Loop couters.
    int cut;                //Sets the point to split the individual genes for
        recombination.
    int sample_A, sample_B; //The numeric label of the individuals A and B to be
        recombined.

    double  COPY[bt];           //To copy the individuals during the
        recombination.

    /*Runs through 'ps*rf' individuals*/
    for (i = 0; i <= ps * rf; ++i)
    {
        sample_A = random_integer(0, ps - 1);           //randomly choosing
            individual A
        //printf_s("Sample A is %u\n", sample_A);       //and displays its
            numeric label (for verification)
        sample_B = random_integer(0, ps - 1);           //then randomly chooses
            individual B
        //printf_s("Sample B is %u\n", sample_B);       //and displays its
            numeric label (for verification).

        for (j = 0; j < al; ++j)
        {
            cut = random_integer(0, bt - 1);                //Randomly chooses
                the individuals split point
            //printf_s("Cut at %u\n", cut);                 //and displays the
                point (for verification).

            /*Runs through each gene up to the 'cut' point.*/
            for (k = 0; k <= cut; ++k)
```

```
            {
875             //printf_s("Sample A %u Sample B %u becomes\n", GENE[sample_A][j
                    ], GENE[sample_B][j]);//prints the initial situation (for
                    verification)
                COPY[k] = BINARY[sample_A][j][k];           //stores the copy A
                    gene
877             BINARY[sample_A][j][k] = BINARY[sample_B][j][k];    //replaces it
                     by the B gene
                BINARY[sample_B][j][k] = (int)COPY[k];          //then replaces B
                     gene by the copy of A
879             //printf_s("Sample A %u Sample B %u\n", GENE[sample_A][j], GENE[
                    sample_B][j]);//and prints the final situation (for
                    verification).
            }
881     }
    }
883 }

885
int random_integer(int floor, int roof)
887 {
    int result; //To store the generated random number as integer.
889
    double res; //To store the generated random number.
891
    result = roof + 1;  //Making the 'result' outside range to make it go into
        the "while".
893 //printf_s("Random Number  %u\n", (unsigned int)((double)number));//Prints '
        rand_s' original result (for verification)

895 /*Generates a random number until it is inside the range*/
    while (result > roof || result < floor)
897 {
        err = rand_s(&number);  //Creates the random number, stores in 'number'
            and returns 0(false) or 1(true) for 'err'.
899
        if (err != 0)                                   //If the 'err' is true
901     {
            printf("The rand_s function failed!\n");    //prints it on screen
903         exit(0);                                    //and closes aplication.
        }
905
        res = ((double)number / ((double)UINT_MAX + 1) * (double)((double)roof +
            1)) + (double)floor;//Generates a random number inside the range
907     result = (unsigned int)res; //stores it as integer;
    }
909
```

```
    //printf_s("Random Result %u\n", result);   //prints on screen (for
        verification)
    return(result);                             //and returns the 'result' to
        where it's needed.
}


void keep_most_fit()
{
    int j, k;   //Loop counter for gene.

    /*Runs through all the genes*/
    for (j = 0; j < al; ++j)
    {
        for (k = 0; k < bt; ++k)
        {
            //printf_s("Before %d ", GENE[min_index][j]);   //prints the gene
                from worst individual (for verification)
            BINARY[min_index][j][k] = BINARY[max_index][j][k];      //replaces it
                 with the gene from best individual
            //printf_s("After %d\n", GENE[min_index][j]);   //and prints the
                modification (for verification).
        }
    }
}


void mutation()
{
    int j;          //Loop counter.
    int sample, gen, bit;//To select the individual and gene to be mutated.

    /*Runs through 'ps*al*mf' genes from all individual*/
    for (j = 0; j <= (int)(ps * al * bt * mf); ++j)
    {
        sample = random_integer(0, ps - 1);             //randomly choosing an
            individual
        //printf("Sample %u ", sample);                 //printing the choise (
            for verification).
        gen = random_integer(0, al - 1);                //Then randomly chooses a
             gene
        //printf("gen %u\n", gen);                      //printing its position (
            for verification)
        bit = random_integer(0, bt - 1);
        //printf("Gene %u", GENE[sample][gen]);         //and value (for
            verification).
        BINARY[sample][gen][bit] = random_integer(0, 1);        //Replaces the
```

```
            original gene value for a random one
947         //printf(" became %u\n", GENE[sample][gen];    //and prints the mutated
            gene (for verification).
    }
949 }
```

APÊNDICE B – CÓDIGO PARA ENCONTRAR A MÁSCARA DE FASE ÓTIMA EM UMA ABSORÇÃO SEQUENCIAL DE DOIS FÓTONS NUMA AMOSTRA ATÔMICA DENSA

```
/*This code uses a genectic algorithm to optimize the spectral phase of
a signal laser pulse to be utilized in an sequential two-photno absorption.
The atomic model was provided by Daniel Felinto. The genectic algorithm was
built and modified for 2D optmization by Lucas Melo. The two programs were
put together by Lucas Melo.*/
#define _CRT_RAND_S         //MUST BE ON TOP! Used for rand_s function.

#include    <iostream>
#include    <limits.h>
#include    <time.h>
#include    <stdio.h>
#include    <stdlib.h>
#include    <math.h>


/*Data for the Genetic Algorithm*/
#define     ps  20          //Population Size.
#define     al  128         //Array Lenght.
#define     rf  0.6         //Recombination Frequency.
#define     mf  0.0003//0.001       //Mutation Frequency.
#define     it  5000            //Iterations.
#define     rn  1           //Number of runs.
#define     bt  8           //Number of Genes.

int random_integer(int floor, int roof);    //For random integers generation,
    going from 'floor' to 'roof' value.

double random_double(double roof, double floor);
double response(double input[], int index);                 //To simulate a
    system response.

double pulse(double in[], int indx);

void random_population();                   //To create the initial population.
void fitness_function();                    //The selection process.
void keep_most_fit();                       //Replaces worse fits for the best
    ones.
void recombination();                       //Mixes individual properties.
void mutation();                            //Random changes on individuals.

double avg_value;               //To store the population's avarage position and
```

```
    response value.
double max_value, min_value;                //To store the min and max values of
    the response
double max0, max_mem, max_binary, max_tau_bin;

int max_index, min_index;                   //and the respective indexes.
int r, ite;                                     //To count the number of runs.

int             BINARY[ps][al][bt];     //All the Genes, organized by Population
    Size, Array Lenght and BiT.
//int               TAU_BIN[ps][ts];

double          GENE[ps][al];   //Each element in this vector represents a gene
    of an individual inside the population.
double          MASK[3][al];

FILE* arq1;                 //For the files.
char fname[_MAX_PATH];      //For the files names.

unsigned int    number; //To store the rand_s result
errno_t         err;    //and track for errors when called.

/*Data for the Integration*/

#define NR_END 1
#define FREE_ARG char*

#define     SWAP(a,b)   tempr=(a);(a)=(b);(b)=tempr

/*  constantes  */

#define     doispi  6.28318530717959


/*  parametros do meio  */
#define     wl      (doispi*377.69) /* frequencia do laser em rad*(ps)^-1 */
#define     wa      (doispi*377.36) /* frequencia do atomo em rad*(ps)^-1 */
#define     T2      (285.0)          /* T2 */
#define     OD       (720.0)//(38.0)           /* distancia percorrida na amostra
    */
#define     ODm     (OD/4.0)


/*  parametros do laser  */

#define     Tp  0.190           /* largura temporal do pulso de sinal */
#define     Tpc 0.084           /* largura temporal do pulso de control */
```

```
81
83  /*  parametros da grade temporal  */
85  #define     N   1024*2          /* numero de elementos da grade */
    #define     dt  0.01          /* diferencial de tempo */
87  #define     Nl  71
89  #define     Thp 0.001       /* area do pulso de prova */
    #define     Thc 0.10        /* area do pulso de controle */
91
    /*Parametros da grade de difracao*/
93  #define     lk  40                  /* logistic function inclination factor, for
        phase mask smoothing */
    #define     gratres 0.5                 /* [rad/ps] grating's angular frequency
        resolution */
95
    double D, Soma;
97
    double AwR, TaR;
99  double AwI, TaI;
101 double* x, * fx, * xc2, * fxc2;
    double* xc, * fxc, * FAw, * EAw;
103
    double Staus[Nl][2];
105
    void nrerror(char error_text[]);
107 double* dvector(int nl, int nh);
    void free_dvector(double* v, int nl, int nh);
109
    void GaussPulse(double coord[], double data[], double nn, double dd, double width
        , double area);
111 void Dessintonia(double coord[], double data[], double nn, double dd, double Dd);
    void GVD(double coord[], double data[], int nn, double dd, double L);
113 void FFT(double coord[], double data[], int nn, double dd);
    void four1(double data[], int nn, int isign);
115 void IFFT(double coord[], double data[], int nn, double dd);
    void Aw(double w);
117 void prop_lin(double coord[], double data[], int nn, double dd);
    void prop_lin2(double coord[], double data[], int nn, double dd, double ipt[],
        int ind);
119 void Tw1w2Bmod(int c1, int c2);
121 double** dmatrix(long nrl, long nrh, long ncl, long nch);
    void free_dmatrix(double** m, long nrl, long nrh, long ncl, long nch);
123
```

```
int main()
{
    int i, c1, c2;          //Loop counter.
    clock_t t;      //The clock-type variable to measure the run time.
    double tt;      //The double-type variable to store the elapsed time.

    double fa1R;
    double fa1I;

    double frs;
    double fis;

    t = clock();                //Start time

    /*Stores the maximum possible value possible, based on the number of bt. Will
         be used to normalize future values.*/
    max_binary = 0;
    for (i = 0; i < bt; ++i)
    {
        max_binary += pow(2, i);
    }

    max_mem = 0;

    /* memoria para vetores dos campos e das funcoes auxiliares Aw e EAw */

    x = dvector(0, N);
    fx = dvector(0, 2 * N);

    xc = dvector(0, N);
    fxc = dvector(0, 2 * N);

    xc2 = dvector(0, N);
    fxc2 = dvector(0, 2 * N);

    FAw = dvector(0, 2 * N);
    EAw = dvector(0, 2 * N);

    /*Create the files header*/
    snprintf(fname, _MAX_PATH, "InitialMacro.dat");
    fopen_s(&arq1, fname, "w");
    if (arq1 == 0)
    {
        printf("ERRO EM 'arq1' !\n");
        system("pause");
        exit(0);
    }
```

```
    else
171 {
        fprintf(arq1, "AvgVal\tMinVal\tMaxVal\n");
173     fclose(arq1);
    }
175
    snprintf(fname, _MAX_PATH, "FinalMacro.dat");
177 fopen_s(&arq1, fname, "w");
    if (arq1 == 0)
179 {
        printf("ERRO EM 'arq1' !\n");
181     system("pause");
        exit(0);
183 }
    else
185 {
        fprintf(arq1, "AvgVal\tMinVal\tMaxVal\n");
187     fclose(arq1);
    }
189
    /* Definindo os pulsos gaussianos */
191
    GaussPulse(x, fx, (double)N, dt, Tp, Thp);
193
    D = wl - wa;
195 Dessintonia(x, fx, (double)N, dt, D);

197 GVD(x, fx, (int)N, dt, 0.0);

199 GaussPulse(xc2, fxc2, (double)N, dt, Tpc, Thc);

201 FFT(xc2, fxc2, N, dt);

203 /*Normalizing Integral*/
    frs = 0;
205 fis = 0;

207 for (c1 = 0; c1 < N; c1 += 1) {
        c2 = N - 1 - c1;
209     xc[c1] = xc2[c1];
        fxc[2 * c1] = fxc2[2 * c2];
211     fxc[2 * c1 + 1] = -fxc2[2 * c2 + 1];
        //fa1 = Aw(xc[c1]);
213     Aw(xc[c1]);
        fa1R = AwR;
215     fa1I = AwI;
        FAw[2 * c1] = fa1R;
```

```
217         FAw[2 * c1 + 1] = fa1I;
            EAw[2 * c1] = exp(-OD * fa1R) * cos(OD * fa1I);
219         EAw[2 * c1 + 1] = -exp(-OD * fa1R) * sin(OD * fa1I);

221         frs += EAw[2 * c1];
            fis += EAw[2 * c1 + 1];
223     }

225     /* propagacao linear do campo sinal sem controle */

227     prop_lin(x, fx, N, dt);

229     /*Loop for the number of program runs*/
        for (r = 0; r < rn; ++r)
231     {
            printf("Run #%d\n", r);
233

235         snprintf(fname, _MAX_PATH, "Statistics_%d.dat", r);
            fopen_s(&arq1, fname, "w");
237         if (arq1 == 0)
            {
239             printf("ERRO EM 'arq1' !\n");
                system("pause");
241             exit(0);
            }
243         else
            {
245             fprintf(arq1, "AvgVal\tMinInd\tMinVal\tMaxInd\tMaxVal\n");
                fclose(arq1);
247         }

249         random_population();        //Creates initial population

251         /*runs the process 'it' times*/
            for (i = 0; i < it; ++i)
253         {
                ite = i;
255

                printf("Iteration #%d\n", i);   //Prints the run number (for
                    verification).
257             fitness_function(); //checking population fitness
                keep_most_fit();    //keeping only the bests individuals
259             recombination();    //making them exchange information
                mutation();         //and suffer random alterations.
261

                /*In the first iteration*/
```

```
263             if (i == 0)
                {
265                 snprintf(fname, _MAX_PATH, "InitialMacro.dat");
                    fopen_s(&arq1, fname, "a+");
267                 if (arq1 == 0)
                    {
269                     printf("ERRO EM 'arq1' !\n");
                        system("pause");
271                     exit(0);
                    }
273                 else
                    {
275                     fprintf(arq1, "%g\t%g\t%g\n", avg_value, min_value, max_value
                            );
                        fclose(arq1);
277                 }
                }
279         }

281         snprintf(fname, _MAX_PATH, "FinalMacro.dat");
            fopen_s(&arq1, fname, "a+");
283         if (arq1 == 0)
            {
285             printf("ERRO EM 'arq1' !\n");
                system("pause");
287             exit(0);
            }
289         else
            {
291             fprintf(arq1, "%g\t%g\t%g\n", avg_value, min_value, max_value);
                fclose(arq1);
293         }
        }
295
        tt = ((double)clock() - (double)t) / (double)CLOCKS_PER_SEC;    //Stores the
            elapsed time
297     printf("Elapsed time: %g seconds\n", tt);                           //and
            prints it.

299     snprintf(fname, _MAX_PATH, "Data.dat");
        fopen_s(&arq1, fname, "w");
301     if (arq1 == 0)
        {
303         printf("ERRO EM 'arq1' !\n");
            system("pause");
305         exit(0);
        }
```

```
307     else
        {
309         fprintf(arq1, "Elapsed Time: %g seconds", tt);
            fclose(arq1);
311     }

313     /* liberacao da memoria dos vetores de campo e das funcoes auxiliares Aw e
            EAw */

315     free_dvector(x, 0, N);
        free_dvector(fx, 0, 2 * N);
317
        free_dvector(xc, 0, N);
319     free_dvector(fxc, 0, 2 * N);

321     free_dvector(xc2, 0, N);
        free_dvector(fxc2, 0, 2 * N);
323
        free_dvector(FAw, 0, 2 * N);
325     free_dvector(EAw, 0, 2 * N);

327     //free_dvector(Statistics1, 0, it);

329     _fcloseall();                           //Closes any remaining file.

331     system("pause");                            //Waits for the user to press a
            key before closing the program.

333     return(0);                              //Retunrs an integer, as required by
            the 'main()' type.
    }
335
    int random_integer(int floor, int roof)
337 {
        int result; //To store the generated random number as integer.
339
        double res; //To store the generated random number.
341
        result = roof + 1;  //Making the 'result' outside range to make it go into
            the "while".
343     //printf_s("Random Number  %u\n", (unsigned int)((double)number));//Prints '
            rand_s' original result (for verification)

345     /*Generates a random number until it is inside the range*/
        while (result > roof || result < floor)
347     {
            err = rand_s(&number);  //Creates the random number, stores in 'number'
```

```
            and returns 0(false) or 1(true) for 'err'.

        if (err != 0)                                   //If the 'err' is true
        {
            printf("The rand_s function failed!\n");    //prints it on screen
            exit(0);                                    //and closes aplication.
        }

        res = ((double)number / ((double)UINT_MAX + 1) * (double)((double)roof +
            1)) + (double)floor;//Generates a random number inside the range
        result = (unsigned int)res; //stores it as integer;
    }

    //printf_s("Random Result %u\n", result);   //prints on screen (for
        verification)
    return(result);                             //and returns the 'result' to
        where it's needed.
}

double random_double(double floor, double roof)
{
    double res; //To store the generated random number.

    res = roof + 1; //Making the 'result' outside range to make it go into the "
        while".
    //printf_s("Random Number  %u\n", (unsigned int)((double)number));//Prints '
        rand_s' original result (for verification)

    /*Generates a random number until it is inside the range*/
    while (res > roof || res < floor)
    {
        err = rand_s(&number);  //Creates the random number, stores in 'number'
            and returns 0(false) or 1(true) for 'err'.

        if (err != 0)                                   //If the 'err' is true
        {
            printf("The rand_s function failed!\n");    //prints it on screen
            exit(0);                                    //and closes aplication.
        }

        res = ((double)number / ((double)UINT_MAX + 1) * (roof + 1)) + floor;//
            Generates a random number inside the range
    }

    //printf_s("Random Result %g\n", res);  //prints on screen (for verification)
    return(res);                                //and returns the result to where
         it's needed.
```

```
}

void random_population()
{
    int i, j, k;    //Loop counters.
    /*Runs for each individual in population*/
    for (i = 0; i < ps; ++i)
    {
        /*each chromossome of the individual*/
        for (j = 0; j < al; ++j)
        {
            /*and each gene of the chromosome*/
            for (k = 0; k < bt; ++k)
            {
                /*Diferent types of initial population*/

                /*Silberberg Start Mask
                if (j >= 57 && j <= 69)
                {
                    BINARY[i][j][k] = (int)(((int)(max_binary / 4)) / pow(2, k));
                        // This builds the equivalent of pi/2 with the available
                        binaries
                }

                else
                {
                    BINARY[i][j][k] = 0;
                }
                */
                BINARY[i][j][k] = 0;
                //BINARY[i][j][k] = random_integer(0, 1);   //gives it a random
                    integer number 0 or 1
                //printf_s("%g - %d - %d\n", GENE[i][j], i, j);     //and prints
                    on screen (for verification).
            }
        }
    }
}

void fitness_function()
{
    int i, j, k;        //Loop couters. 'i' for individual and 'j' for gene.

    double outcome; //To store the system response.

    /*To check for problems in the GENE matrix
    for (i = 0; i < ps; ++i)                    //Runs for each individual in
```

```
        population
    {
431         for (j = 0; j < al; ++j)                //and each gene of the individual
            {
433             //GENE[i][j] = random_integer(0, 1);    //gives it a random integer
                    number 0 or 1
                if (GENE[i][j] > 1 || GENE[i][j] < 0)
435             {
                    printf_s("%d - %d - %d\n", GENE[i][j], i, j);       //and prints
                        on screen (for verification).
437                 getc(stdin);
                }
439         }
    }
441 */

443 avg_value = 0;      //Making it 0 to prevent alocation problems

445 /*Runs through each individual.*/
    for (i = 0; i < ps; ++i)
447 {
        for (j = 0; j < al; ++j)
449     {
            GENE[i][j] = 0;
451         for (k = 0; k < bt; ++k)
            {
453             GENE[i][j] += ((double)BINARY[i][j][k]) * (pow(2, k));
            }
455         GENE[i][j] = doispi * (GENE[i][j]) / max_binary;
        }
457     /*
        if (j == al)
459     {
            GENE[i][j] = 0;
461         for (k = 0; k < bt; ++k)
            {
463             GENE[i][j] += ((double)BINARY[i][j][k]) * pow(2, k);
            }
465         //printf("Tau = %g \n", GENE[i][j]);
            GENE[i][j] = ts * ((GENE[i][j]) / max_binary - 0.5);
467         //printf("Tau = %g \n", GENE[i][j]);
        }
469     */
        outcome = pulse(GENE[i], i);    //Calls the 'response' function for the '
            sum' value and stores the 'outcome'.
471     //printf("%g\n", outcome);
```

```
473         /*If it is the first run*/
            if (i == 0)
475         {
                max_value = outcome;    //stores the sum as the max outcome value
477             //max_position = sum[i];    //its position
                max_index = i;          //and index. Since it is the only sample for
                    now
479
                min_value = outcome;    //do the same to min value
481             //min_position = sum[i];    //position
                min_index = i;          //and index
483
                if (ite == 0)
485             {
                    max_mem = outcome;
487             }
            }
489
            /*Otherwise*/
491         else
            {
493             /*If the new outcome value is bigger than the last maximum outcome
                    value*/
                if (outcome > max_value)
495             {
                    max_value = outcome;    //keep it as the maximum value
497                 //max_position = sum[i];    //its position
                    max_index = i;          //and index.
499             }

501             /*If the new outome value is smaller than the last minimum outcome
                    value*/
                if (outcome < min_value)
503             {
                    min_value = outcome;    //keep it as the minimum value
505                 //min_position = sum[i];    //its position
                    min_index = i;          //and index.
507             }
            }
509
            avg_value += outcome;   //and system response.
511         //printf_s("Individual %d Sums %d\n", i, sum[i]);   //Prints the decimal
                value of the respective individual (for verification).
        }
513
        avg_value /= ps;    //and do the same for the system response.
515
```

```
    /*Saving the genectic algorithm progress*/
517     snprintf(fname, _MAX_PATH, "Statistics_%d.dat", r);
    fopen_s(&arq1, fname, "a+");
519     if (arq1 == 0)
    {
521         printf("ERRO EM 'arq1' !\n");
        system("pause");
523         exit(0);
    }
525     else
    {
527         fprintf(arq1, "%g\t%d\t%g\t%d\t%g\n", avg_value, min_index, min_value,
            max_index, max_value);
        fclose(arq1);
529     }
}
531
void keep_most_fit()
533 {
    int j, k;   //Loop counter for gene.
535
    /*Runs through all the genes*/
537     for (j = 0; j < al; ++j)
    {
539         for (k = 0; k < bt; ++k)
        {
541             //printf_s("Before %d ", GENE[min_index][j]);   //prints the gene
                from worst individual (for verification)
            BINARY[min_index][j][k] = BINARY[max_index][j][k];      //replaces it
                 with the gene from best individual
543             //printf_s("After %d\n", GENE[min_index][j]);   //and prints the
                modification (for verification).
        }
545     }
}
547
void recombination()
549 {
    int i, j, k;                //Loop couters.
551     int cut;                //Sets the point to split the individual genes for
        recombination.
    int sample_A, sample_B; //The numeric label of the individuals A and B to be
        recombined.
553
    int COPY[bt];           //To copy the individuals during the recombination.
555
    /*Runs through 'ps*rf' individuals*/
```

```
557     for (i = 0; i <= ps * rf; ++i)
        {
559         sample_A = random_integer(0, ps - 1);           //randomly choosing
                individual A
            //printf_s("Sample A is %u\n", sample_A);       //and displays its
                numeric label (for verification)
561         sample_B = random_integer(0, ps - 1);           //then randomly chooses
                individual B
            //printf_s("Sample B is %u\n", sample_B);       //and displays its
                numeric label (for verification).
563
            for (j = 0; j < al; ++j)
565         {
                cut = random_integer(0, bt - 1);                //Randomly chooses
                    the individuals split point
567             //printf_s("Cut at %u\n", cut);                 //and displays the
                    point (for verification).

569             /*Runs through each gene up to the 'cut' point.*/
                for (k = 0; k <= cut; ++k)
571             {
                    //printf_s("Sample A %u Sample B %u becomes\n", GENE[sample_A][j
                        ], GENE[sample_B][j]);//prints the initial situation (for
                        verification)
573                 COPY[k] = BINARY[sample_A][j][k];           //stores the copy A
                        gene
                    BINARY[sample_A][j][k] = BINARY[sample_B][j][k];    //replaces it
                         by the B gene
575                 BINARY[sample_B][j][k] = COPY[k];           //then replaces B
                        gene by the copy of A
                    //printf_s("Sample A %u Sample B %u\n", GENE[sample_A][j], GENE[
                        sample_B][j]);//and prints the final situation (for
                        verification).
577             }
            }
579     }
    }
581
    void mutation()
583 {
        int j;          //Loop counter.
585     int sample, gen, bit;//To select the individual and gene to be mutated.

587     /*Runs through 'ps*al*mf' genes from all individual*/
        for (j = 0; j <= (int)(ps * al * bt * mf); ++j)
589     {
            sample = random_integer(0, ps - 1);             //randomly choosing an
```

```
            individual
591         //printf("Sample %u ", sample);                 //printing the choise (
                for verification).
            gen = random_integer(0, al - 1);                //Then randomly chooses a
                 gene
593         //printf("gen %u\n", gen);                      //printing its position (
                for verification)
            bit = random_integer(0, bt - 1);
595         //printf("Gene %u", GENE[sample][gen]);         //and value (for
                verification).
            BINARY[sample][gen][bit] = random_integer(0, 1);        //Replaces the
                original gene value for a random one
597         //printf(" became %u\n", GENE[sample][gen]);    //and prints the mutated
                gene (for verification).
        }
599 }

601 double pulse(double in[], int indx)
    {
603     time_t first, second;

605     int c1, c2;

607     double SAmax, Stmax, SmaxAUX;

609     first = time(NULL); /* inicio da contagem de tempo do programa */

611     /* reiniciar */

613     GaussPulse(x, fx, (double)N, dt, Tp, Thp);
        Dessintonia(x, fx, (double)N, dt, D);
615     GVD(x, fx, (int)N, dt, 0.0);

617     for (c2 = 0; c2 < Nl; ++c2) {
            Staus[c2][0] = -0.4 + c2 * 0.025;
619         Staus[c2][1] = 0.0;
        }
621
        prop_lin2(x, fx, N, dt, in, indx);
623
        SAmax = -1000;
625     Stmax = -1000;
        for (c2 = 0; c2 < Nl; ++c2) {
627         //  printf_s("Staus = %g\n", Staus[c2][1]);
            SmaxAUX = (Soma - Staus[c2][1]) / Soma;
629         //  printf_s("SmaxAUX = %g\n", SmaxAUX);
            if (SAmax < SmaxAUX) {
```

```
631             SAmax = SmaxAUX;
                Stmax = Staus[c2][0];
633             //      printf_s("SAmax = %g\n", SAmax);
            }
635     }

637     //printf_s("SAmax = %g\n", SAmax);

639     /*Keeping the best results in files*/
        if ((Soma - Staus[16][1]) / Soma > max_mem)
641     {
            max_mem = (Soma - Staus[16][1]) / Soma;
643
            snprintf(fname, _MAX_PATH, "ArbitraryPhase_%d.dat", r);
645         fopen_s(&arq1, fname, "w");
            if (arq1 == 0)
647         {
                printf("ERRO EM 'arq1' !\n");
649             system("pause");
                exit(0);
651         }
            else
653         {
                for (c1 = 0; c1 < al; ++c1)
655             {
                    fprintf(arq1, "%g\n", GENE[max_index][c1]);
657             }
                fclose(arq1);
659         }

661         snprintf(fname, _MAX_PATH, "Alpha x Tau_%d.dat", r);
            fopen_s(&arq1, fname, "w");
663         if (arq1 == 0)
            {
665             printf("ERRO EM 'arq1' !\n");
                system("pause");
667             exit(0);
            }
669         else
            {
671             for (c2 = 0; c2 < Nl; ++c2)
                {
673                 fprintf(arq1, "%g\t%g\t%g\t%g\n", Staus[c2][0], Soma, Staus[c2
                        ][1], (Soma - Staus[c2][1]) / Soma);
                }
675             fclose(arq1);
            }
```

```
677     }

679     second = time(NULL); /* final da contagem de tempo do programa */
        printf("t de maquina: %f s\n", difftime(second, first));
681
        //return(SAmax); //returns the best possible result
683     return((Soma - Staus[16][1]) / Soma); //returns the result for tau = 0 ps
}
685
void prop_lin(double coord[], double data[], int nn, double dd)
687 {
        int c1;
689     //fcomplex fa1, fa2, fa3;
        double fa1R, fa2R, fa3R;
691     double fa1I, fa2I, fa3I;

693     FFT(coord, data, nn, dd);

695     Soma = 0.0;
        for (c1 = 0; c1 < nn; ++c1)
697     {
            fa1R = EAw[2 * c1];
699         fa1I = EAw[2 * c1 + 1];

701         fa2R = data[2 * c1];
            fa2I = data[2 * c1 + 1];
703
            fa3R = fa1R * fa2R - fa1I * fa2I;
705         fa3I = fa1R * fa2I + fa1I * fa2R;

707         Soma += (fa3R * fa3R + fa3I * fa3I) / (nn * dd);
        }
709     //printf("Soma = %g\n", Soma);
}
711

713 void prop_lin2(double coord[], double data[], int nn, double dd, double ipt[],
        int ind)
{
715

717     int c1, c2, c3, taun;

719     double a2, a3, a4;

721     double frs, fis;
        frs = 0;
```

```
723     fis = 0;

725     //fcomplex fa1, fa2, fa3, fa4, fa5, fa6, fa7, fa8, fa9, Ta;
        double fa1R, fa2R, fa3R, fa4R, fa5R, fa6R, fa7R, fa8R, fa9R, fa0R;
727     double fa1I, fa2I, fa3I, fa4I, fa5I, fa6I, fa7I, fa8I, fa9I, fa0I;

729     double* xa, * fxa;

731     xa = dvector(0, N);
        fxa = dvector(0, 2 * N);
733
        /*Procedure to apply the Phase Mask*/
735     FFT(coord, data, nn, dd);

737     for (c1 = 0; c1 < 128; ++c1) {
            MASK[0][c1] = c1 - 128.0 / 2.0;                 // Setting possible
                integer values ranging from -64 to 64
739         MASK[1][c1] = MASK[2][c1] = 0.0;                // Setting all else to 0
                to prevent allocation problems
        }
741
        for (c1 = 0; c1 < 128; ++c1)
743     {
            MASK[0][c1] = c1 - 128.0 / 2.0;                 // Setting possible
                integer values ranging from -64 to 64
745         MASK[1][c1] = MASK[2][c1] = 0.0;                // Setting all else to 0
                to prevent allocation problems
            MASK[1][c1] = ipt[c1];
747     }

749     a2 = 0.0;
        c3 = -1;                                                    // Variable used to
            keep track of the bit section in the following loop:
751     for (c1 = 0; c1 < nn; ++c1)
        {
753         if (fabs(coord[c1]) <= 64)
            {
755             c2 = (int)floor(coord[c1]) + 64;
                if (c2 < 128 && c2 >= 0)
757             {
                    /* Smooth Phase Mask */
759                 if (fabs(coord[c1] - floor(coord[c1])) <= gratres / 2 || fabs(
                        coord[c1] - floor(coord[c1])) >= 1 - gratres / 2)
                    {
761                     /* Left Side */
                        if (fabs(coord[c1] - floor(coord[c1])) <= gratres / 2)
763                     {
```

```
                        /* Border Exception */
765                     if (floor(coord[c1]) == -64)
                        {
767                         a2 = MASK[1][c2] / (1 + exp(-lk * (coord[c1] - floor(
                                coord[c1]))));

769                         data[2 * c1] = data[2 * c1] / (1 + exp(-lk * (coord[
                                c1] - floor(coord[c1]))));
                            data[2 * c1 + 1] = data[2 * c1 + 1] / (1 + exp(-lk *
                                (coord[c1] - floor(coord[c1]))));
771                     }
                        else
773                     {
                            a2 = MASK[1][c2 - 1] + (MASK[1][c2] - MASK[1][c2 -
                                1]) / (1 + exp(-lk * (coord[c1] - floor(coord[c1
                                ]))));
775                     }
                    }
777
                    /* Right Side */
779                 if (fabs(coord[c1] - floor(coord[c1])) >= 1 - gratres / 2)
                    {
781                     /* Border Exception */
                        if (floor(coord[c1]) == 63)
783                     {
                            a2 = MASK[1][c2] - MASK[1][c2] / (1 + exp(-lk * (
                                coord[c1] - (floor(coord[c1] + 1)))));
785
                            data[2 * c1] = data[2 * c1] - data[2 * c1] / (1 + exp
                                (-lk * (coord[c1] - (floor(coord[c1] + 1)))));
787                         data[2 * c1 + 1] = data[2 * c1 + 1] - data[2 * c1 +
                                1] / (1 + exp(-lk * (coord[c1] - (floor(coord[c1]
                                 + 1)))));
                        }
789                     else
                        {
791                         if (c2 < 127)
                            {
793                             a2 = MASK[1][c2] + (MASK[1][c2 + 1] - MASK[1][c2
                                    ]) / (1 + exp(-lk * (coord[c1] - (floor(coord
                                    [c1] + 1)))));
                            }
795                     }
                    }
797             }
                else
799             {
```

```
                    a2 = MASK[1][c2];                                   // Store the
                        angular phase diference,
801                 }
                }
803             else
                {
805                 a2 = 0.0;
                }
807         }
            else
809         {
                a2 = 0.0;
811             if (floor(coord[c1]) == -65 || floor(coord[c1]) == 64)
                {
813                 if (floor(coord[c1]) == -65)
                    {
815                     if (fabs(coord[c1] - floor(coord[c1])) >= 1 - gratres / 2)
                        {
817                         a2 = MASK[1][0] / (1 + exp(-lk * (coord[c1] - (floor(
                                coord[c1] + 1)))));

819                         data[2 * c1] = data[2 * c1] / (1 + exp(-lk * (coord[c1] -
                                 (floor(coord[c1] + 1)))));
                            data[2 * c1 + 1] = data[2 * c1 + 1] / (1 + exp(-lk * (
                                coord[c1] - (floor(coord[c1] + 1)))));
821                     }
                        else
823                     {
                            data[2 * c1] = 0;
825                         data[2 * c1 + 1] = 0;
                        }
827                 }

829                 if (floor(coord[c1]) == 64)
                    {
831                     if (fabs(coord[c1] - floor(coord[c1])) <= gratres / 2)
                        {
833                         a2 = MASK[1][127] - MASK[1][127] / (1 + exp(-lk * (coord[
                                c1] - floor(coord[c1]))));

835                         data[2 * c1] = data[2 * c1] - data[2 * c1] / (1 + exp(-lk
                                 * (coord[c1] - floor(coord[c1]))));
                            data[2 * c1 + 1] = data[2 * c1 + 1] - data[2 * c1 + 1] /
                                (1 + exp(-lk * (coord[c1] - floor(coord[c1]))));
837                     }
                        else
839                     {
```

```
                        data[2 * c1] = 0;
841                     data[2 * c1 + 1] = 0;
                    }
843             }
            }
845         else
            {
847             data[2 * c1] = 0;
                data[2 * c1 + 1] = 0;
849         }
        }
851
        a3 = data[2 * c1];                              // the field real part
853     a4 = data[2 * c1 + 1];                          // and also the field
            imaginary part.
        data[2 * c1] = a3 * cos(a2) - a4 * sin(a2);     // Then, apply a complex
            rotation using the angular phase diference to obtain the resulting
            field
855     data[2 * c1 + 1] = a3 * sin(a2) + a4 * cos(a2);

857 }

859 /*Integral with modified pulse*/
    for (c1 = 0; c1 < nn; ++c1)
861 {
        //printf("%d\t%g\t%g\n", c1, coord[c1], (Soma - Staus[39][1]) / Soma);
863     fa1R = data[2 * c1];
        fa1I = data[2 * c1 + 1];
865
        fa2R = fxc[2 * c1];
867     fa2I = fxc[2 * c1 + 1];

869     fa6R = EAw[2 * c1];
        fa6I = EAw[2 * c1 + 1];
871
        fa7R = fa6R * fa1R - fa6I * fa1I;
873     fa7I = fa6R * fa1I + fa6I * fa1R;


875
        for (c2 = 0; c2 < nn; ++c2)
877     {
            fa3R = data[2 * c2];
879         fa3I = data[2 * c2 + 1];

881         Tw1w2Bmod(c1, c2);

883
```

```
            fa0R = fa2R * TaR - fa2I * TaI;
            fa0I = fa2R * TaI + fa2I * TaR;

            fa4R = fa3R * fa0R - fa3I * fa0I;
            fa4I = fa3R * fa0I + fa3I * fa0R;

            xa[c2] = coord[c2];
            fxa[2 * c2] = fa4R;
            fxa[2 * c2 + 1] = fa4I;

        }

        IFFT(xa, fxa, nn, 1 / (nn * dd));

        for (c2 = 0; c2 < Nl; ++c2)
        {
            taun = (int)(N / 2) + (int)(Staus[c2][0] / dd);
            fa3R = 0.0;
            fa3I = 0.0;

            for (c3 = 0; c3 < taun; ++c3)
            {
                fa3R += fxa[2 * c3];
                fa3I += fxa[2 * c3 + 1];
            }

            fa9R = dd * fa3R;
            fa9I = dd * fa3I;

            fa4R = cos(coord[c1] * Staus[c2][0]);
            fa4I = sin(coord[c1] * Staus[c2][0]);

            fa0R = fa4R * fa9R - fa4I * fa9I;
            fa0I = fa4R * fa9I + fa4I * fa9R;

            fa5R = 0.5 * Thc * fa0R;
            fa5I = 0.5 * Thc * fa0I;

            fa8R = fa7R + fa5R;
            fa8I = fa7I + fa5I;

            Staus[c2][1] += (fa8R * fa8R + fa8I * fa8I) / (nn * dd);
        }
    }

    free_dvector(xa, 0, N);
    free_dvector(fxa, 0, 2 * N);
```

```
931 }

933 void GaussPulse(double coord[], double data[], double nn, double dd, double width
        , double area)
    {
935     double A, a1;
        int c1;
937
        A = width * sqrt(doispi / (2 * 1.38)); /* area do pulso com largura total a
            meia altura de width */
939
        for (c1 = 0; c1 < nn; ++c1)
941     {
            coord[c1] = (c1 - nn / 2) * dd;          /* escala temporal */
943         a1 = coord[c1] / width;
            data[2 * c1] = area * exp(-1.38 * a1 * a1) / A;      /* parte real do
                campo */
945         data[2 * c1 + 1] = 0;                           /* parte imaginaria do
                campo */
        }
947 }

949 void Dessintonia(double coord[], double data[], double nn, double dd, double Dd)
    {
951     int c1;
        //fcomplex fa1, fa2, fa3;
953     double fa1R, fa2R, fa3R;
        double fa1I, fa2I, fa3I;
955
        for (c1 = 0; c1 < nn; ++c1)
957     {
            fa1R = data[2 * c1];
959         fa1I = data[2 * c1 + 1];

961         fa2R = cos(Dd * coord[c1]);
            fa2I = sin(Dd * coord[c1]);
963
            fa3R = fa1R * fa2R - fa1I * fa2I;
965         fa3I = fa1R * fa2I + fa1I * fa2R;

967         data[2 * c1] = fa3R;
            data[2 * c1 + 1] = fa3I;
969     }
    }
971
    void GVD(double coord[], double data[], int nn, double dd, double L)
973 {
```

```
    /* propagacao atraves de SQ1 com lambda de 800 nm */

    int c1;
    double a2, a4, a5, k2, w1;

    FFT(coord, data, nn, dd);

    k2 = L * Tp * Tp / 2.0;  /* beta2 */

    for (c1 = 0; c1 < nn; ++c1) {
        w1 = coord[c1];
        a2 = k2 * (w1 * w1 / 2.0);
        a4 = data[2 * c1];
        a5 = data[2 * c1 + 1];
        data[2 * c1] = a4 * cos(a2) + a5 * sin(a2);
        data[2 * c1 + 1] = -a4 * sin(a2) + a5 * cos(a2);
    }

    IFFT(coord, data, nn, 1 / (nn * dd));
}

void nrerror(char error_text[])
{
    fprintf(stderr, "Numerical Recipes run-time error...\n");
    fprintf(stderr, "%s\n", error_text);
    fprintf(stderr, "...now exiting to system...\n");
    exit(1);
}

double* dvector(int nl, int nh)
/* Allocates a double vector with range [nl..nh] */
{
    double* v;
    v = (double*)malloc(static_cast<int64_t>((unsigned)(nh - nl + 1)) * sizeof(
        double));
    if (!v) nrerror((char*)"allocation failure in dvector()");
    return v - nl;
}

void free_dvector(double* v, int nl, int nh)
/* Frees a double vector allocated by dvector() */
{
    free((char*)(v + nl));
}

void four1(double data[], int nn, int isign)
{
```

```
    int n, mmax, m, j, istep, i;
    double wtemp, wr, wpr, wpi, wi, theta;
    double tempr, tempi;

    n = nn << 1;
    j = 1;
    for (i = 1; i < n; i += 2) {
        if (j > i) {
            SWAP(data[j], data[i]);
            SWAP(data[j + 1], data[i + 1]);
        }
        m = n >> 1;
        while (m >= 2 && j > m) {
            j -= m;
            m >>= 1;
        }
        j += m;
    }
    mmax = 2;
    while (n > mmax) {
        istep = mmax << 1;
        theta = isign * (6.28318530717959 / mmax);
        wtemp = sin(0.5 * theta);
        wpr = -2.0 * wtemp * wtemp;
        wpi = sin(theta);
        wr = 1.0;
        wi = 0.0;
        for (m = 1; m < mmax; m += 2) {
            for (i = m; i <= n; i += istep) {
                j = i + mmax;
                tempr = wr * data[j] - wi * data[j + 1];
                tempi = wr * data[j + 1] + wi * data[j];
                data[j] = data[i] - tempr;
                data[j + 1] = data[i + 1] - tempi;
                data[i] += tempr;
                data[i + 1] += tempi;
            }
            wr = (wtemp = wr) * wpr - wi * wpi + wr;
            wi = wi * wpr + wtemp * wpi + wi;
        }
        mmax = istep;
    }
}

void FFT(double coord[], double data[], int nn, double dd)
{
    int c1;
```

```
    double  tempr;

    for (c1 = 0; c1 < nn; c1 += 2) {
        SWAP(data[c1], data[c1 + nn]);
        SWAP(data[c1 + 1], data[c1 + 1 + nn]);
    }

    four1(data - 1, nn, 1);

    for (c1 = 0; c1 < nn; c1 += 2) {
        SWAP(data[c1], data[c1 + nn]);
        SWAP(data[c1 + 1], data[c1 + 1 + nn]);
    }
    for (c1 = 0; c1 < nn; c1 += 1) {
        coord[c1] = 2 * 3.14159265 * ((double)c1 - (double)nn / 2) / (nn * dd);
        data[2 * c1] *= dd;
        data[2 * c1 + 1] *= dd;
    }
}

void IFFT(double coord[], double data[], int nn, double dd)
{
    int c1;
    double  tempr;

    for (c1 = 0; c1 < nn; c1 += 2) {
        SWAP(data[c1], data[c1 + nn]);
        SWAP(data[c1 + 1], data[c1 + 1 + nn]);
    }

    four1(data - 1, nn, -1);

    for (c1 = 0; c1 < nn; c1 += 2) {
        SWAP(data[c1], data[c1 + nn]);
        SWAP(data[c1 + 1], data[c1 + 1 + nn]);
    }

    for (c1 = 0; c1 < nn; c1 += 1) {
        coord[c1] = ((double)c1 - (double)nn / 2) / (nn * dd);
        data[2 * c1] *= dd;
        data[2 * c1 + 1] *= dd;
    }
}

void Aw(double w)
{
    //fcomplex  a;
```

```
    double a1;

    a1 = 1.0 + w * w * T2 * T2;
    //a.r = 1.0 / a1;
    AwR = 1.0 / a1;
    //a.i = w * T2 / a1;
    AwI = w * T2 / a1;

    //return(a);
}

void Tw1w2Bmod(int c1, int c2) {
    //fcomplex fa1, fa2, fa3, fa4, fa5, fa6, fb1, fb2, fb3, fb4, fb5, fb6;
    double  fa1R, fa2R, fa3R, fa4R, fa5R, fa6R, fb1R, fb2R, fb3R, fb4R, fb5R,
        fb6R, fb0R;
    double  fa1I, fa2I, fa3I, fa4I, fa5I, fa6I, fb1I, fb2I, fb3I, fb4I, fb5I,
        fb6I, fb0I;

    fa1R = FAw[2 * c1];
    fa1I = FAw[2 * c1 + 1];

    fa2R = FAw[2 * c2];
    fa2I = FAw[2 * c2 + 1];

    fa3R = fa1R - fa2R;
    fa3I = fa1I - fa2I;

    fb3R = ODm * fa3R;
    fb3I = ODm * fa3I;

    fb4R = exp(fb3R) * cos(fb3I);
    fb4I = exp(fb3R) * sin(fb3I);

    fb5R = exp(-fb3R) * cos(fb3I);
    fb5I = -exp(-fb3R) * sin(fb3I);

    fb6R = fb4R - fb5R;
    fb6I = fb4I - fb5I;

    fb1R = fa1R + fa2R;
    fb1I = fa1I + fa2I;

    fb2R = (-OD / 2.0) * fb1R;
    fb2I = (-OD / 2.0) * fb1I;

    fa4R = exp(fb2R) * cos(fb2I);
    fa4I = exp(fb2R) * sin(fb2I);
```

```
1159
        fa6R = fa4R * fb6R - fa4I * fb6I;
1161    fa6I = fa4R * fb6I + fa4I * fb6R;

1163    if (sqrt(fa3R * fa3R + fa3I * fa3I) > (1E-10))
        {
1165        fb0R = (fa6R * fa3R + fa6I * fa3I) / (fa3R * fa3R + fa3I * fa3I);
            fb0I = (fa6I * fa3R - fa6R * fa3I) / (fa3R * fa3R + fa3I * fa3I);
1167
            fa5R = fa1R * fb0R - fa1I * fb0I;
1169        fa5I = fa1R * fb0I + fa1I * fb0R;
        }
1171
        else
1173    {
            fb0R = 2.0 * ODm * fa4R;
1175        fb0I = 2.0 * ODm * fa4I;

1177        fa5R = fa1R * fb0R - fa1I * fb0I;
            fa5I = fa1R * fb0I + fa1I * fb0R;
1179    }

1181    TaR = fa5R;
        TaI = fa5I;
1183 }
```