Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Engenharia Elétrica
Disciplina: Comunicações Ópticas

Comparação entre o SSFM simétrico e assimétrico com e sem passo adaptativo

Aluno: Lucas De Oliveira Lobo

Campina Grande, 28 de junho de 2023.

1. Introdução

Nos últimos anos, a comunicação óptica vem desempenhando um papel fundamental no desenvolvimento de redes de alta velocidade e transmissão de dados. Nesse contexto, técnicas eficientes de processamento de sinais são essenciais para melhorar o desempenho e a confiabilidade dos sistemas de comunicações ópticas.

$$\frac{\partial A}{\partial z} = -\frac{i}{2}\beta_2 \frac{\partial^2 A}{\partial t^2} - \frac{\alpha}{2}A + i\gamma|A|^2 A$$

Figura 1: Equação de propagação na fibra, sendo destacada a parcela linear (equação em azul, e a parcela não linear (equação em vermelho).

O método Split Step Fourier Method é uma técnica amplamente utilizada para modelar e simular sistemas de comunicação óptica. Ele permite a análise da propagação de sinais ópticos em fibras através da discretização do domínio espacial, permitindo a simulação de fenômenos complexos, como a dispersão cromática, a não-linearidade e a atenuação.

Uma característica importante do método de divisão do passo de Fourier é a possibilidade de utilização de duas variantes distintas: o método simétrico e o método não-simétrico, e com passo fixo ou passo adaptativo

1.1 Método não simétrico

O método consiste em dividir a equação diferencial em dois passos principais: o passo linear e o passo não linear. No passo linear, é realizada uma transformada de Fourier para obter a solução no domínio da frequência. Em seguida, é aplicado um operador linear no domínio da frequência para evoluir a solução no tempo. No passo não linear, a solução é transformada novamente para o domínio do espaço sendo aplicada uma operação não linear para considerar os termos não-lineares presentes na

equação original. Esse processo é repetido iterativamente até que a solução convirja para a solução desejada.

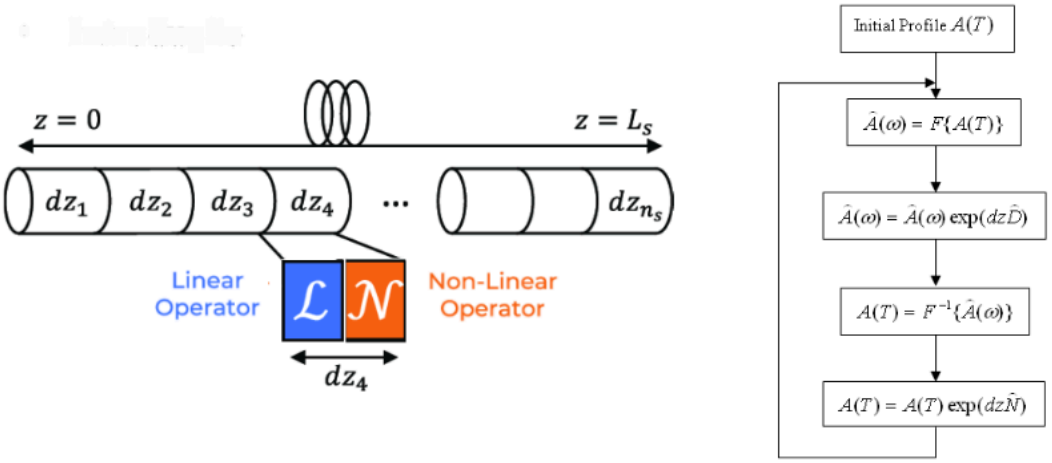


Figura 2: Representação esquemática do método não simétrico assim como o diagrama de blocos para sua implementação

1.2 Método simétrico

Esse método é bastante semelhante ao não simétrico, porém, a componente linear de cada iteração é subdivida em dois grupos, para ser calculada tanto antes quanto após o cálculo da parcela não linear, a ideia é que com isso se diminua os erros causados pela separação das duas parcelas da equação.

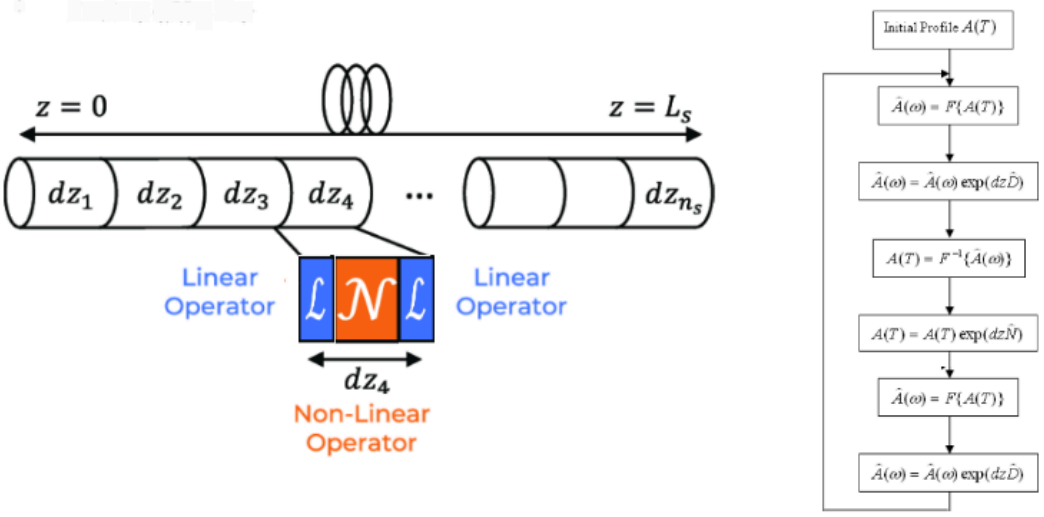


Figura 3: Representação esquemática do método simétrico assim como o diagrama de blocos para sua implementação

1.3 Passo adaptativo

É conhecido que em cálculo numérico o erro é um função do tamanho do passo utilizado (h), logo, existe sempre uma tentativa de balancear precisão e tempo de simulação, a técnica de passo adaptativo é uma forma de cálculo automática e iterativa de um tamanho de passo de forma que minimize o tempo de simulação mantendo o erro dentro de um valor tolerado.

Para isso é necessário se estimar o valor do erro em uma iteração, para isso, foi usado como calculo do erro a diferença entre duas simulações: uma com 1 iteração de passo de tamanho normal, e outra resultante de 2 iterações com tamanho/2. de forma que:

$$y_{n+1}^{(1)} - y_{n+1}^{(0)} = \tau_{n+1}^{(1)}$$

Onde o sufixo (0) denota o primeiro resultado, e (1) o último.

Calculando, esse erro é possível estimar iterativamente uma melhor aproximação para o próximo passo:

$$h \to 0.9 \times h \times \min\left(\max\left(\left(\frac{\text{tol}}{2\left|\tau_{n+1}^{(1)}\right|}\right)^{1/2}, 0.3\right), 2\right)$$

e caso o erro seja menor que a tolerância, estimar um melhor valor para y, que será utilizado como base para a próxima iteração:

$$y_{n+1}^{(2)} = y_{n+1}^{(1)} + \tau_{n+1}^{(1)}$$

2. Objetivo

Neste trabalho, será realizado um estudo comparativo entre o método simétrico e o método não-simétrico do Split Step Fourier Method, no contexto de comunicações ópticas. O objetivo é analisar as vantagens e desvantagens de cada abordagem em termos de precisão e eficiência computacional, considerando diferentes cenários de transmissão e parâmetros do sistema. Além de implementar e discutir as melhorias de um código utilizando o passo adaptativo.

3. Metodologia

Nesse trabalho foi realizada 3 simulações distintas, com 2 variáveis:

1. O tamanho do enlace de fibra, que impacta diretamente no número de iterações necessárias para se alcançar uma determinada precisão.
2. O número de bits transmitidos na simulação, aumentando o tempo de processamento necessário para se computar as transformadas e inversas de Fourier.

Tendo esses parâmetros em mente foi montado um conjunto de 3 simulações:

1. Propagação de um sinal de 1000 bits em um enlace de 100 quilômetros.
2. Propagação de um sinal de 1000 bits em um enlace de 200 quilômetros.
3. Propagação de um sinal de 2000 bits em um enlace de 100 quilômetros.

Os parâmetros da fibra, sinal e simulação foram os seguintes:

$$\alpha = 0 \ \frac{dB}{Km}$$

$$(A\ atenuação\ foi\ desconsiderada\ para\ maximizar\ os\ efeitos\ não\ lineares)$$

$$D = 18e^{-6} \ \frac{ps}{nm * km}$$

$$\gamma = 1.3e^{-3} \ \frac{W^{-1}}{m}$$

$$\lambda = 1550nm$$

$$Modulação = OOK$$

$$Rs = 10e^{9}$$

$$Sps = 32$$

$$Potencia = 10 \ dBm$$

As metrificas utilizadas para avaliar cada algoritmo foi o tempo necessário para sua execução, assim como o Root Mean Squared Error (RMSE) que está sendo usado para medir a diferença do algoritmo para uma dada precisão em relação a um resultado obtido com uma simulação de altíssima precisão, além do erro utilizado pelo algoritmo de passo adaptativo.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2}$$

Figura 4: Equação utilizada para o cálculo do RMSE

Por fim, os códigos desenvolvidos para se implementar o passo adaptativo nos dois métodos foram os seguintes:

```
def Adaptative_SSF_non_symmetric(E, hz, Lspan, alpha, gamma, D, Fc,
Fs,tol):
    λ  = c/Fc
    α  = 1e-3*alpha/(10*np.log10(np.exp(1)))
    β2 = -(D*λ**2)/(2*np.pi*c)

    Nfft = len(E)
    ω = 2*np.pi*Fs*np.fft.fftfreq(Nfft)

    z = 0
    erro = 100000
    n = 0
    hz1 = hz
    while z <= Lspan:
        while(erro>tol):
            # Operador linear
            hz = hz1
            E = np.fft.fft(E)
            E0 = E*np.exp(-α*hz+1j*(β2/2)*(ω**2)*hz)
            E1 = E*np.exp(-α*hz/2+1j*(β2/2)*(ω**2)*hz/2)

            # Operador não linear
            E0 = np.fft.ifft(E0)
            E1 = np.fft.ifft(E1)
            E0 = E0*np.exp(1j*gamma*(np.abs(E0)**2)*hz)
            E1 = E1*np.exp(1j*gamma*(np.abs(E1)**2)*hz/2)

            E1 = np.fft.fft(E1)
            E1 = E1*np.exp(-α*hz/2+1j*(β2/2)*(ω**2)*hz/2)
            E1 = np.fft.ifft(E1)
            E1 = E1*np.exp(1j*gamma*(np.abs(E1)**2)*hz/2)

            erro = RMSE(E1,E0)
            hz1 = 0.9*hz*min(max((tol/(2*erro))**0.5,0.3),2)
            n += 1
            E = np.fft.ifft(E)
        E = 2*E1 - E0
        z += hz
        erro = 100000
    return E,n
```

```
def Adaptative_SSF_symmetric(E, hz, Lspan, alpha, gamma, D, Fc,
Fs,tol):
    λ  = c/Fc
    α  = 1e-3*alpha/(10*np.log10(np.exp(1)))
    β2 = -(D*λ**2)/(2*np.pi*c)

    Nfft = len(E)

    ω = 2*np.pi*Fs*np.fft.fftfreq(Nfft)
    z = 0

    E = np.fft.fft(E)
    erro = 1000
    n = 0
    hz1 = hz
    while z <= Lspan:
        while(erro>tol):
            hz = hz1
            # Primeiro passo - operador linear
            E0 = E*np.exp(-α*(hz/2)+1j*(β2/2)*(ω**2)*(hz/2))
            E1 = E*np.exp(-α*(hz/4)+1j*(β2/2)*(ω**2)*(hz/4))
            # Operador não linear
            E0 = np.fft.ifft(E0)
            E1 = np.fft.ifft(E1)
            E0 = E0*np.exp(1j*gamma*(np.abs(E0)**2)*hz)
            E1 = E1*np.exp(1j*gamma*(np.abs(E1)**2)*hz/2)

            # Segundo passo - operador linear
            E0 = np.fft.fft(E0)
            E1 = np.fft.fft(E1)
            E0 = E0*np.exp(-α*(hz/2)+1j*(β2/2)*(ω**2)*(hz/2))
            E1 = E1*np.exp(-α*(hz/4)+1j*(β2/2)*(ω**2)*(hz/4))

            #2° loop de E1
            E1 = E1*np.exp(-α*(hz/4)+1j*(β2/2)*(ω**2)*(hz/4))
            E1 = np.fft.ifft(E1)
            E1 = E1*np.exp(1j*gamma*(np.abs(E1)**2)*hz/2)
            E1 = np.fft.fft(E1)
            E1 = E1*np.exp(-α*(hz/4)+1j*(β2/2)*(ω**2)*(hz/4))
```

```
            erro = RMSE(E1,E0)
            hz1 = 0.9*hz*min(max((tol/(2*erro))**0.5,0.3),2)
            n += 1
        E = 2*E1 - E0
        z += hz
        erro = 100000
    E = np.fft.ifft(E)
    return E,n
```

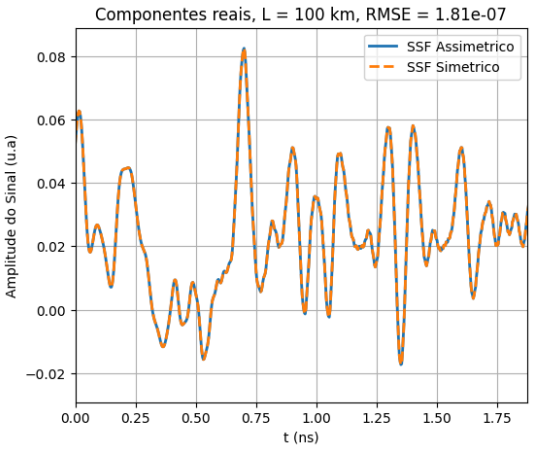
4. Resultados

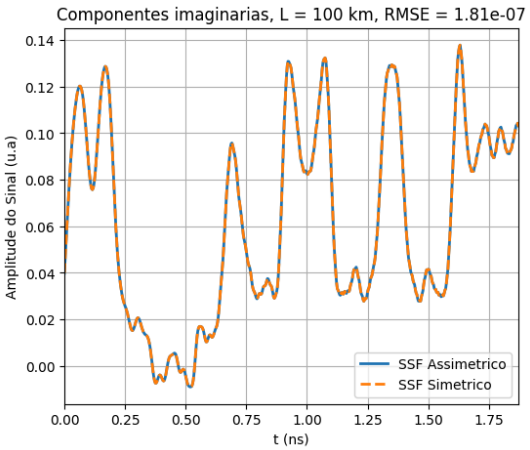
4.1 Propagação de um sinal de 1000 bits em um enlace de 100 quilômetros

4.1.1 Passo estático

Nesse primeiro caso a simulação foi feita com um total de 100000 iterações, garantindo um passo de 1 metro.

O sinal usado como base para o cálculo do erro possuiu o seguinte formato:





Fazendo diversas simulações com diferentes precisões podemos extrair os seguintes gráficos:

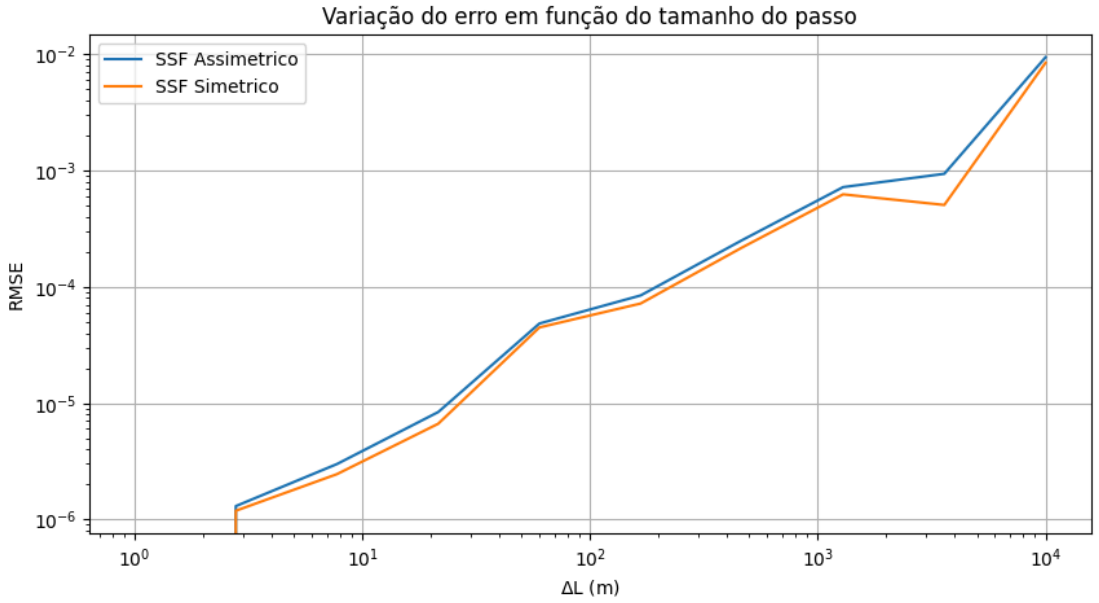


Figura 5: Gráfico da variação do erro em função da precisão.

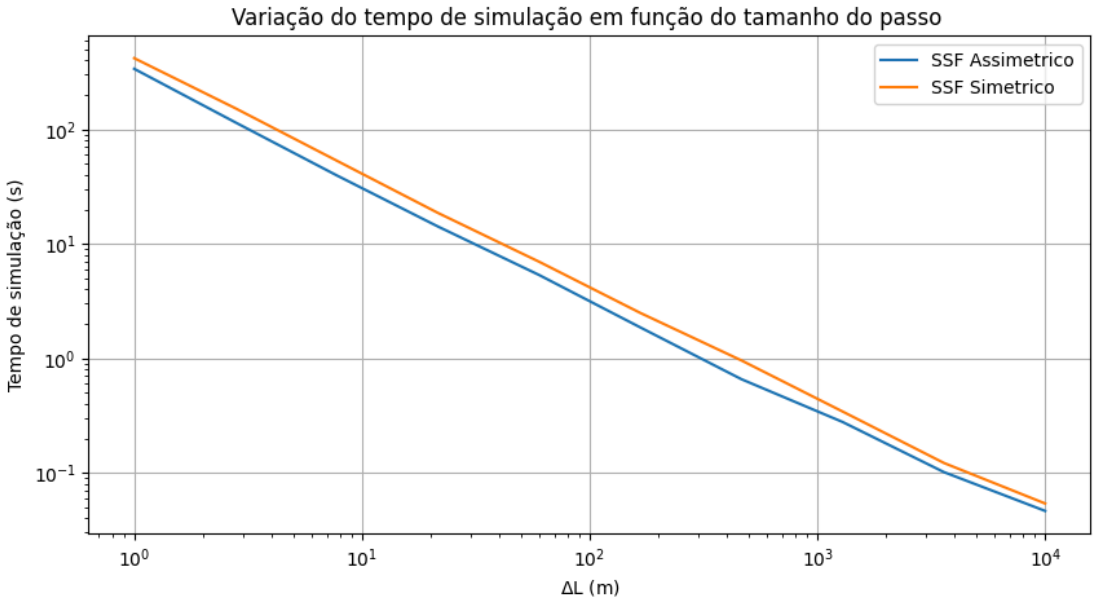


Figura 6: Gráfico da variação do erro em função da precisão.

Unindo as informações obtidas em ambos os gráficos, é possível se criar uma nova função que cruza a precisão da simulação com o tempo de execução do código:

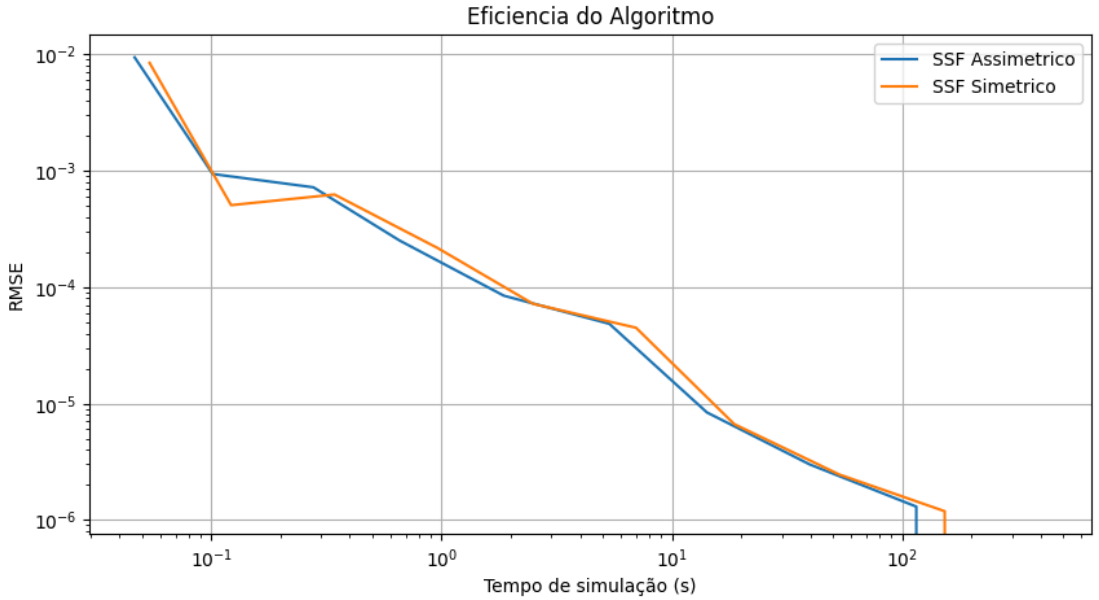
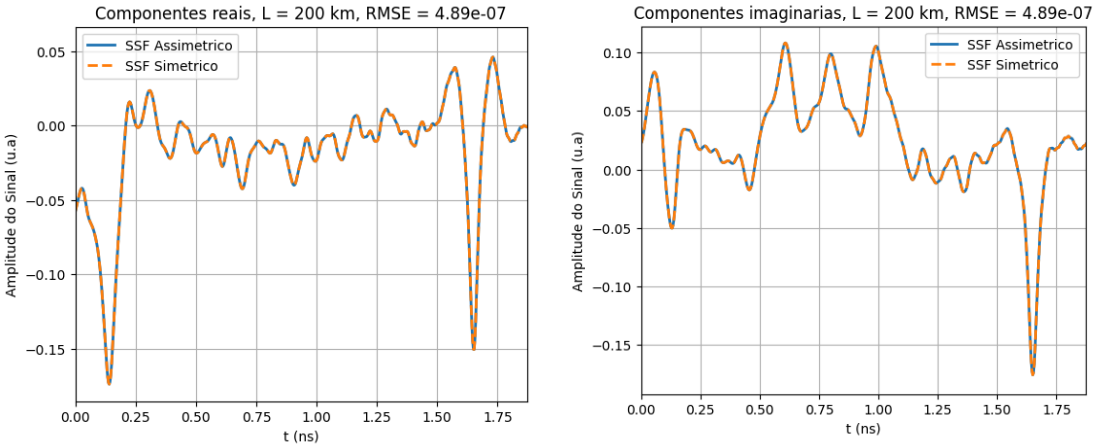


Figura 7: Gráfico da variação do erro em função do tempo de simulação.

4.2 Propagação de um sinal de 1000 bits em um enlace de 200 quilômetros

A mesma metodologia foi aplicada nessa simulação, porém o número de iterações precisou ser dobrado para manter o passo do cálculo consistente, o objetivo era analisar se o aumento do comprimento da simulação poderia causar mudanças nos resultados.



Obtendo os seguintes gráficos:

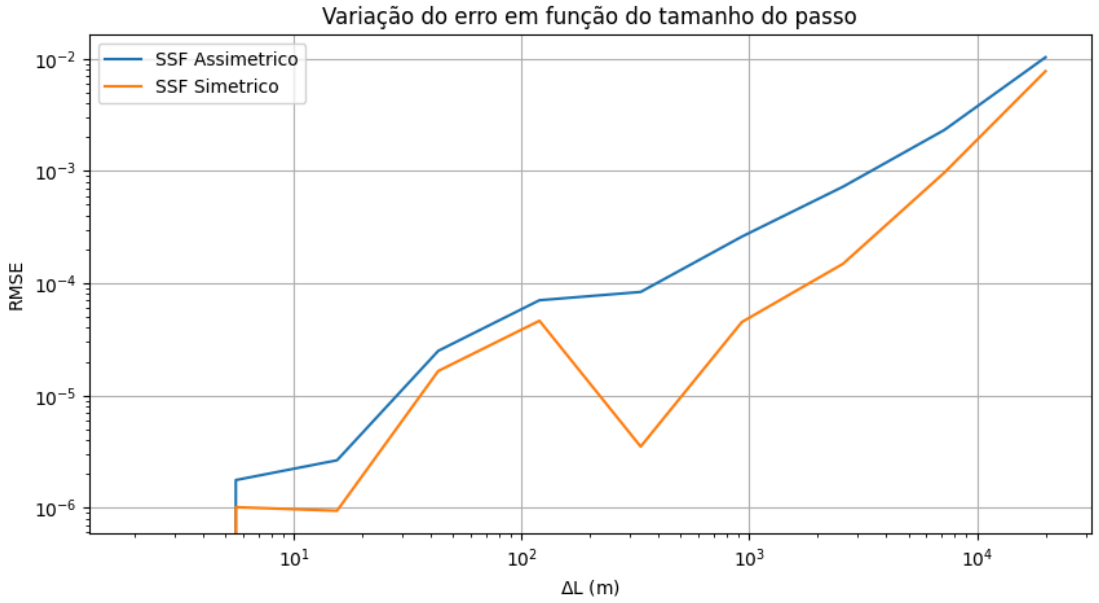


Figura 8: Gráfico da variação do erro em função da precisão.

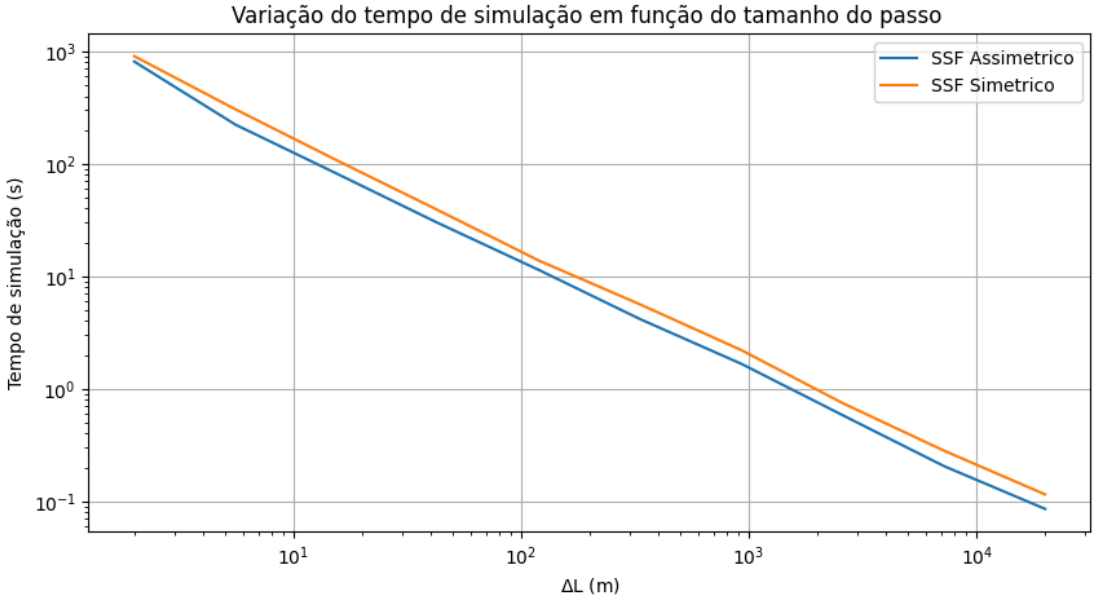


Figura 9: Gráfico da variação do erro em função da precisão.

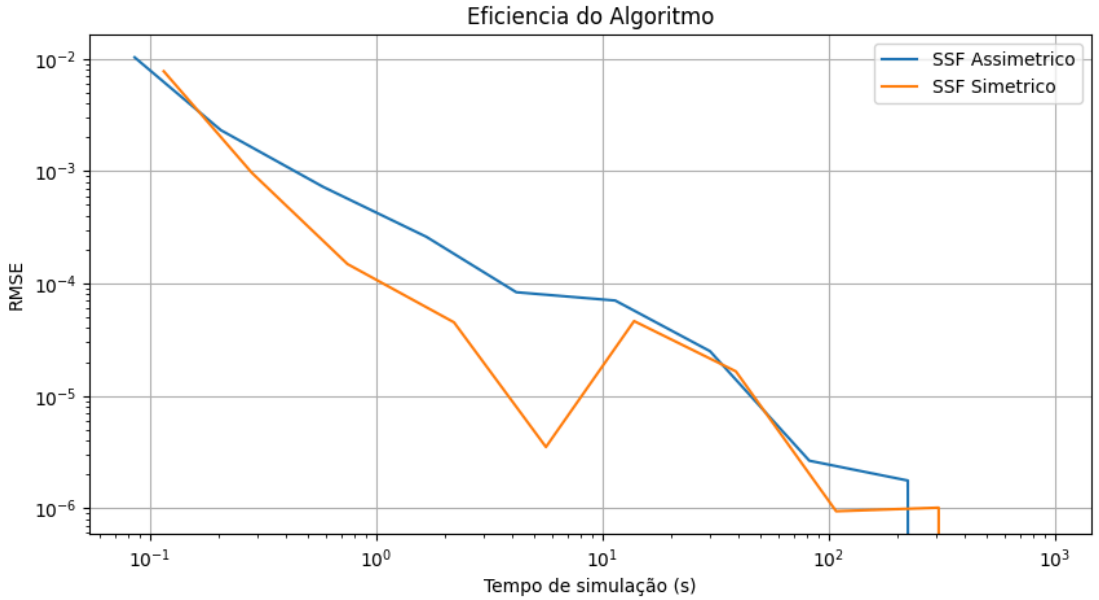
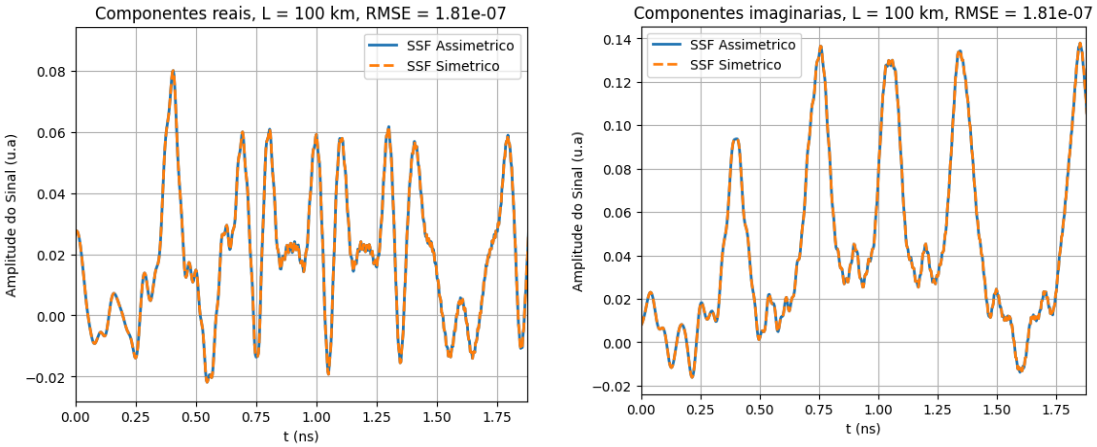


Figura 10: Gráfico da variação do erro em função do tempo de simulação.

4.3 Propagação de um sinal de 2000 bits em um enlace de 100 quilômetros

A mesma metodologia foi aplicada nessa simulação, porém o número de bits foi dobrado, e por consequência, o tamanho dos dados que precisavam ser operados, para analisar como mudanças no tempo de simulação base poderiam afetar a relação erro x tempo.



Obtendo os seguintes gráficos:

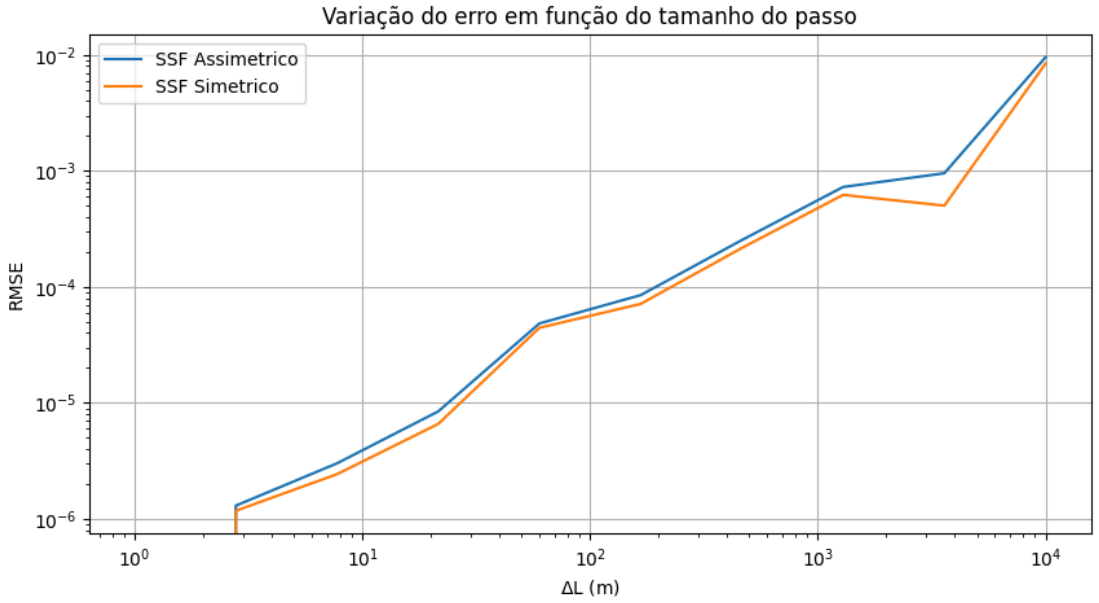


Figura 11: Gráfico da variação do erro em função da precisão.

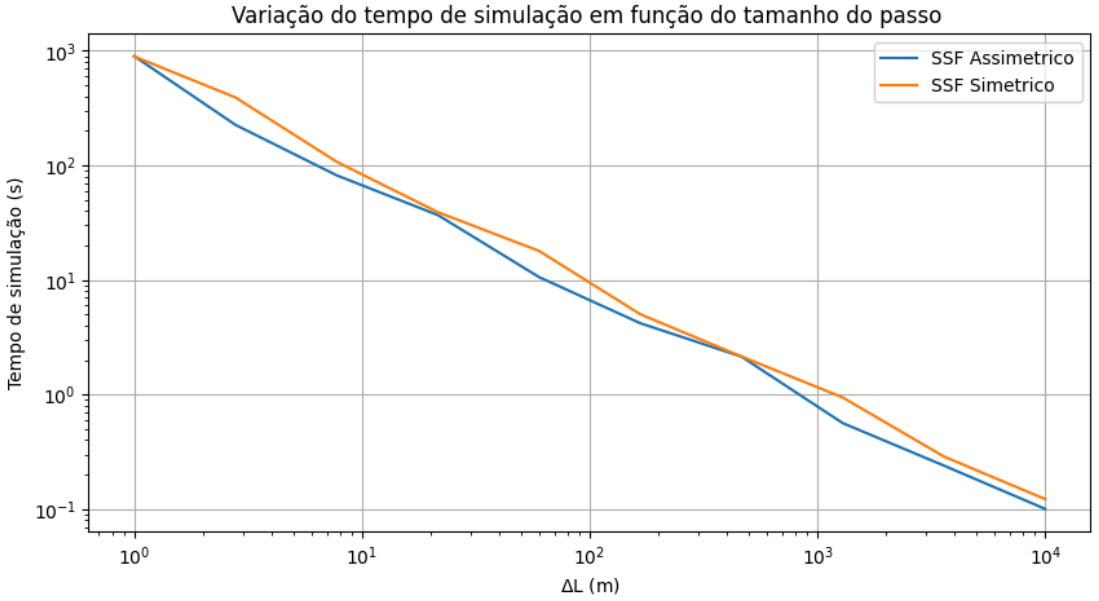


Figura 12: Gráfico da variação do erro em função da precisão.

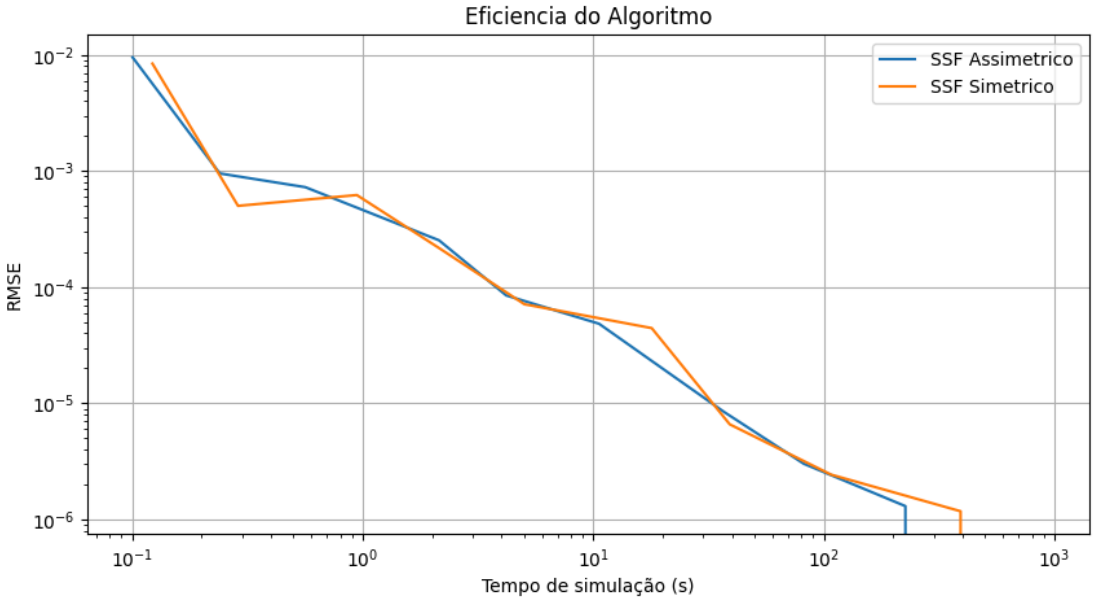


Figura 13: Gráfico da variação do erro em função do tempo de simulação.

4.4 Passo adaptativo

4.4.1 Método Assimétrico

Como tolerância, foi utilizado o RMSE entre as simulações de referência =1.81e-7, e o tamanho de passo inicial foi mantido como 1 metro.

Após a simulação foi constatado que o erro final entre a simulação com passo adaptativo e o de referência foi de 1.81e-6, o tempo de simulação foi de 20.47 segundos e o número de iterações totais foi de 752.

Visualizando os resultados é possível ver que são extremamente semelhantes:

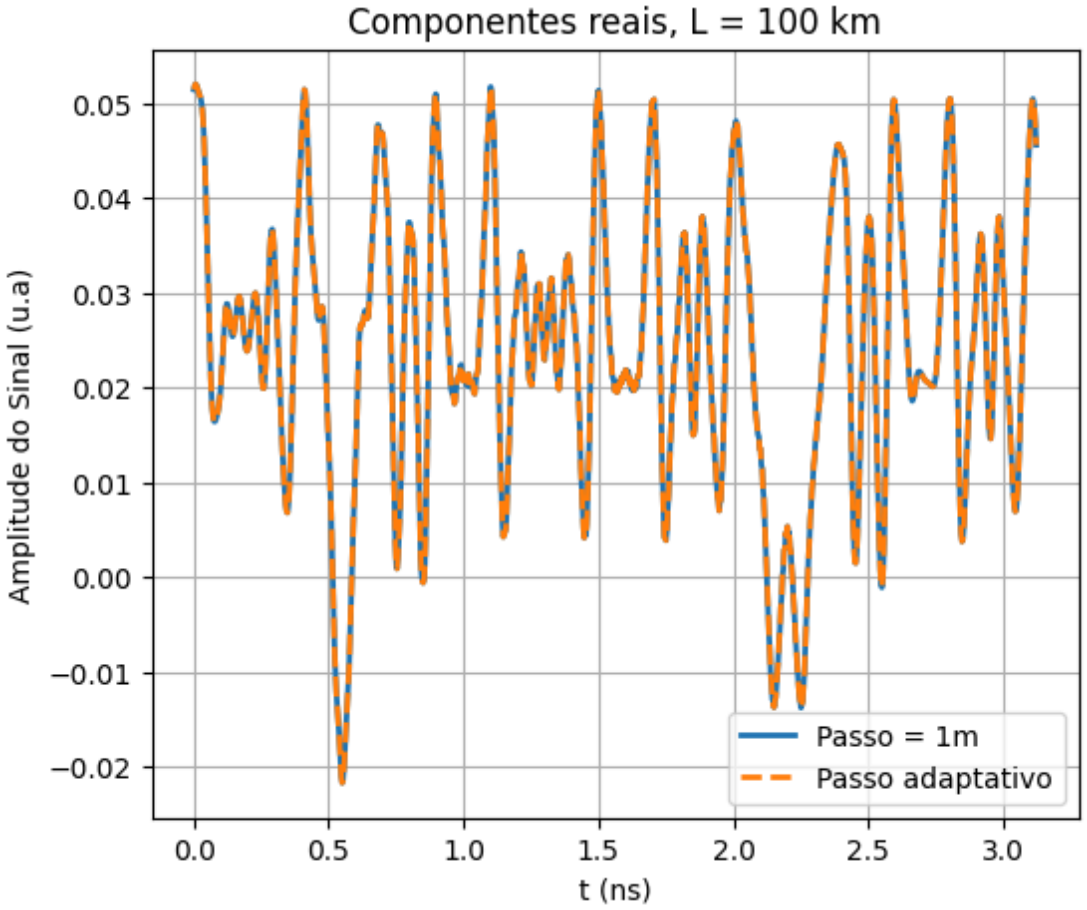


Figura 14: Análise dos resultados no tempo

Podemos também analisar como o tamanho do passo se comportou durante a simulação:

Figura 15: Análise do tamanho do passo

4.4.1 Método simétrico

Como tolerância, foi utilizado o RMSE entre as simulações de referência =1.81e-7, e o tamanho de passo inicial foi mantido como 1 metro.

Após a simulação foi constatado que o erro final entre a simulação com passo adaptativo e o de referência foi de 1.43e-6, o tempo de simulação foi de 43.45 segundos e o número de iterações totais foi de 1479.

Visualizando os resultados é possível ver que são extremamente semelhantes:

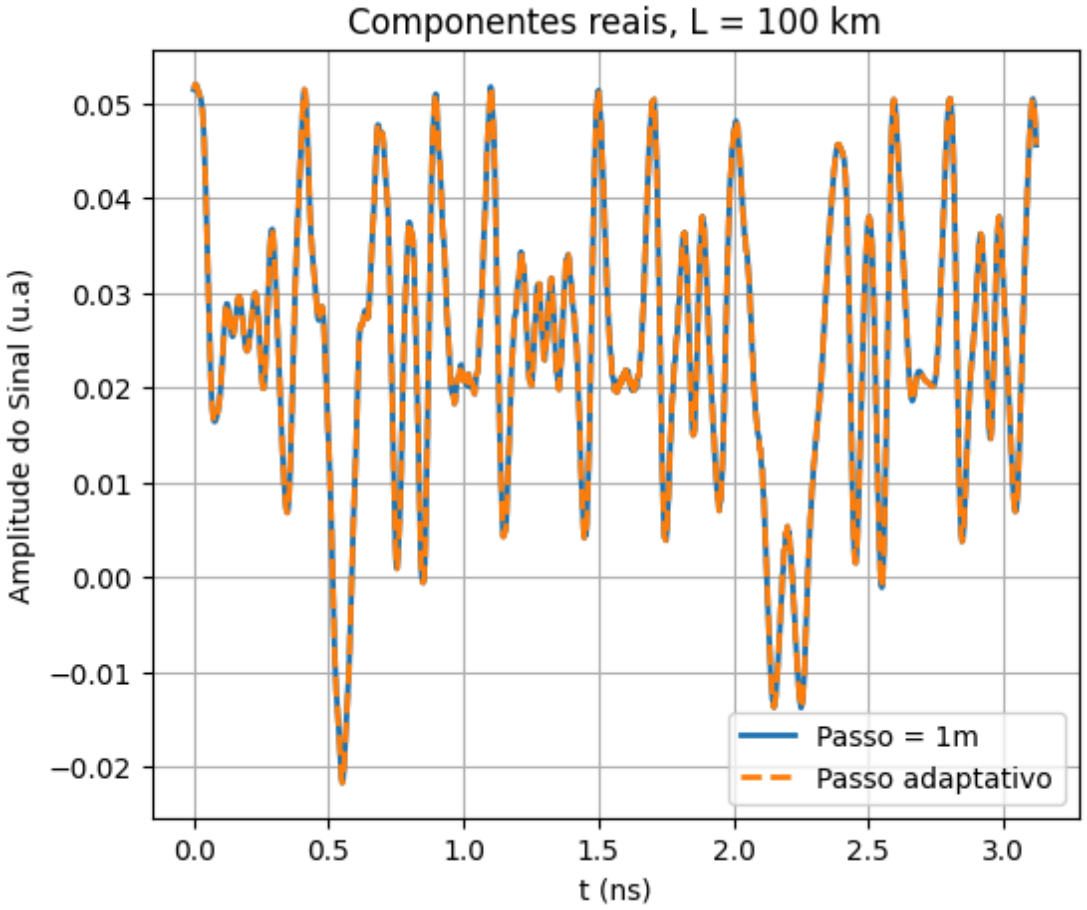


Figura 16: Análise dos resultados no tempo

Podemos também analisar como o tamanho do passo se comportou durante a simulação:

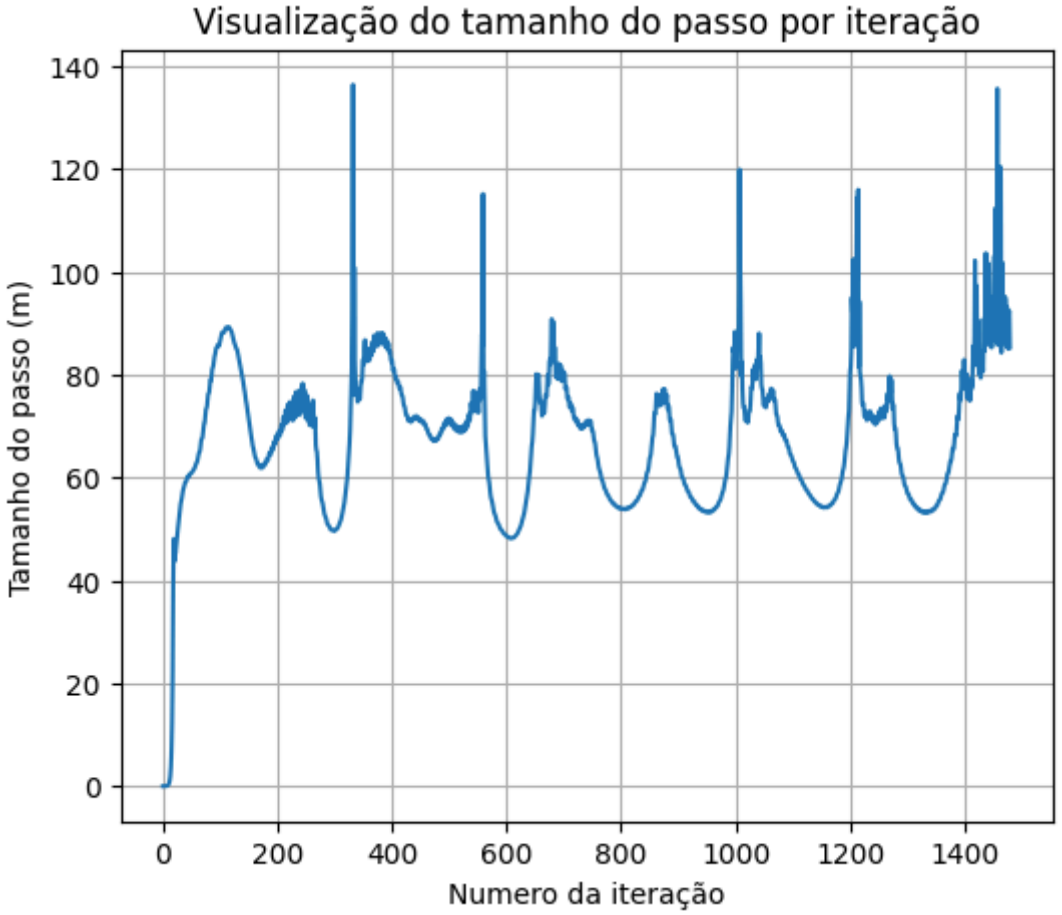


Figura 17: Análise do tamanho do passo

5 Conclusão

Comparando os resultados com passo estático, é possível ver que a medida que o comprimento L da simulação aumenta, a relação anterior permaneceu constante, apenas escalonando o tempo necessário por um fator de 2x.

Aumentar o número de bits simulados, por outro lado, fez com que a diferença entre a eficiência dos dois métodos diminuísse, sugerindo que talvez para maiores simulações o método não simétrico seja mais útil, devido à menor quantidade de operações realizadas.

Implementar o código com passo adaptativo trouxe uma grande melhoria no tempo de simulação, sem introduzir erro significativos, como pode ser visto pela seguinte tabela:

Assimétrico	Erro (RMSE)	Número de iterações	Tempo de Simulação (s)
Passo estático	1.81e-7	100000	20.47
Passo adaptativo	1.81e-6	752	~350

Simétrico	Erro (RMSE)	Número de iterações	Tempo de Simulação (s)
Passo estático	1.43e-6	100000	43.45
Passo adaptativo	1.81e-7	1479	~450

É importante ressaltar que em implementações reais, outras medidas devem ser tomadas para tornar essas simulações as mais eficientes possíveis. (Utilizar uma linguagem de programação como C, ou CUDA pode trazer melhorias significativas na velocidade de simulação)