

# **Formulação e Implementação de Adaptatividade Dinâmica em Malhas HigTree**

**Erick Rodrigues Canterle<sup>1</sup> Pedro Vinicius Souza Coimbra<sup>2</sup>  
Thiago Ferreira de França e Queiroz<sup>3</sup> Vinícius de Sá Ferreira<sup>4</sup>**

<sup>1234</sup>ICMC - Instituto De Ciências Matemáticas e de Computação  
Universidade de São Paulo

**12 de dezembro de 2025**



# Índices



1. Interpolação de malha
2. Refinamento Adaptativo da Interface
  - Funções
  - Fluxo de Execução
  - Exemplo
3. Fração de Volume
4. Referências



# Interpolação de malha

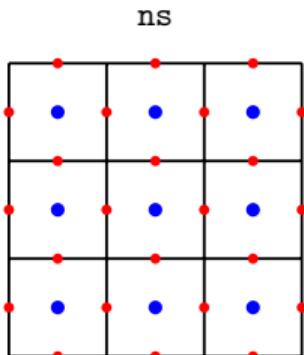


# Interpolação

1. A cada etapa de refinamento, uma nova malha é criada.
2. Os campos (velocidade, pressão, viscosidade, densidade, fração de volume) precisam ser transferidos da malha antiga para a nova.
3. Esse processo é feito por interpolação baseada em estêncil.

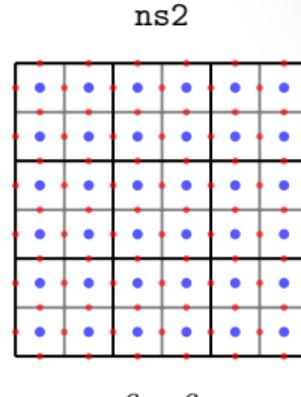
- Para cada célula ou face da nova malha:
  - pega-se o centro da célula ou face;
  - busca-se na malha antiga a vizinhança correspondente;
  - monta-se um estêncil local;
  - avalia-se o campo antigo naquele ponto.
- O valor interpolado é colocado na propriedade distribuída da nova malha.





$3 \times 3$

`higflow_interpolate_pressure()`  
`higflow_interpolate_velocity()`



$6 \times 6$



# Interpolação de Pressão

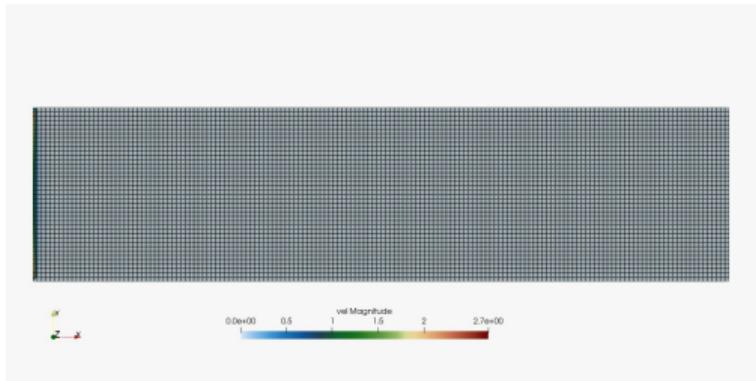
- Itera-se sobre todas as células da nova malha.
- Para cada centro:
  - constrói-se um estêncil no domínio antigo;
  - usa-se `compute_value_at_point` para obter a pressão interpolada;
  - escreve-se na propriedade `dpp` da nova malha.



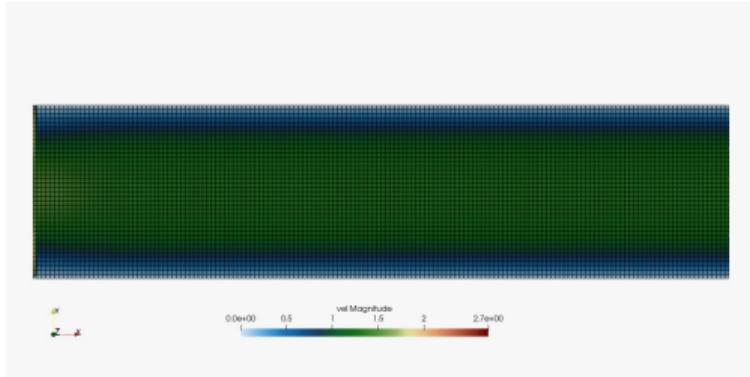
# Interpolação de Velocidade

- Velocidades são definidas nas faces.
- Para cada face da nova malha:
  - obtém-se o centro da face;
  - constrói-se o estêncil das faces antigas;
  - usa-se `dp_interpolate_from_stencil`;
  - escreve-se em `dpu[dim]` da nova malha.

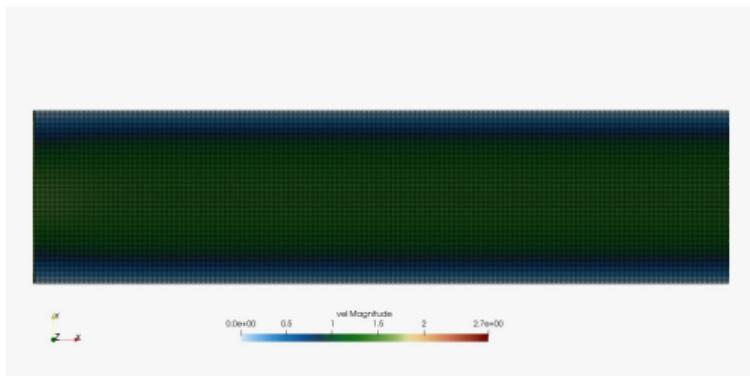
## Tempo inicial ( $t = 0$ )



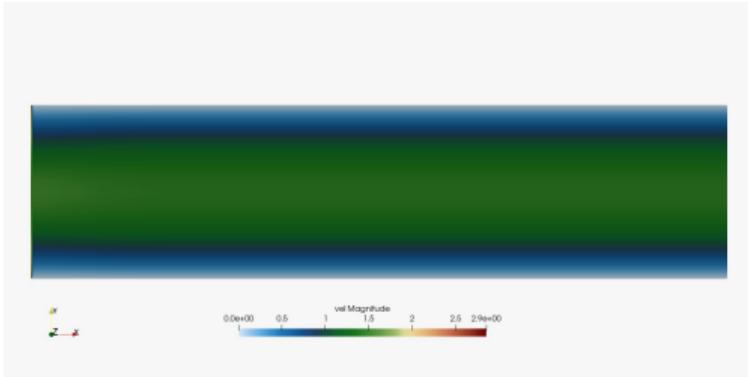
$t = 50$  (malha grossa)



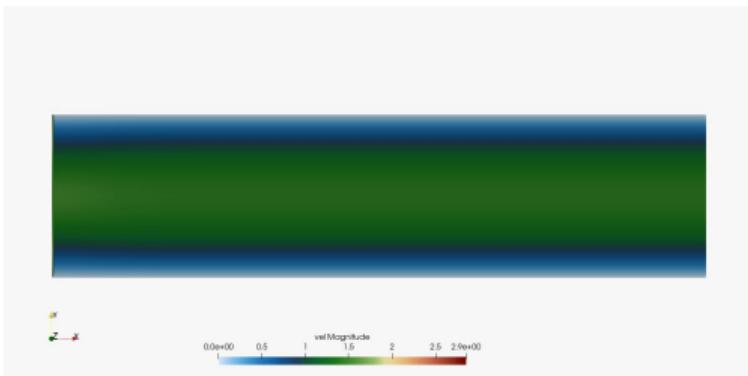
$t = 50$  (malha fina interpolada)



$t = 100$  (caso interpolado)



$t = 100$  (caso rodado na malha fina)





# Refinamento Adaptativo da Interface



# Função de Controle e Parâmetros

## Função principal: `higflow_make_adapted_tree_params`

```
hig_cell *higflow_make_adapted_tree_params(  
    higflow_solver *ns,  
    int num_levels,  
    real thresholds[]  
)
```

- **ns**: Ponteiro para o Solver Navier-Stokes.
- **num\_levels**: Número máximo de níveis de refinamento.
- **thresholds[]**: Array contendo os limiares de distância para cada nível de refinamento.
- **Retorno**: Raiz da nova árvore HiG com a malha adaptada.



# Identificação da Interface (Coleta de Sementes)

## Critério de Coleta de Células da Interface

```
// Coletar células que contêm a interface
real val = dp_get_value(ns->ed.mult.dpfracvol, clid);
if (val > 0.001 && val < 0.999) {
    hig_get_center(c, seeds[seed_count].center);
    seed_count++;
}
```

- Células com fração volumétrica (dpfracvol) entre 0.001 e 0.999 são identificadas como contendo a interface.
- O centro dessas células é armazenado como "semente", servindo como referência de distância para o refinamento.



# Critério de Refinamento Multi-nível

## Determinação do Nível Alvo

```
// Verificar distância da célula à semente mais próxima
real dist = sqrt(dist_sq_c(center_s, center_n));
int target_level = 0;

// Determinar nível alvo baseado nos limiares
for (int l = num_levels - 1; l >= 0; l--) {
    if (dist <= thresholds[l]) {
        target_level = l + 1;
        break;
    }
}
```

- O nível de refinamento (*target\_level*) é determinado pela distância (*dist*) da célula à semente da interface mais próxima.
- Quanto maior o nível de refinamento aplicado, menor a distância.
- Exemplo: *thresholds[0]* (0.05) para Nível 1; *thresholds[1]* (0.03) para Nível 2.



# Execução e Propagação do Refinamento

## Passos do Algoritmo de Refinamento Progressivo

```
// Executar múltiplas passadas para propagação completa
for (int pass = 0; pass < num_levels + 1; pass++) {
    // Encontrar células que precisam ser refinadas
    if (current_lvl < target_level) {
        to_refine[refine_count++] = neigh;
    }

    // Aplicar refinamento
    for(int i=0; i<refine_count; i++) {
        if(hig_get_number_of_children(to_refine[i]) == 0) {
            int nc[DIM];
            for(int d=0; d<DIM; d++) nc[d] = 2;
            hig_refine_uniform(to_refine[i], nc);
        }
    }
}
```

- Múltiplas passadas ( $\text{num\_levels} + 1$ ) são necessárias para garantir a propagação completa do refinamento pelos vizinhos.
- A função `hig_refine_uniform` divide uma célula folha em  $2^D$  células filhas, onde  $D$  é a dimensão.



# Engrossamento Controlado da Malha

## Critério de Engrossamento e Estabilidade

```
// Critério de distância com histerese
if (all_leaves) {
    real dist = sqrt(min_d2);
    real threshold = thresholds[parent_lvl] + coarsen_hys;

    if (dist > threshold) {
        safe_to_merge = 1;
    }
}

// Aplicação
if (safe_to_merge) {
    hig_merge_children(parents_to_merge[i]);
}
```

- O engrossamento (fusão de células filhas) ocorre apenas em regiões distantes da interface.
- É adicionada uma margem de segurança (`coarsen_hys = 0.005`) ao limiar de distância para evitar oscilações no refinamento/engrossamento (histerese).
- O processo é realizado em 4 passadas para garantir a estabilidade numérica da malha.



# Fluxo de Execução

1. **Clonagem da Malha Original:** Cria-se uma cópia da HiGTree para aplicar as modificações.
2. **Identificação da Interface:** Coleta de células "sementes" para guiar o refinamento.
3. **Refinamento Progressivo:** Múltiplas passadas de refinamento baseadas na distância aos limiares.
4. **Engrossamento Controlado:** Remoção de refinamento em regiões distantes da interface (usando histerese).
5. **Retorno da Malha Adaptada:** Substituição da malha original pela nova malha adaptada, preservando a topologia da interface.



# Exemplo com `threshold[] = {0.05, 0.03}`

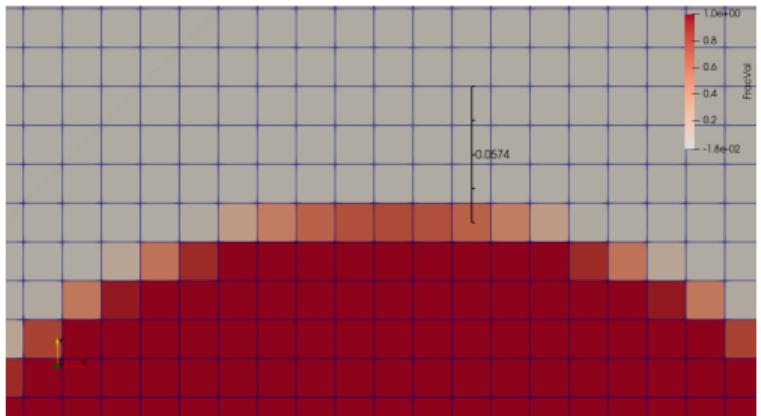


Figura 1: Transição: Nível 0

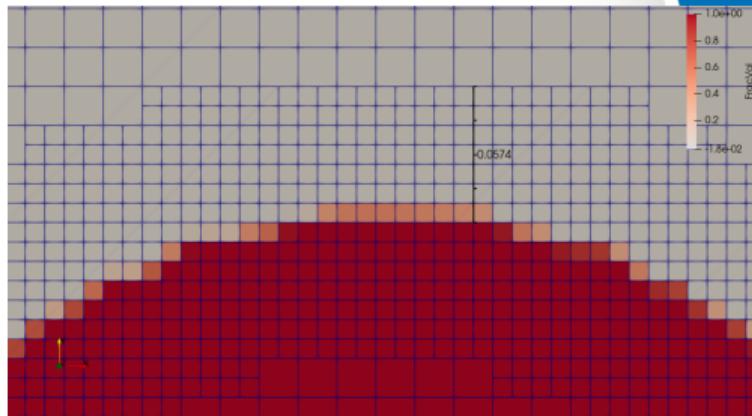


Figura 2: Resultado: Nível 1



# Refinamento Adicional para Nível 2

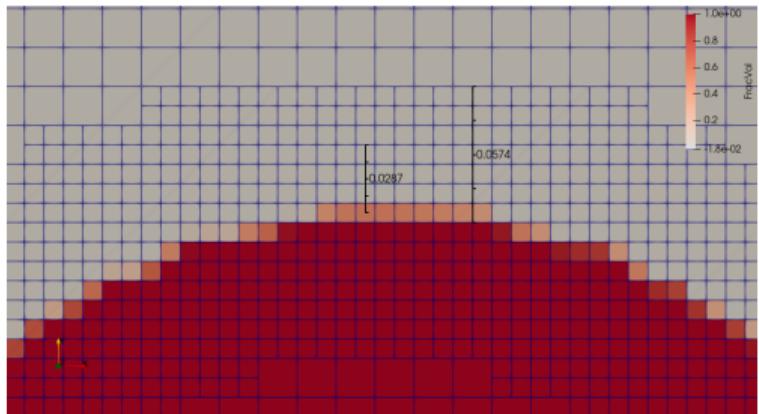


Figura 3: Transição: Nível 1

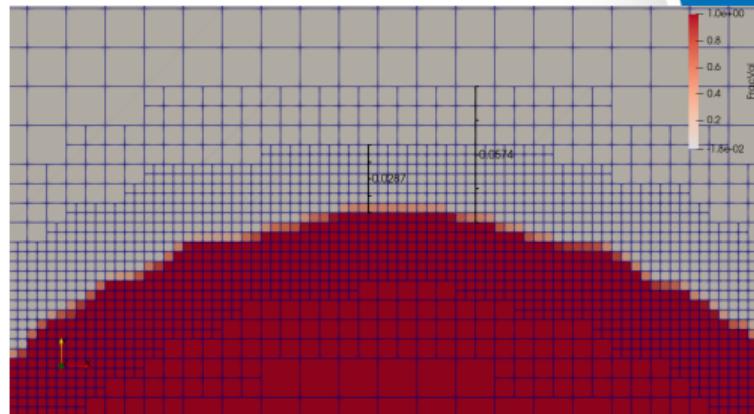


Figura 4: Resultado: Níveis 1 e 2



# Evolução Temporal do Refinamento da Bolha

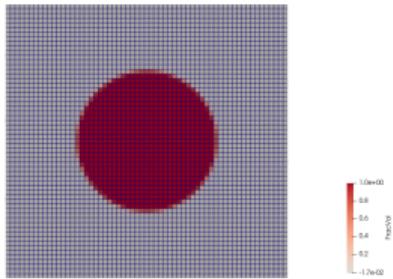


Figura 5: Tempo Inicial  
( $t = 0$ )

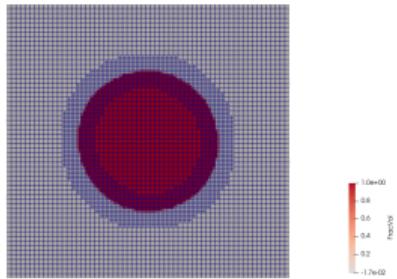


Figura 6: Tempo Intermediário  
( $t = 5$ )

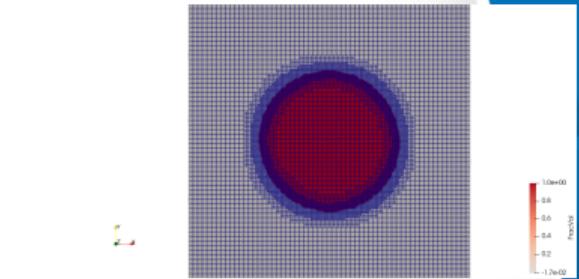


Figura 7: Tempo Final  
( $t = 10$ )



# Fração de Volume



# Difusão da Fração de Volume

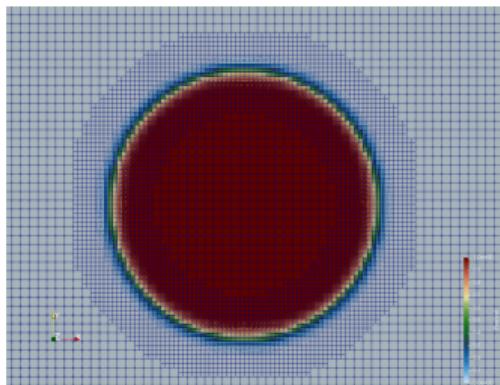


Figura 8: step = 5

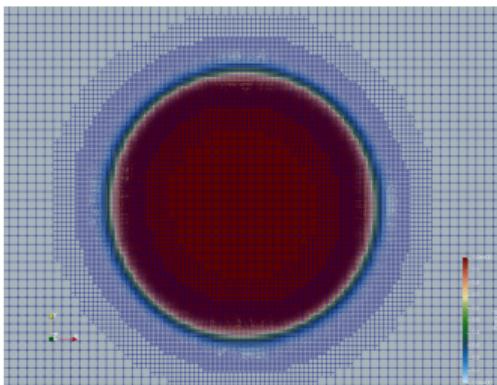


Figura 9: step = 15

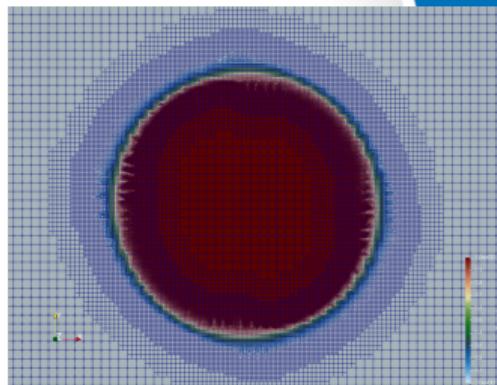


Figura 10: step = 25

# Filtro



```
// Valor definido empiricamente (altamente testado)
if (ns->par.step == 5 || ns->par.step == 10){
    real val = 0.4;
    // Operador ternário para calcular a fração de volume sharp
    fracvol = (fracvol > 1 - val) ? 1 : (fracvol < val ? 0 : fracvol);
}
```



# Aplicação do filtro na Fração de Volume

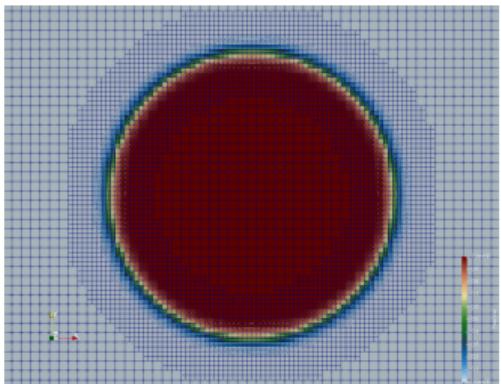


Figura 11: step = 5

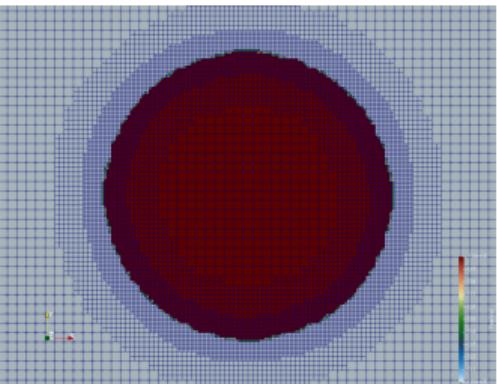


Figura 12: step = 15

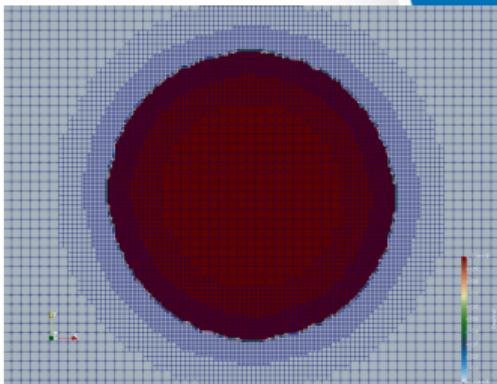


Figura 13: step = 25



# Muito Obrigado Pela Atenção

**Dúvidas e Sugestões**

[ecanterle@usp.br](mailto:ecanterle@usp.br),

[pedro.coimbra@usp.br](mailto:pedro.coimbra@usp.br),

[queiroztff@usp.br](mailto:queiroztff@usp.br),

[desaferreira@usp.br](mailto:desaferreira@usp.br)

# Referências I

