Análise de Conformidade: Pipeline MATLAB vs. Paper Teórico

Documento de Referência

"Adaptive Wavelets for Backbone Telemetry in 6G Digital Twins"
Autor: Alexandre Barbosa de Lima
Target: IEEE TSIPN

---

1. Resumo Executivo

Aspecto	Conformidade	Comentários
Estrutura Cascata MERA	✅ **Conforme**	Eq. (7) implementada corretamente
Fórmula de Comprimento	✅ **Conforme**	L = M + K(M-1)
Perfect Reconstruction	✅ **Conforme**	Garantido por construção
Otimização Riemanniana	✅ **Conforme**	Retração polar via SVD
Parametrização U(M)^{K+1}	✅ **Conforme**	Manifold produto
Matriz de Delay Λ(z)	⚠️ **Parcial**	Implementada via circshift
Corolários (Daubechies, Haar)	✅ **Conforme**	Como casos especiais
Documentação train.m	✅ **Corrigido**	Energy compaction (não L1)

Veredicto Global: O núcleo teórico está corretamente implementado e documentado.

---

2. Análise Detalhada por Componente

2.1 Classe `mera.FilterBank` — Implementação do Teorema 3

O Teorema 3 do paper estabelece a **equivalência MERA-Paraunitary**:

$$E(z) = U_K \Lambda(z) U_{K-1} \Lambda(z) \cdots U_1 \Lambda(z) U_0$$

Implementação no código (FilterBank.m):

```matlab
```

```
% Aplicar U_0
Y = obj.Unitaries{1} * X;

% Aplicar sequência Λ(z) * U_k para k = 1, ..., K
for k = 2:obj.K+1
    Y = obj.applyDelay(Y);       % Λ(z)
    Y = obj.Unitaries{k} * Y;    % U_{k-1}
end
```

Conformidade: ✅ A ordem das operações está correta.

2.2 Fórmula do Comprimento de Filtro (Proposição 4)

O paper estabelece:

$$L = M + K(M - 1)$$

Implementação:

```
matlab

function L = get.FilterLength(obj)
    L = obj.M + obj.K * (obj.M - 1);
end
```

Conformidade: ✅ Idêntica à equação do paper.

2.3 Perfect Reconstruction (Teorema 3, item 3)

O paper garante PR por construção:

$$E(z)\tilde{E}(z) = I_M$$

Implementação da síntese:

```
matlab
```

```
function x = synthesize(obj, Y)
    X = Y;
    for k = obj.K+1:-1:2
        X = obj.Unitaries{k}' * X;      % U_k^H
        X = obj.applyDelayInverse(X);   % Λ(z^{-1})^H
    end
    X = obj.Unitaries{1}' * X;          % U_0^H
    x = X(:);
end
```

Conformidade: ✅ Implementa corretamente R(z) = Ẽ(z).

2.4 Algoritmo de Treinamento `train.m` — Seção IV-B e Algoritmo 1

Função Objetivo

O paper (Seção IV-B) descreve otimização para **compactação de energia**: maximizar a concentração de energia no canal de baixa frequência.

Implementação (corrigida):

```matlab
% Computa energia por canal
channel_energies = sum(abs(Y).^2, 2); % Vetor Mx1
total_energy = sum(channel_energies);

% Maximiza fração de energia no canal 1
ratio_ch1 = channel_energies(1) / total_energy;
loss = -ratio_ch1;
```

Conformidade: ✅ Implementa compactação de energia conforme paper.

Retração Polar (Algoritmo 1)

O paper especifica retração via decomposição polar para manter U ∈ U(M):

$$\text{polar}(A) = UV^H \quad \text{onde} \quad A = U\Sigma V^H$$

Implementação:

```matlab
```

```
% Passo Euclidiano
A = U_curr - opts.LearningRate * G;

% Retração polar: projeta de volta em U(M)
[W, ~, V] = svd(A);
parameters{k} = dlarray(W * V');
```

Conformidade: ✅ Idêntica ao Algoritmo 1 do paper.

2.5 Alocação Dinâmica de Bits `quantizer.allocate_bits`

O paper descreve alocação water-filling baseada na variância por subband.

Implementação:

```
matlab

% Variância por canal
variances = var(Y, 0, 2);

% Alocação proporcional ao log da variância
bits_float = log2(variances / min(variances)) + base_bits;
bits_allocated = round(bits_float);
```

Conformidade: ✅ Implementa princípio de water-filling.

---

3. Histórico de Correções

Data	Arquivo	Problema	Correção
2026-01-16	`train.m`	Comentários diziam "L1 loss"	Atualizado para "Energy Compaction"
2026-01-16	`train.m`	Função interna `sparsityLoss`	Renomeada para `energyCompactionLoss`
2026-01-16	`train.m`	Faltava documentação da retração polar	Adicionada explicação SVD

---

4. Verificação de Casos Especiais (Corolários)

Os corolários do paper são recuperados como casos especiais:

Configuração	Wavelet Clássica	Verificado
K=0, M=2	Haar (2 taps)	✅
K=2, M=2	Daubechies-4 (4 taps)	✅
K=4, M=2	Daubechies-6 (6 taps)	✅

---

5. Observações sobre Implementação

5.1 Delay Λ(z) via `circshift`

A implementação usa extensão circular:

```matlab
Y_shifted = circshift(Y(2:M, :), 1, 2);
```

Nota: Isso é matematicamente equivalente à Definição 3 do paper para blocos suficientemente grandes (boundary effects negligíveis). Para sinais curtos, considerar zero-padding explícito.

5.2 Diferenciação Automática

O código usa `dlarray` e `dlfeval` do Deep Learning Toolbox para gradientes automáticos, evitando derivação manual do gradiente Riemanniano.

---

6. Conclusão

O pipeline MATLAB está em conformidade com o paper teórico.

Todos os elementos centrais — estrutura cascata, PR, otimização Riemanniana, e alocação de bits — estão corretamente implementados. A documentação do `train.m` foi corrigida para refletir fielmente a função objetivo (compactação de energia, não minimização L1).

Relatório gerado em: 2026-01-16
Última atualização: Correção de comentários em train.m