Tópicos para Apresentação
Equação de Helmholtz - Métodos Numéricos

INTRODUÇÃO

- **Equação de Helmholtz**: $\nabla^2 u + k^2 u = 0$ em $\Omega = [0,1]^2$
  - Equação diferencial parcial elíptica
  - Surge da equação da onda com solução harmônica no tempo
  - Modela fenômenos ondulatórios em regime harmônico
- **Interpretação dos termos**:
  - $\nabla^2 u$: Operador Laplaciano — mede a curvatura espacial da solução
  - $k^2 u$: Termo reativo — impõe caráter oscilatório
  - $u(x, y)$: Campo escalar desconhecido que queremos determinar
- **Número de onda** k:
  - Relacionado à frequência espacial: $k = \omega/c$
  - Quanto maior k, maior a frequência espacial
  - Valores testados: $k = 1, 20, 40, 100$
  - Para k grande, requer malhas mais refinadas
- **Aplicações práticas**:
  - Acústica arquitetônica
  - Propagação eletromagnética
  - Vibrações mecânicas
  - Difração e espalhamento de ondas
- **Objetivo do trabalho**:
  - Aplicar o Método das Diferenças Finitas Centradas de Segunda Ordem (MDFC2)
  - Analisar efeitos numéricos, especialmente poluição numérica
  - Investigar influência do número de onda k
  - Avaliar uso de IA como ferramenta auxiliar

METODOLOGIA

- **Parâmetro N — Refinamento da malha**:
  - N: número de subintervalos em cada direção do domínio $[0, 1]^2$
  - Malha uniforme: dividimos $[0, 1]$ em N partes iguais
  - Tamanho do passo: $h = 1/N$
  - Total de pontos da malha: $(N + 1) \times (N + 1)$ pontos
  - Pontos internos (incógnitas): $(N - 1) \times (N - 1) = (N - 1)^2$ incógnitas
  - Valores testados: $N = 64, 128, 192, 256$
  - Exemplo: $N = 64 \Rightarrow h = 1/64 = 0.015625$, $63 \times 63 = 3.969$ incógnitas
  - Exemplo: $N = 256 \Rightarrow h = 1/256 = 0.00390625$, $255 \times 255 = 65.025$ incógnitas
- **Discretização MDFC2 (Método das Diferenças Finitas Centradas de 2ª Ordem)**:
  - Stencil de 5 pontos: centro + 4 vizinhos (norte, sul, leste, oeste)
  - Equação discretizada:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + k^2 u_{i,j} = 0$$

  - Primeiro termo: discretização de $\frac{\partial^2 u}{\partial x^2}$
  - Segundo termo: discretização de $\frac{\partial^2 u}{\partial y^2}$
  - Terceiro termo: termo reativo $k^2 u$ avaliado no ponto (i, j)
- **Sistema linear resultante**:
  - Forma: $A\mathbf{u} = \mathbf{b}$
  - Matriz $A = L + k^2 I$, onde L é o Laplaciano discreto
  - Dimensão: $(N - 1)^2 \times (N - 1)^2$
  - Matriz esparsa: apenas 5 diagonais não-nulas
  - Vetor $\mathbf{b}$: construído a partir das condições de contorno Dirichlet
- **Matrizes esparsas — Formatos**:
  - **LIL (List of Lists)**: formato usado para montagem eficiente da matriz
    - Permite inserção rápida de elementos
    - Ideal para construção da matriz
  - **CSC (Compressed Sparse Column)**: formato usado para solução
    - Otimizado para operações de álgebra linear
    - Usado pelo `scipy.sparse` para solvers
    - Conversão LIL → CSC após montagem
  - Vantagem: memória irrisória comparada à matriz densa
  - Exemplo: $N = 200 \Rightarrow$ matriz densa $\approx$ 12 GB RAM (inviável)
  - Com esparsidade: apenas algumas dezenas de MB
- **Solvers implementados — Métodos de solução do sistema linear**:
  - **SPLU — Sparse LU (Fatoração LU Esparsa)**:

    - Nome completo: `scipy.sparse.linalg.splu`
    - Tipo: Método direto (resolve exatamente, sem iterações)
    - Processo:
      1. Fatora a matriz A em $A = LU$ (Lower-Upper)
      2. Resolve $Ly = b$ (substituição progressiva)
      3. Resolve $Ux = y$ (substituição regressiva)
    - Eficiente para: $N \leq 192$ (até ~ 40.000 incógnitas)
    - Complexidade: $O(N^3)$ operações, $O(N^2)$ memória
    - Vantagem: solução exata, sem erro de iteração
    - Desvantagem: custo cresce rapidamente com N
  - **GMRES+ILU — Generalized Minimal Residual com pré-condicionador ILU**:
    - **GMRES**: `scipy.sparse.linalg.gmres`
      - Nome completo: Generalized Minimal Residual Method
      - Tipo: Método iterativo (aproximações sucessivas)
      - Ideia: encontra solução no espaço de Krylov
      - Minimiza resíduo $\|b - Ax\|_2$ em cada iteração
      - Parâmetros: restart=100, maxiter=1000
      - Tolerâncias: rtol=10^{-8}, atol=10^{-12}
    - **ILU**: `scipy.sparse.linalg.spilu`
      - Nome completo: Incomplete LU Factorization
      - Tipo: Pré-condicionador (melhora convergência do GMRES)
      - Processo: fatora $A \approx LU$ de forma incompleta (mais rápida)
      - Parâmetros: drop_tol=10^{-3}, fill_factor=20
      - Função: acelera convergência do GMRES
    - Eficiente para: $N \geq 256$ (sistemas grandes)
    - Complexidade: $O(N^2)$ por iteração
    - Vantagem: escalável para sistemas muito grandes
    - Desvantagem: pode não convergir se mal condicionado
  - **Seleção automática (modo "auto")**:
    - Heurística: escolhe solver baseado no tamanho do sistema
    - Se $n \leq 40.000$ incógnitas: usa SPLU
    - Se $n > 40.000$ incógnitas: usa GMRES+ILU
    - Estratégia de fallback robusta:
      - Se ILU falha e $n \leq 100.000$: tenta SPLU
      - Se ILU falha e $n > 100.000$: usa GMRES sem pré-condicionador
    - Garante solução para todos os casos

- **Solução exata para validação**:
  - Forma: $u_{\text{exata}} = \sum_{i=1}^{3} \cos(k(x \cos\theta_i + y \sin\theta_i))$
  - Superposição de 3 ondas planas
  - Cada onda tem direção definida pelo ângulo θ_i
  - Imposta como condição de Dirichlet na fronteira
  - Permite calcular erro exato: $u_{\text{num}} - u_{\text{exata}}$

- **6 grupos de testes**:

  - Cada grupo tem diferentes combinações de ângulos $\Theta = \{\theta_1, \theta_2, \theta_3\}$
  - Grupo 1: $\Theta = \{0, \pi/8, \pi/4\}$
  - Grupo 2: $\Theta = \{\pi/16, \pi/8, 3\pi/16\}$
  - Grupo 3: $\Theta = \{\pi/8, 3\pi/16, \pi/4\}$
  - Grupos 4, 5, 6: outras combinações
  - Permite análise abrangente do comportamento do método

- **Otimizações implementadas**:
  - **Cache de matrizes**: reutiliza matrizes Laplacianas para mesmo N
  - Ganho: $\sim 90\%$ mais rápido na construção
  - **Paralelização**: processamento paralelo de múltiplos casos
  - Ganho: 3-4 vezes mais rápido
  - **Pré-aquecimento**: construção prévia de estruturas

RESULTADOS PRINCIPAIS

Quantificação do Erro

- **Norma $L^2(\Omega)$ integral**:
  - Definição: $\|u\|^2_{L^2(\Omega)} = \int_\Omega |u(x,y)|^2 \, dx \, dy$
  - Erro relativo: $E_{L^2} = \frac{\|u_{\text{aprox}} - u_{\text{exata}}\|_{L^2(\Omega)}}{\|u_{\text{exata}}\|_{L^2(\Omega)}}$

- **Aproximação discreta pela regra do retângulo**:
  - $\|u\|^2_{L^2(\Omega)} \approx h^2 \sum_{i,j} |u_{i,j}|^2 = h^2 \|\mathbf{u}\|_2^2$
  - Portanto: $\|u\|_{L^2(\Omega)} \approx h\|\mathbf{u}\|_2$
  - Erro calculado: $E_{L^2} \approx \frac{h\|\mathbf{U}_{\text{aprox}} - \mathbf{U}_{\text{exata}}\|_2}{h\|\mathbf{U}_{\text{exata}}\|_2}$
  - Justificativa: aproximação de ordem $O(h^2)$, consistente com ordem do método

Resultados por número de onda k

- $k = 1$ **(baixa frequência)**:
  - Erro relativo: 10^{-7} a 10^{-8} (excelente precisão)
  - Taxa de convergência: $p \approx 2.0$ (ordem 2 confirmada)
  - Comportamento: método funciona perfeitamente
  - Exemplos:
    - $N = 64$: erro 6.4×10^{-7}
    - $N = 128$: erro 1.6×10^{-7} (4x menor)
    - $N = 256$: erro 4.1×10^{-8} (15x menor)

- $k = 20$ **(frequência média)**:
  - Erro relativo: 4.2×10^{-2} para $N = 128$ (4.2%)
  - Taxa de convergência: $p \approx 1.5 - 1.8$ (degradada)

  - Comportamento: início do efeito de poluição numérica
  - Exemplos:
    * $N = 64$: erro 2.1×10^{-1} (21%)
    * $N = 128$: erro 4.2×10^{-2} (4.2%, 5x menor)
    * $N = 256$: erro 1.0×10^{-2} (1.0%, 21x menor)

- $k = 40$ **(alta frequência)**:
  - Erro relativo: 2.8×10^{-1} para $N = 128$ (28%)
  - Taxa de convergência: $p \approx 1.2 - 1.5$ (degradada)
  - Comportamento: degradação significativa
  - Requer $N \geq 256$ para precisão aceitável

- $k = 100$ **(muito alta frequência)**:
  - Erro relativo: > 1.0 (poluição numérica severa)
  - Taxa de convergência: $p < 1$ (muito degradada)
  - Comportamento: erro pode ultrapassar 300%
  - Exemplos:
    * $N = 64$: erro 2.5 (250%)
    * $N = 128$: erro 1.5 (150%)
    * $N = 256$: erro 3.6 (360%, piora!)
  - Mesmo com refinamento, erro permanece alto

Regra de Ouro — Pontos por comprimento de onda

- **Definição**: $N_\lambda = \frac{2\pi N}{k}$ pontos por comprimento de onda
  - N: número de subintervalos
  - k: número de onda
  - Comprimento de onda: $\lambda = 2\pi/k$
- **Recomendação**: $N_\lambda 10 - 20$ pontos por comprimento de onda
- **Exemplos**:
  - $k = 1$, $N = 64$: $N_\lambda = \frac{2\pi \times 64}{1} \approx 402$ (excelente)
  - $k = 20$, $N = 128$: $N_\lambda = \frac{2\pi \times 128}{20} \approx 40$ (bom)
  - $k = 100$, $N = 256$: $N_\lambda = \frac{2\pi \times 256}{100} \approx 16$ (marginal)
- **Requisito para alta frequência**: $N_\lambda \geq 20 - 30$ para precisão aceitável

Desempenho Computacional

- $N = 64$:
  - Incógnitas: $63 \times 63 = 3.969$
  - Tempo de construção: 0.001 s
  - Tempo de solução: 0.008 s

  - Solver: SPLU
- $N = 128$:
  - Incógnitas: $127 \times 127 = 16.129$
  - Tempo de construção: 0.003 s
  - Tempo de solução: 0.039 s
  - Solver: SPLU
- $N = 192$:
  - Incógnitas: $191 \times 191 = 36.481$
  - Tempo de construção: 0.006 s
  - Tempo de solução: 0.117 s
  - Solver: SPLU
- $N = 256$:
  - Incógnitas: $255 \times 255 = 65.025$
  - Tempo de construção: 0.012 s
  - Tempo de solução: 0.423 s
  - Solver: GMRES+ILU (necessário para sistemas grandes)
- **Otimizações**:
  - Cache: $\sim 90\%$ mais rápido na construção
  - Paralelização: 3-4 vezes mais rápido no processamento

ANÁLISE E DISCUSSÃO

- **Poluição numérica**:
  - Definição: mesmo com refinamento, erro cresce com aumento de k
  - Causa: poucos pontos por comprimento de onda ($N_\lambda < 20$)
  - Mecanismo:
    1. Poucos pontos por comprimento de onda
    2. Erro de fase: velocidade de fase numérica $\neq$ exata
    3. Acúmulo: erro se propaga e amplifica
    4. Resultado: solução numérica "atrasa"espacialmente
  - Consequência: taxa de convergência degrada ($p < 2$)
- **Comparação entre grupos**:
  - Diferenças pequenas mas significativas
  - Devido à direção das ondas planas na solução exata
  - Efeito secundário comparado à influência de k e N
- **Validação**:
  - Resultados consistentes com literatura

  - Taxa de convergência $p \approx 2.0$ para k pequeno (conforme teoria)
  - Requisito de malha $N_\lambda \geq 10 - 20$ (alinhado com literatura)
  - Poluição numérica observada conforme esperado
- **Limitações do método**:
  - Poluição numérica inerente a métodos de baixa ordem
  - Custo computacional cresce com N^2
  - Limitado a domínios retangulares simples
  - Para k grande, requer malhas muito refinadas

CONCLUSÕES

- **MDFC2 adequado** para $k \leq 20$
  - Boa precisão com malhas razoáveis
  - Erros aceitáveis ($< 5\%$ com $N \geq 128$)
- **Taxa de convergência** confirma ordem 2 para k pequeno
  - $p \approx 2.0$ para $k = 1$
  - Valida implementação e análise teórica
- **Poluição numérica** limita eficiência para k grande
  - Fenômeno inerente a métodos de baixa ordem
  - Requer técnicas mais sofisticadas para mitigar
- **IA útil** como ferramenta de suporte acadêmico
  - Auxiliou na implementação, análise e documentação
- **Perspectivas futuras**:
  - Métodos de ordem superior (4^{a}, 6^{a} ordem)
  - Malhas adaptativas
  - Pré-condicionadores especializados
  - Extensão para 3D

PONTOS-CHAVE PARA DESTACAR

- **Excelente precisão** para $k = 1$: erro $< 10^{-6}$
- **Boa precisão** para $k = 20$ com $N \geq 128$: erro $< 5\%$
- **Poluição numérica** severa para $k = 100$: erro $> 300\%$
- **Importância do refinamento**: $N = 64 \rightarrow 256$ reduz erro em 21x para $k = 20$
- **Implementação eficiente**: matrizes esparsas, otimizações, seleção automática de solver

DICAS DE APRESENTAÇÃO

- **Início**: Contextualizar equação de Helmholtz e aplicações
- **Metodologia**: Enfatizar simplicidade do MDFC2 e eficiência das matrizes esparsas
  - Explicar claramente o que é N e sua relação com o refinamento
  - Detalhar os solvers: SPLU e GMRES+ILU, com seus nomes completos
- **Resultados**: Mostrar gráficos de superfície 3D e cortes 2D
- **Análise**: Explicar poluição numérica de forma clara
- **Conclusões**: Destacar limitações e perspectivas futuras
- **Tempo**: Distribuir bem o tempo entre seções