Universidade Federal do Paraná
Departamento de Engenharia Elétrica

Pedro Lucca Pereira da Veiga
Adriely Teixeira de Paula

Processamento Digitais de Sinais II
Laboratório 2 - Aplicação de Filtros Adaptativos de Wiener e LMS

Curitiba
2025

1 Exercício 1

O arquivo *Ex_Wiener1.m* contém uma rotina de remoção de ruído de uma imagem através da aplicação de um filtro de Wiener bidimensional. Execute a rotina para a imagem do arquivo *Lenna.bmp*, conforme o código fornecido e realize as tarefas abaixo:

1.1 a. A rotina de aplicação do filtro de Wiener possui o protótipo: *wiener2(sinal, dimensão, var. de ruído)*. Altere a dimensão do filtro de 2x2 a 8x8 e avalie a consequência na imagem restaurada.

A imagem antes e após a adição de ruído (com variância igual a $0,025$) pode ser observada abaixo:

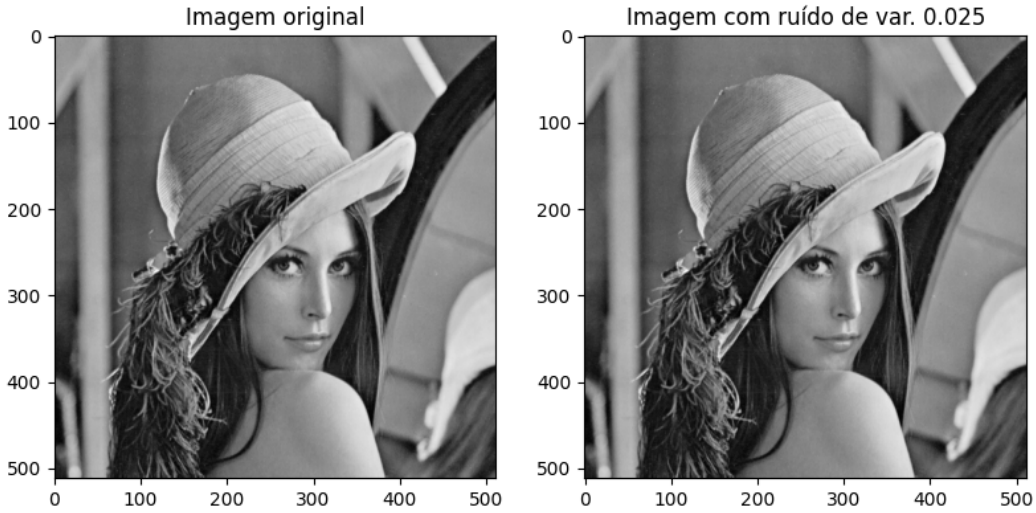

Figure 1: Imagem antes (esquerda) e após (direita) a adição de ruído branco gaussiano

Os resultados do algoritmo para diferentes tamanhos do *kernel* podem ser observados abaixo:

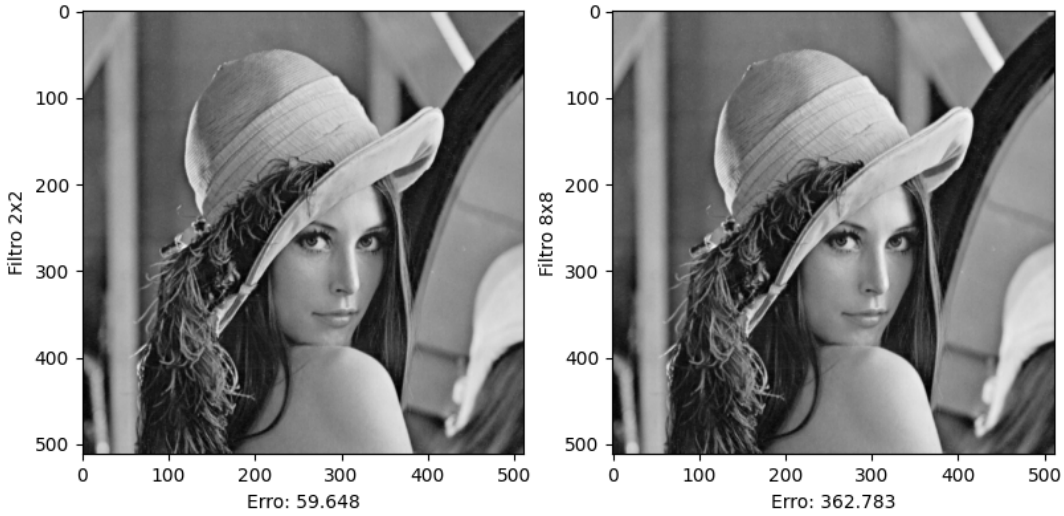

Figure 2: Resultados para filtros de diferentes tamanhos

Observa-se que o valor de erro obtido através do filtro 8x8 é maior, uma vez que um *kernel* de tamanho maior tende a "aproximar" mais as informações contidas na imagem.

1.2 b. Mantendo a dimensão do filtro fixa em 5x5, altere a variância do ruído de $0,015$ a $0,085$, em passos de $0,01$ e avalie os resultados nas imagens processadas.

Os resultados obtidos da utilização do filtro 5x5, para diferentes valores de variância de ruído, podem ser observados abaixo:

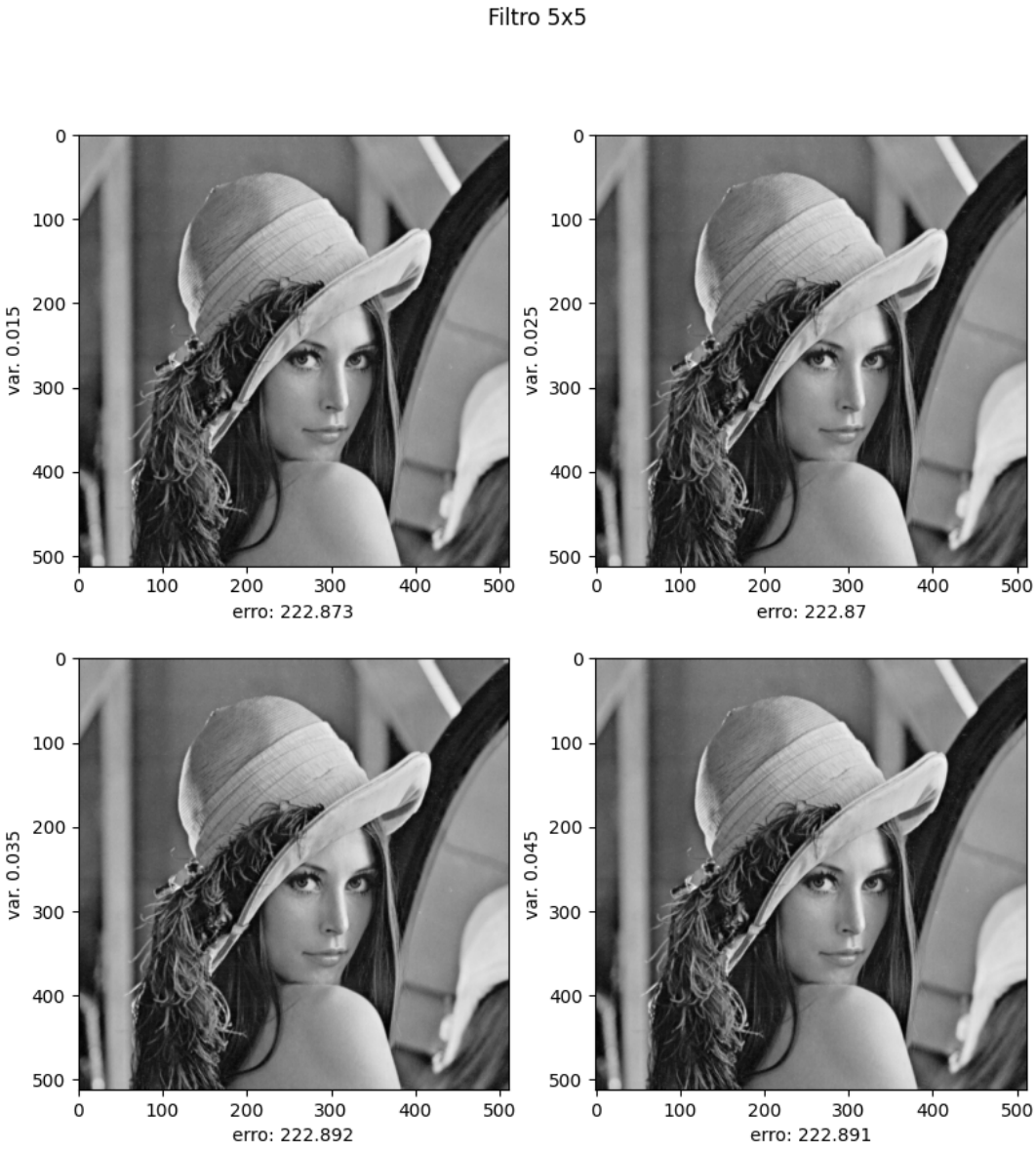

Figure 3: Utilização do filtro 5x5: variância de $0,015$ a $0,045$

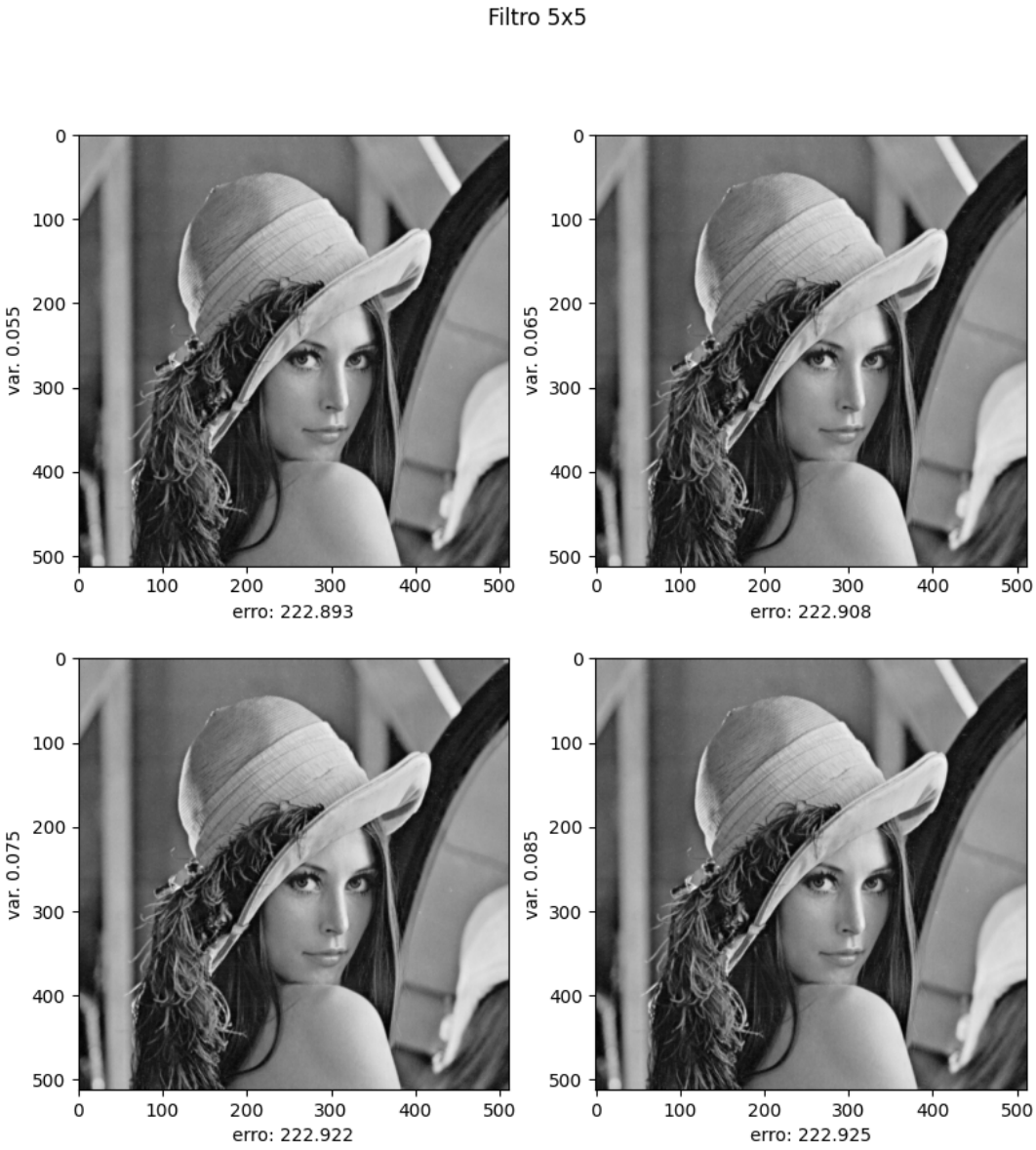

Figure 4: Utilização do filtro 5x5: variância de $0,055$ a $0,085$

1.3 c. A rotina que efetivamente aplica o filtro de Wiener está descrita no arquivo *wiener2.m*. Descreva o funcionamento desta rotina baseado no código Octave/Matlab

O filtro de Wiener implementado no código possui duas

2 Exercício 2

O arquivo *Ex_Wiener2.m* contém uma rotina de ajuste de borramento e remoção de ruído de uma imagem através da aplicação da deconvolução de Wiener. Neste caso são avaliados tanto a adição de ruído gaussiano quanto a existência de ruído de quantização. Execute a rotina para a imagem do arquivo *Lenna.bmp*, conforme o código fornecido e realize as tarefas abaixo:

Pelo código fornecido, tem-se as seguintes imagens:

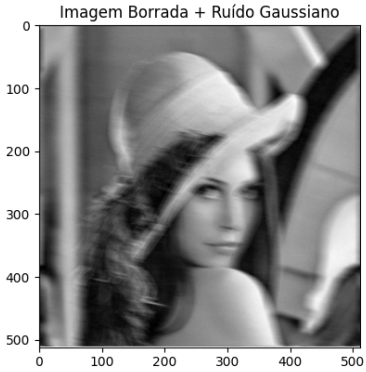

Figure 5: Imagem Borrada + Ruído Gaussiano

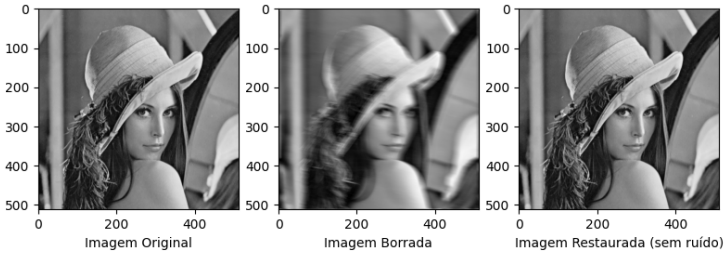

Figure 6: Imagem Original, Imagem Borrada e Imagem Restaurada

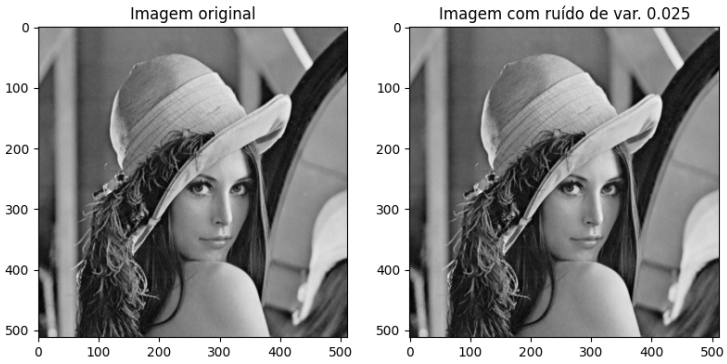

Figure 7: Imagem Original e Imagem com Ruído de Var. 0.025

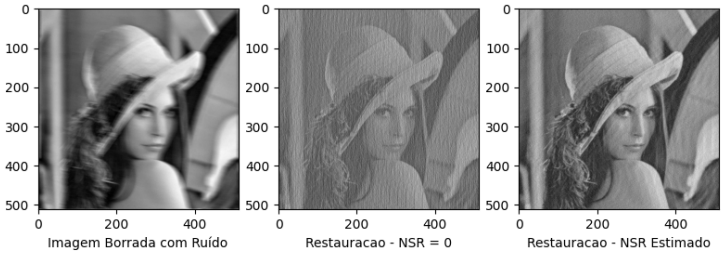

Figure 8: Imagem Borrada com Ruído, com Restauração (NSR=0) e com Restauração (NSR Estimado)

2.1 a. É fácil verificar que o algoritmo não apresenta bons resultados quando não há informação da distribuição do ruído. Entretanto, esta informação nem sempre está disponível. Assim, mantenha a variância do ruído em $0,0001$ e altere em 50% o valor da variância do ruído na aplicação do filtro (na função *deconvwnr*) para mais e para menos, em passos de 10% e avalie os resultados obtidos. Faça esta operação para a imagem com ruído gaussiano adicionado.

Os resultados obtidos para os diferentes níveis de variância de ruído podem ser observados abaixo:

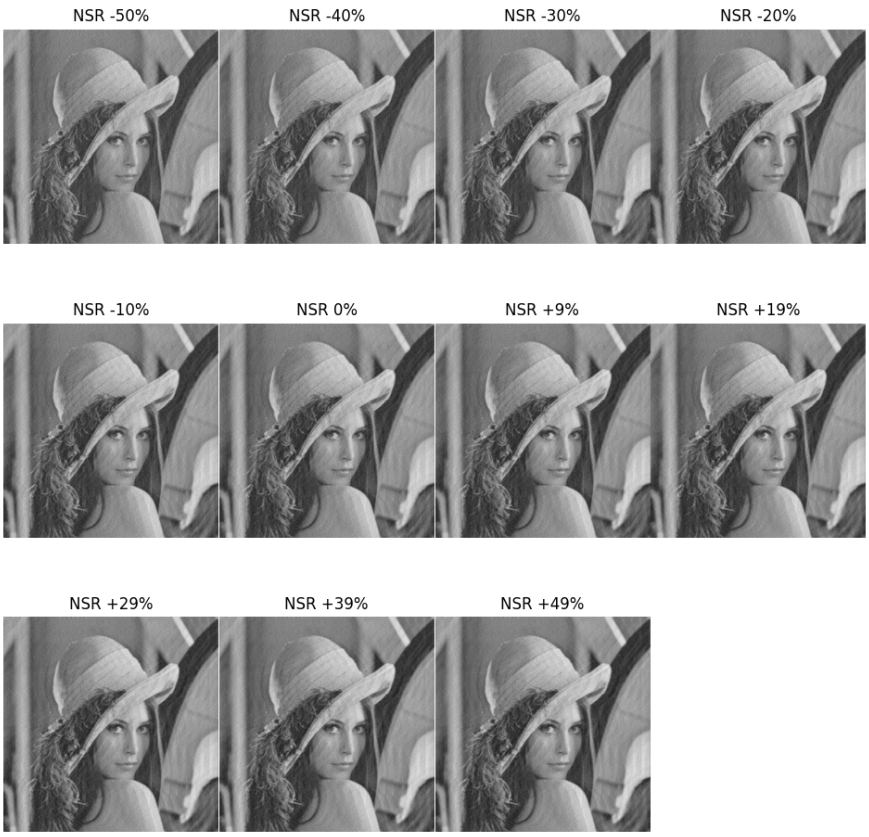

Figure 9: Resultados para diferentes valores de NSR

Quando o valor do NSR (relação ruído/sinal) é muito pequeno em relação ao ruído real presente na imagem, o algoritmo de deconvolução tenta recuperar excessivamente detalhes que, na verdade, são apenas ruído. Como consequência, surgem muitos "defeitos" na imagem restaurada, prejudicando a qualidade visual.

Por outro lado, se o valor do NSR for considerado muito maior do que o ruído real existente, o algoritmo se torna excessivamente conservador na restauração. Nesse caso, ele suaviza demais a imagem, resultando em um aspecto borrado e perda de detalhes importantes.

Assim, a escolha correta do valor do NSR é fundamental para o bom desempenho da deconvolução. Um valor de NSR adequado proporciona um equilíbrio entre remoção do desfoque e controle do ruído, garantindo a melhor qualidade possível na imagem restaurada.

2.2 b. A rotina que efetivamente aplica a deconvolução de Wiener está descrita no arquivo *deconvwnr.m*. Descreva o funcionamento desta rotina baseado no código Octave/Matlab.

A função *deconvwnr()* corrige o desfoque (devido ao PSF - *point spread function*) e o ruído (baseado na estimativa do NSR - *noise-to-signal ratio*). A rotina filtra no domínio da frequência, usando a divisão espectral controlada.

O método é baseado em encontrar um meio-termo ideal entre "desfocar de volta" e "não amplificar o ruído".

3 Exercício 3

O arquivo *lms.m* apresenta um exemplo de aplicação do algoritmo LMS para a composição de um filtro adaptativo para eliminação de ruído de um sinal. Execute a rotina com o código fornecido e realize as tarefas abaixo:

3.1 a. Analise o código e descreva o funcionamento do algoritmo LMS.

O algoritmo LMS tenta ajustar os pesos do filtro adaptativamente para minimizar a diferença entre a saída desejada (d) e a saída do filtro estimado (y_{est}).

O vetor de entrada é uma janela de *sysorder* amostras do sinal de entrada *inp*. A diferença entre a saída desejada e a saída estimada é calculada para retornar o erro.

Os pesos do filtro são ajustados para minimizar esse erro, usando um termo proporcional ao erro e a entrada, multiplicado pela taxa de aprendizado μ. O algoritmo converge quando os pesos se ajustam para aproximar a saída estimada da saída desejada.

O LMS é eficiente para aplicações em tempo real porque realiza os cálculos de forma iterativa e local, sem a necessidade de calcular inversões de matrizes, como outros algoritmos adaptativos.

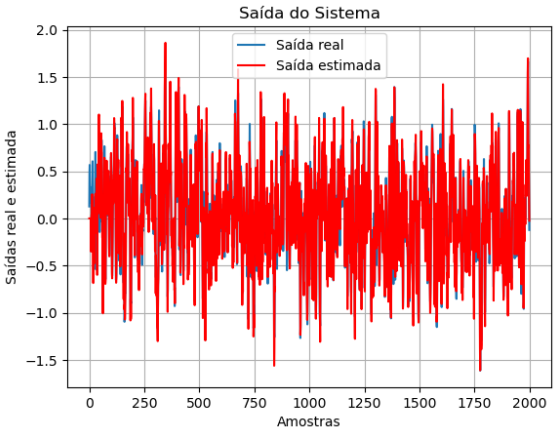

Figure 10: Gráfico da Saída do Sistema

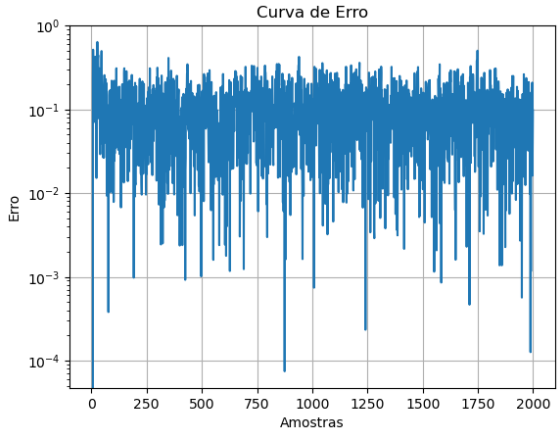

Figure 11: Curva de Erro

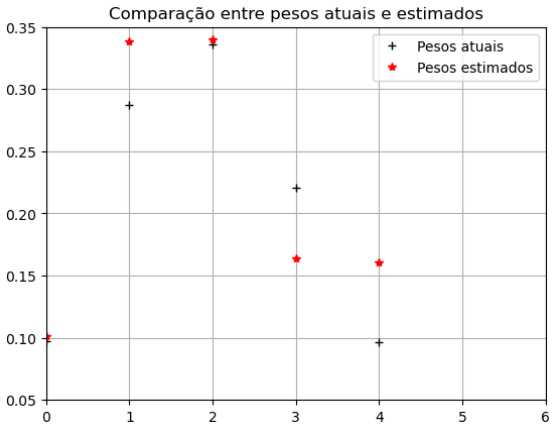

Figure 12: Gráfico Comparação Entre Pesos Atuais e Estimados

3.2 b. Varie a taxa de convergência mu até encontrar o ponto de perda de estabilidade do algoritmo. Qual este valor? Qual a relação dele com o sinal de entrada? Como o erro evolui com o crescimento da variável mu?

A taxa de convergência μ é um parâmetro crucial no algoritmo LMS, pois controla a velocidade de adaptação dos pesos. Valores muito altos de μ podem levar à instabilidade, enquanto valores muito baixos podem resultar em uma convergência lenta.

O valor de μ também pode ser influenciado pelas características do sinal de entrada, em especial pela variância do sinal. Se o sinal de entrada tiver uma alta variância, a adaptação dos pesos precisa ser mais cautelosa (usando um valor menor de μ) para evitar instabilidade.

Para μ muito pequeno, o erro pode diminuir de forma muito lenta. À medida que μ aumenta, o erro pode começar a diminuir mais rapidamente, mas pode even-

tualmente se estabilizar em um valor não ideal, ou até aumentar, caso o valor de μ se torne grande demais. Em relação a instabilidade, quando μ é muito grande, o erro pode aumentar, já que os pesos começam a "oscilar" ao invés de convergir para os valores ideais.

Os resultados obtidos para o erro de uma simulação do algoritmo podem ser observados abaixo:

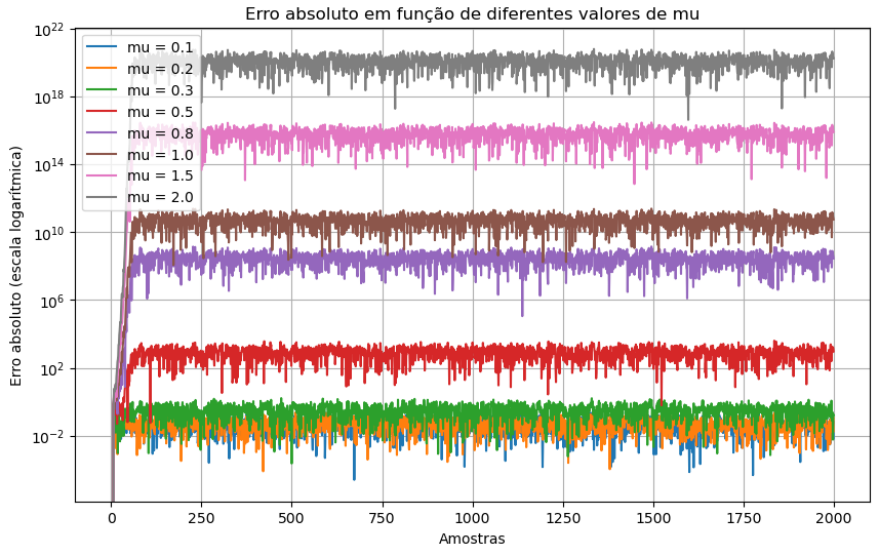

Figure 13: Gráfico do Erro Absoluto com Diferentes Valores de μ

De acordo com o gráfico, conforme o valor de μ aumenta, o erro absoluto também cresce. O valor crítico de μ (limite de estabilidade) é aproximadamente entre 0.8 e 1.0. Para $\mu > 1$, o algoritmo claramente perde a estabilidade.

3.3 c. Modifique o algoritmo para que o sinal de entrada seja o fornecido no arquivo de áudio *flauta.wav*. Adicione ruído gaussiano ao arquivo de áudio, com média 0 e variância 0,0250. Aplique o algoritmo LMS para composição do filtro que irá retirar o ruído, ajustando o passo do algoritmo (mu). Comente os resultados obtidos na filtragem.

O gráfico 14 mostra o sinal original, o sinal com ruído e o sinal filtrado. O algoritmo LMS consegue reduzir o ruído de forma eficaz, mas pode haver algumas distorções no sinal filtrado, especialmente para valores maiores de μ.

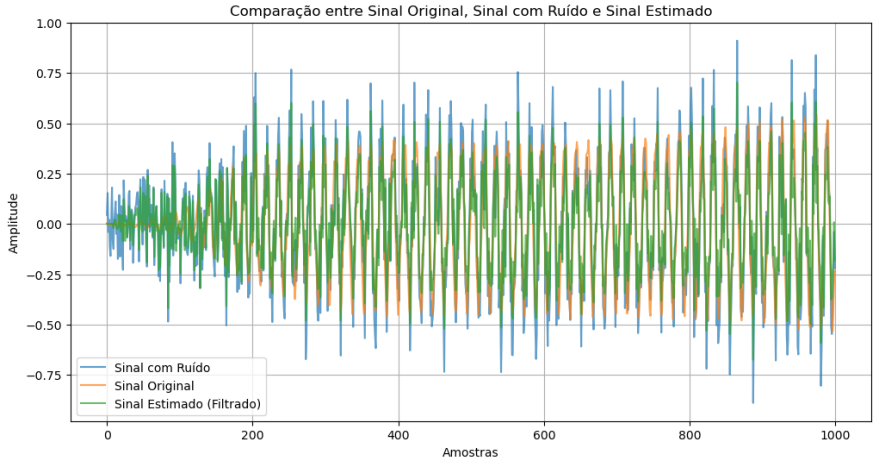

Figure 14: Comparação entre Sinal Original, com Ruído e Estimado

Quando μ é muito pequeno, o erro diminui lentamente. Com o aumento de μ, o erro diminui mais rapidamente no começo, mas pode começar a aumentar para valores muito grandes de μ, devido à instabilidade do filtro.

O áudio filtrado tem uma boa qualidade, com o ruído reduzido. No entanto, valores muito altos de μ podem gerar instabilidade e distorções no áudio, o que é visível no gráfico de erro 15 e pode ser percebido ao ouvir o áudio.

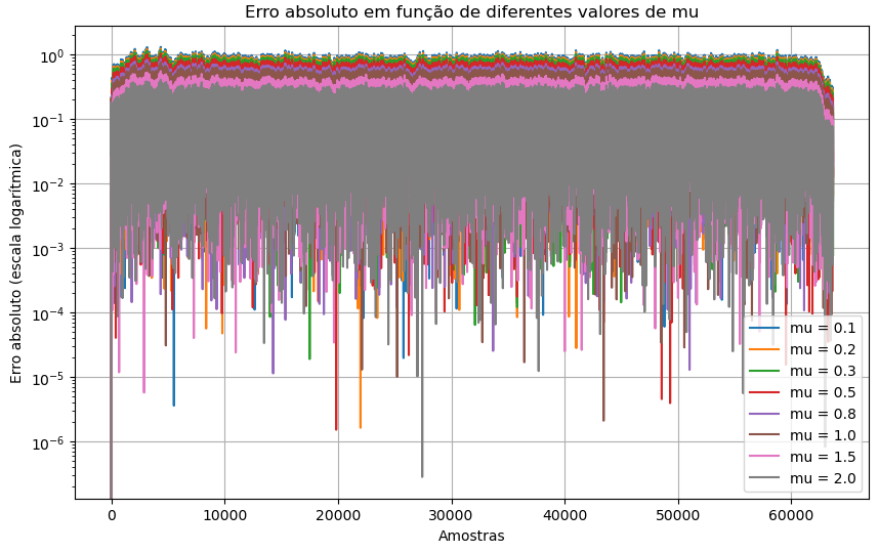

Figure 15: Gráfico do Erro Absoluto com Diferentes Valores de μ