XXX Congresso de Iniciação Científica UNICAMP

Paralelização de código CFD

Palavras-Chave: Paralelização, Camada-Limite, CFD.

Autores:
Gabriel Moreira Beltrami [Unicamp]
Prof. Dr. Rogério Gonçalves dos Santos (orientador) [Unicamp]

INTRODUÇÃO:

A presente iniciação científica tem por objetivo a paralelização de um código CFD de linguagem Fortran90 chamado ConDiRa[1][2] criado pelo doutor Jan Armengol, do qual o modelo RANS K-ε-LS de turbulência foi utilizado. Com isso, possibilita-se o estudo de casos complexos, como análise de espessura de camadas limites com variações de viscosidade e velocidade de um fluido sobre uma placa plana (problema tratado nesta iniciação científica) ou o aprimoramento das aproximações feitas pelos modelos de simulação, como a calibração dos fatores de modelos de turbulência.

METODOLOGIA:

Durante a elaboração do código final, duas opções de paralelização eram possíveis, seguindo o diagrama ao lado.

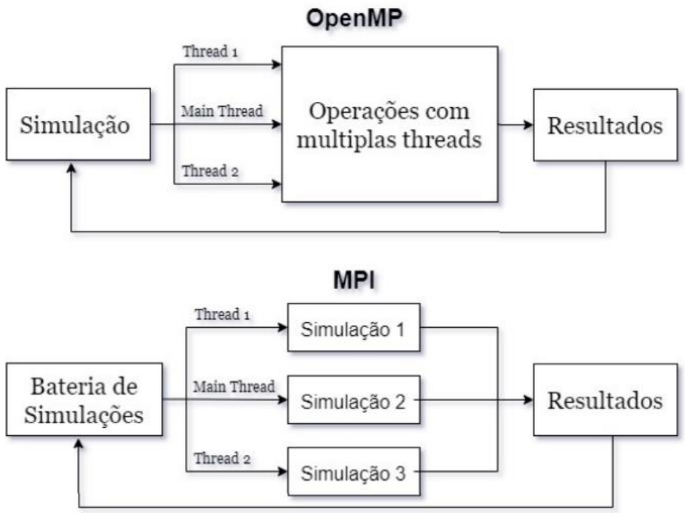

Figura 1 – Diagrama de possíveis paralelizações. – Fonte: Autoria própria.

A primeira tentativa, feita utilizando a biblioteca OpenMP[3], permitiria executar uma única simulação de forma mais rápida, sequencialmente com as restantes da bateria de simulações. Essa metodologia foi abandonada devido à alta complexidade do código, resultando em uma paralelização muito complexa e pouco eficiente, que não atenderia ao objetivo proposto, embora possibilitasse a execução de baterias pequenas de simulações mais complexas.

A segunda metodologia, efetivamente adotada, foi o uso da biblioteca MPI[4], que permitiu uma paralelização fácil e eficiente, na qual paralelizava-se o código como um todo, executando um código em cada núcleo (ou *thread*), com simulações independentes e simultâneas. Assim, permite-se variar um número finito de variáveis (no presente estudo, viscosidade do fluído e velocidade do escoamento na

entrada da cavidade). Dessa forma, é possível estudar diversos tipos de baterias de simulações e os efeitos da mudança das variáveis de estudo no resultado, não importando a complexidade do problema. A desvantagem deste segundo método em relação ao primeiro proposto é que a quantidade de memória RAM utilizada pela máquina é proporcional ao número de simulações simultâneas, enquanto para o primeiro método, consumia-se apenas a memória de uma simulação com pequenas variações momentâneas para a ulitização de diferentes núcleos.

Após essa implementação, utilizou-se da biblioteca *scipy* de Python para a geração de amostras de baixa discrepância, *quasi-alatórias* para a velocidade e viscosidade a serem estudadas pelas simulações. Visando um Reynolds acima de 5e5 ao final da placa, para que haja um regime turbulento, utilizou-se amostras de velocidade de 5 a 10m/s e amostras de viscosidade que variavem entre a viscosidade de hidrogênio e xenônio, com o valor referente ao ar atmosférico contido neste intervalo. Ressalta-se, neste ponto, que as viscosidades resultantes podem resultar em um fluido inexistente, mas que a título de estudos carregam seu valor.

Posteriormente, para o pós-processamento dos dados, elaborou-se mapas do tipo junção de funções de densidade de probabilidades, permitindo uma clara visualização das entradas e saídas das simulações.

Resultados:

Uma vez implementada a biblioteca MPI, obteve-se os seguintes resultados quanto a eficiência da paralelização:

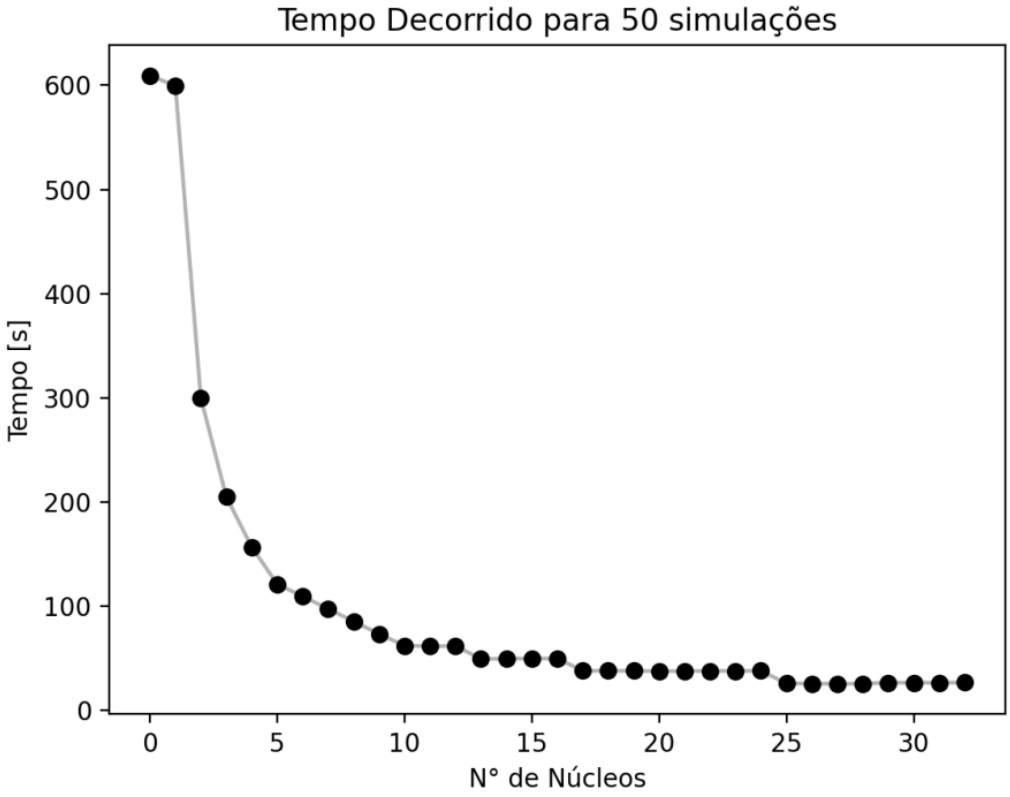

Figura 2 – Tempo decorrido para um total de 50 simulações em função do número de threads/núcleos utilizados. – Fonte: Autoria própria.

Nota-se que o ponto com “zero” núcleos representa o programa original, não modificado. Além disso, ressalta-se que devido ao fato de que o funcionamento desta paralelização é baseado na simultaneidade das simulações, o tempo necessário para que a bateria seja concluida é diretamente relacionado a dois fatores específicos: o tempo médio de cada simulação (no caso acima, de aproximadamente 12 segundos) e a divisão natural do número de total simulações pelo número de núcleos disponíveis, uma vez que o tempo total pode ser representado pelo tempo em que o núcleo responsável pelo maior número de operações as realize completamente.

Em seguida, elabora-se o espaço amostral para um total de 256 simulações, realizadas em 64 núcleos, variando a velocidade do escoamento e a viscosidade do fluido como mostrado no mapa a seguir:

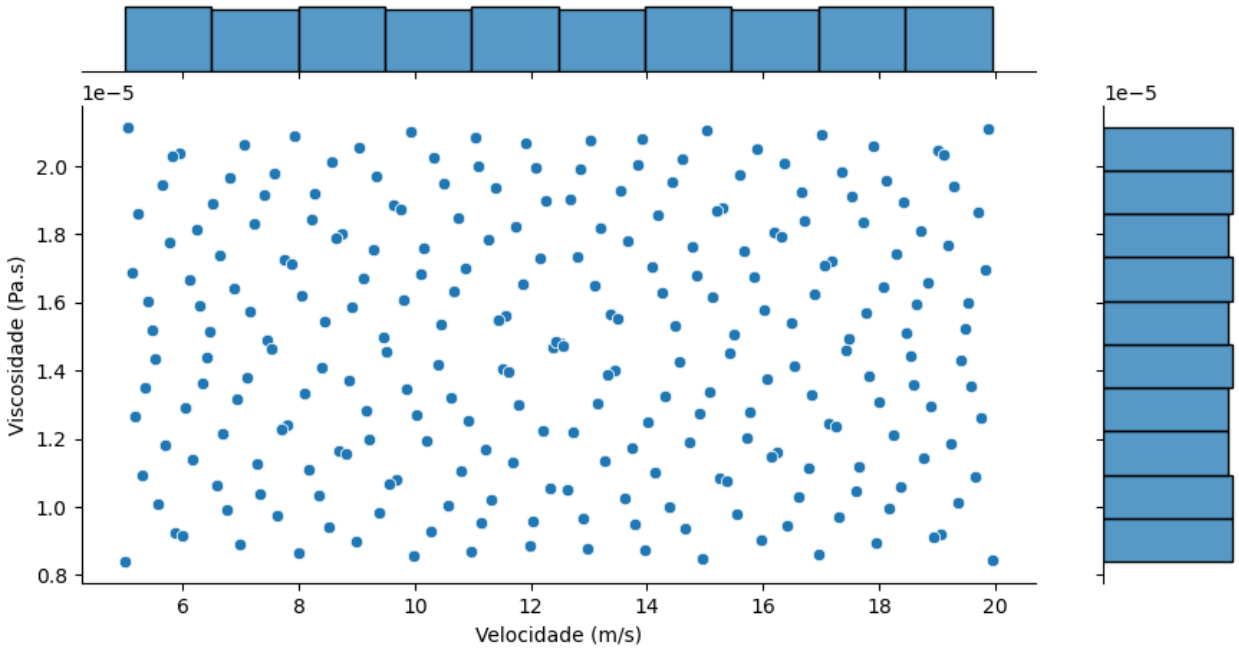

Figura 3 – Diagrama de amostras simuladas. – Fonte: Autoria própria.

Extraindo o contorno de velocidades de uma amostra simulada, tem-se:

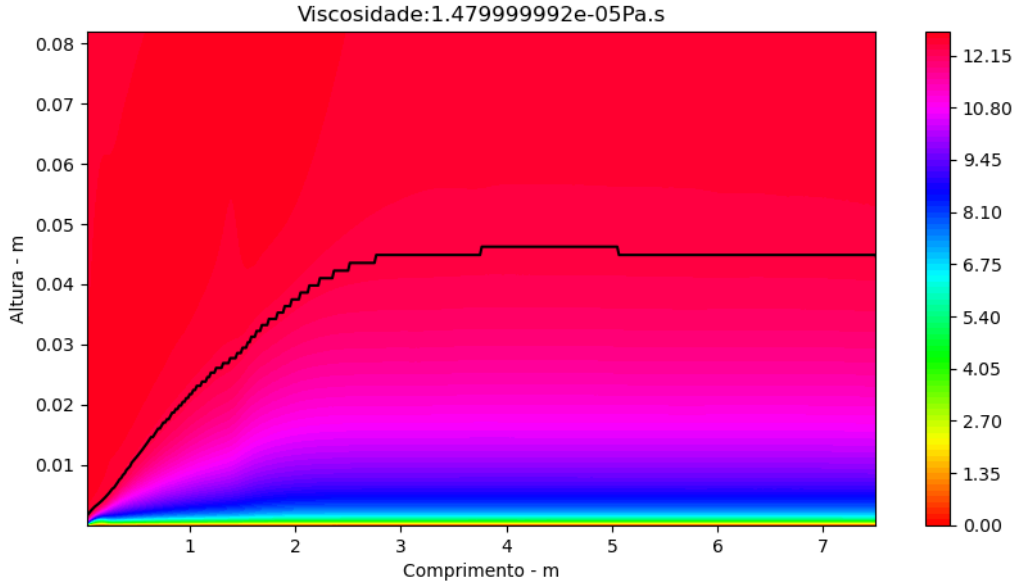

Figura 4 – Mapa de velocidades obtido. – Fonte: Autoria própria.

Ressalta-se neste gráfico a presença de uma linha preta, indicando a espessura da camada limte do fluxo proposto.

Finalmente, organizando os dados de todas as espessuras de camada limite no final da placa, obtém-se o seguinte diagrama, relacionando as com o número de Reynolds.

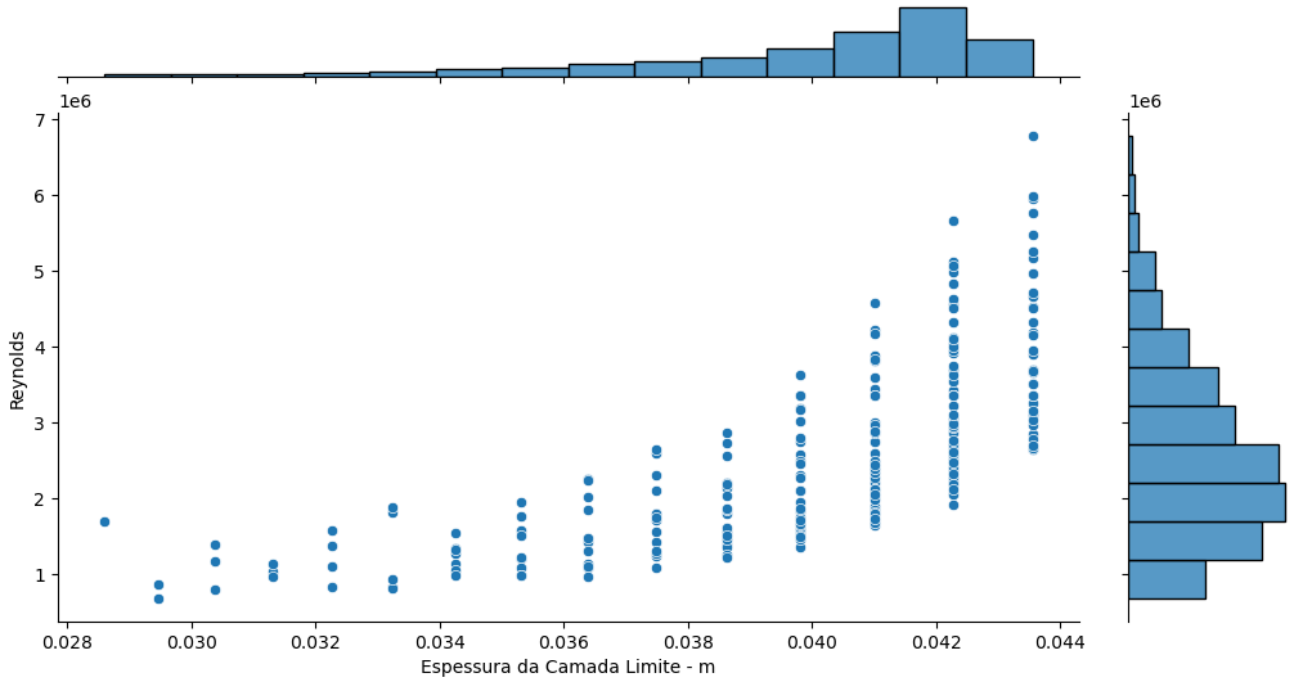

Figura 5 – Distribuição das espessuras de camada limite para determinado Reynolds. – Fonte: Autoria própria.

É possível ressaltar nesse gráfico a concentração de pontos de espessura de camada limite em linhas verticais, devido a discretização da malha utilizada na simulação, que poderia ser mitigado através do refino da malha, requerindo um poder computacional maior ou maior tempo de simulação. Nota-se que a análise feita das amostras dispostas na malha apresentada só foi possível devido à paralelização, uma vez que demorariam dias para simularem individualmente.

Conclusão:

Em suma, a paralelização do código permite explorar a influência de diferentes variáveis simultaneamente e/ou problemas com geometrias ou condições de contornos mais complexas, executando tais tarefas em um tempo significativamente menor dado um poder computacional suficiente. Além disso, a distribuição quasi-aleatória das amostras permite o estudo de eventos probabilísticos/aleatórios que possam aparecer para determinados casos.

Com isso, a evolução de modelos de simulação mais rápidos e acurados se torna possível, refinando os existentes para casos mais específicos ou mais gerais, à escolha do usuário do código.

Referências Bibliográficas:

[1] ARMENGOL, J. BANNWART, F. SANTOS, R. Effects of variable air properties on transient natural convection for large temperature differences. International Journal of Thermal Sciences, Amsterdam: Volume 120, p.67-79, 2017.

[2] Mendoza, C.U., Salinas, C.T., Armengol, J.M., Beicker, R., Santos, R.G., BRAZILIAN CONGRESS OF THERMAL SCIENCES AND ENGINEERING, 16, 2016, Vitória. Numerical investigation of heat transfer by natural convection and radiation in a cavity with participating media. . . Rio de Janeiro: Associação Brasileira de Engenharia e Ciências Mecânicas, 2018. 9p.

[3] Centro Nacional de Pesquisa de Alto Desempenho de São Paulo, Introdução ao OPENMP, Revisão 2014 (apostila).

[4] GROPP, William; LUSK, Ewing; Skjellum, Anthony. Using MPI: Portable Parallel Programming with the Message-Passing Interface. Terceira edição. Massasshusetts: MIT Press, novembro de 2014.