Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
PAD Dinâmica dos Fluidos Computacional
Grupo de alunos: ALG

Resultados da 1ª implementação do escoamento supersônico na placa plana do ALG

Belo Horizonte
2 de janeiro de 2007

1. Introdução

Em C/C++ com a biblioteca gráfica OpenGL e a estrutura de dados ALG, foi implementado a resolução das equações completas de Navier-Stokes a partir do método baseado na iteração de Picard, o qual é montado usando a forma genérica:

$$\boldsymbol{U}^{n+1,\,k+1} = \frac{\Delta t}{\Delta L}\left(\boldsymbol{E}_w^{\,n+1,\,k} - \boldsymbol{E}_e^{\,n+1,\,k} + \boldsymbol{F}_s^{\,n+1,\,k} - \boldsymbol{F}_n^{\,n+1,\,k} \right) + \boldsymbol{U}^n$$

Onde ***U***, ***E*** e ***F*** são vetores de 4 coordenadas que representam as equações bidimensionais de Navier-Stokes.

O programa inicia solicitando valores para o nível de refinamento inicial da malha, o nível máximo ao refinar, o nível mínimo ao desrefinar, o incremento de tempo e o erro máximo permitido na iteração de Picard. Para obter uma malha uniforme durante toda a simulação, é necessário que o refinamento inicial, o máximo e o mínimo sejam iguais.

Na interface gráfica do OpenGL, onde podem ser vistos os gráficos de cor e a malha, a legenda de cores é um novo recurso, inicialmente feito pelo aluno Henrique Favarini e depois modificado pelo aluno André Medeiros. O campo vetorial para a velocidade foi feito pelo aluno Maycon Costa. As posições dos vértices dos triângulos representando as setas dos vetores foram computados por intersecções entre retas e círculos, mas depois isso foi discutido e refeito usando matrizes de rotação, o que reduziu ligeiramente o gasto computacional com simplicidade.

A respeito do código fonte do programa, vários nomes de variáveis e funções foram revistos e melhorados, a fim de aumentar a legibilidade do código.

2. Discussão dos resultados obtidos

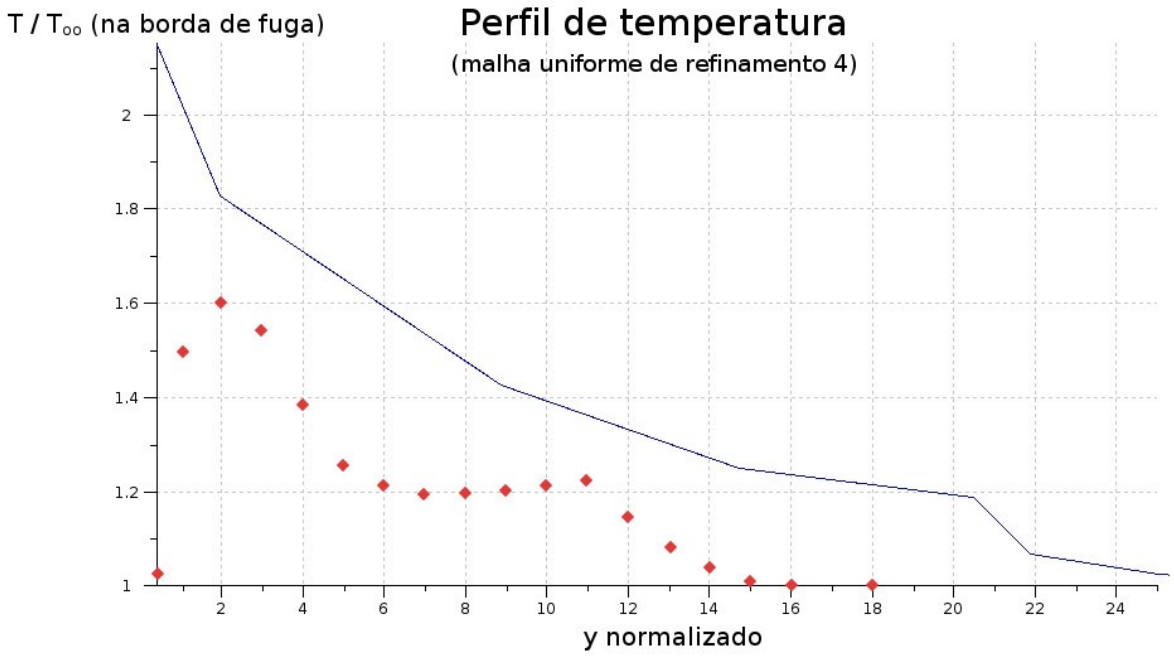
Heuristicamente, descobrimos alguns valores para a configuração inicial que geralmente permitem estabilidade na solução numérica. Para malhas uniformes de refinamento de nível 6 ou menor, o incremento de tempo utilizado é 10^{-10}, ou 10^{-11}, ou 10^{-12}, ou menores. Incrementos maiores resultam na desestabilização da solução. O erro máximo permitido na iteração de Picard geralmente utilizado é 10^{-1} ou menores, como 10^{-5}. Ainda desconsiderando o refinamento adaptativo ao longo do tempo, o nível de refinamento das malhas uniformes geralmente são de 1 até 6. Valores acima dessa faixa costumam deixar a aplicação muito pesada em computadores comuns.

Para comparar os resultados obtidos em malhas uniformes, traçamos gráficos da temperatura e pressão em relação à posição y normalizada na borda de fuga (células vizinhas à fronteira leste do domínio). Posição y normalizada é dada por:

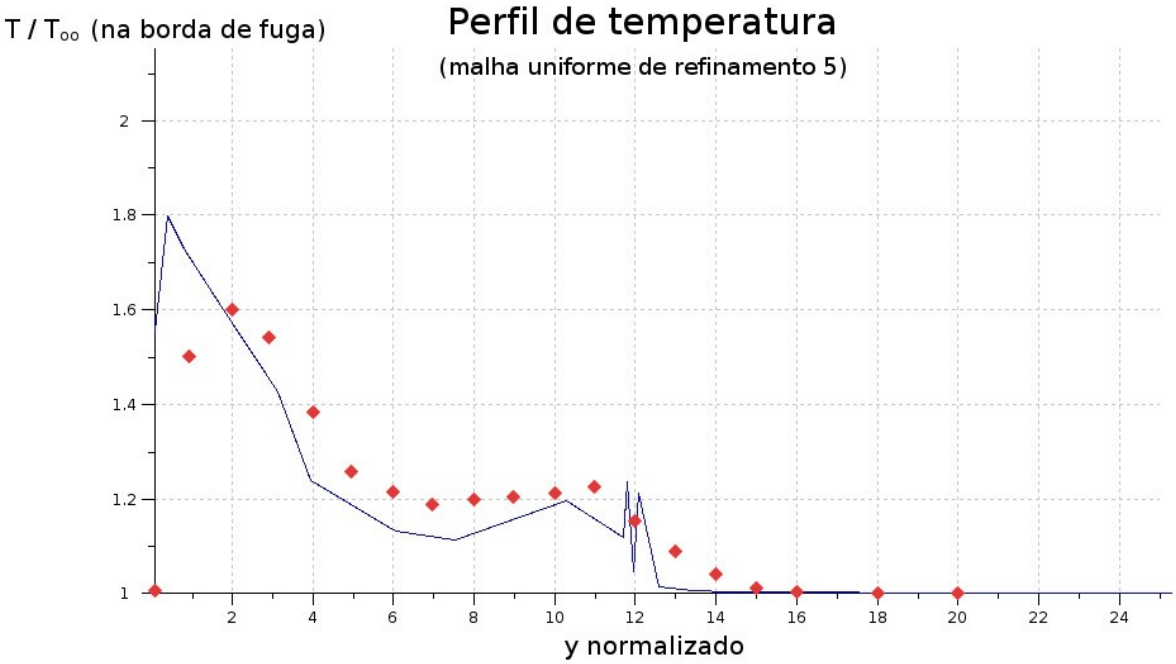
$$\bar{y} = \frac{y}{x}\sqrt{Re_x}$$

A seguir estão os gráficos para diferentes níveis de refinamento das malhas uniformes, onde as linhas azuis indicam a solução numérica obtida, e os pontos vermelhos são valores da solução exata. Em todos eles o incremento de tempo é 10^{-11}, o erro máximo é 10^{-5}, e tempo dado para a solução entrar na situação estacionária é da ordem de 10^{-8}.

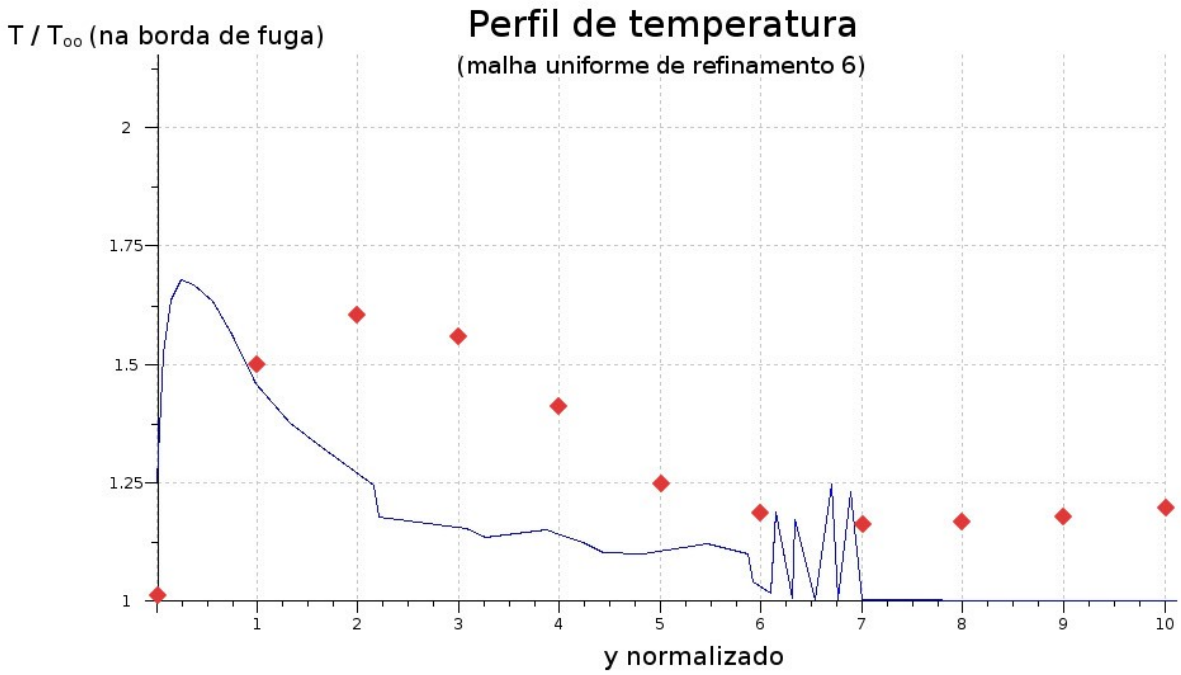
Gráficos da temperatura na borda de fuga



Qui Dez 7 12:32:57 2006

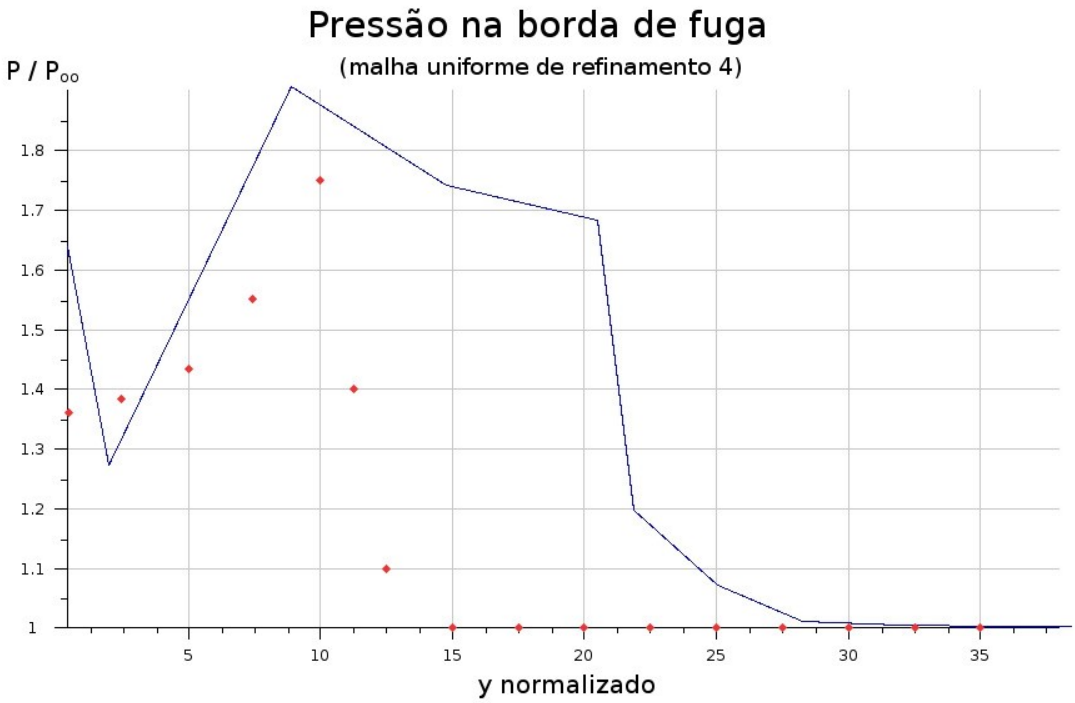


Qui Dez 7 12:40:40 2006



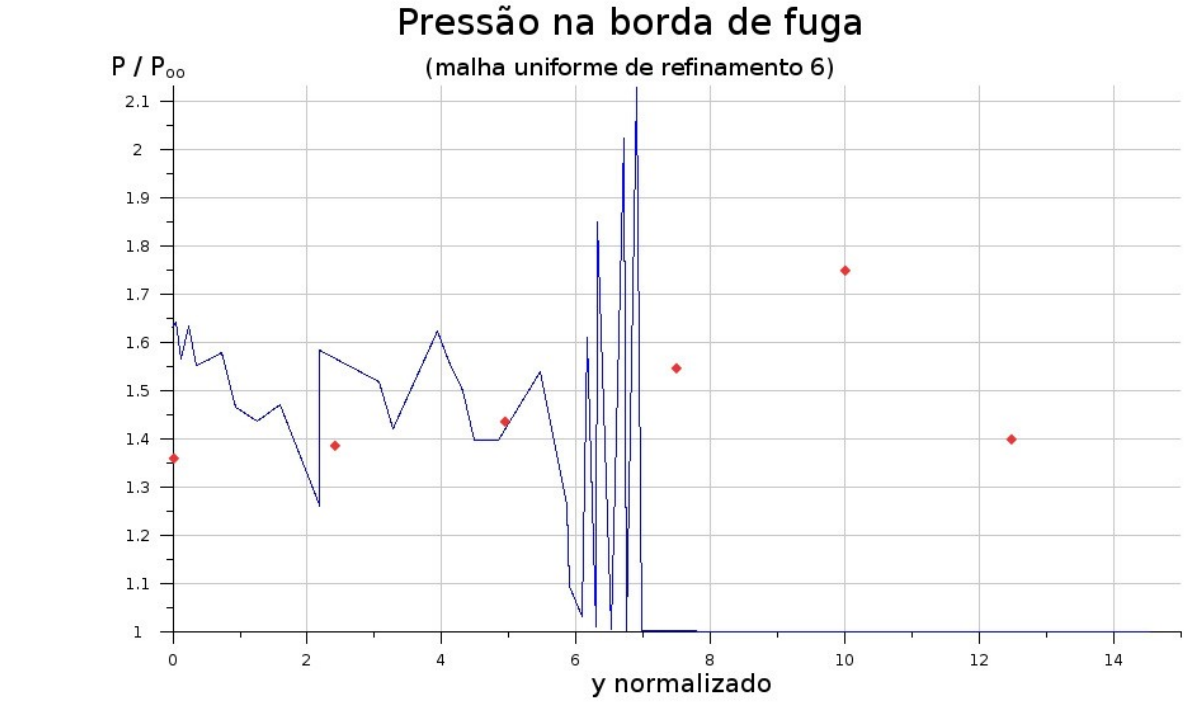
Qui Dez 7 13:01:37 2006

Gráficos da pressão na borda de fuga



Qua Dez 13 12:16:38 2006

Qua Dez 13 12:31:46 2006



Qua Dez 13 12:37:37 2006

Como pode ser facilmente visto, a estabilidade é menor a medida que o refinamento da malha é maior, devido a oscilações acentuadas próximas à onda de choque. Exceto a temperatura na malha de refinamento 5, o resultado raramente se aproxima da solução exata.

Interessante é notar que se variarmos o erro máximo na iteração de Picard, permitindo erros menores (por volta de 10^{-10}), depois de certo tempo o programa entra em um laço infinito (ou indefinidamente longo), logo o erro de alguma das variáveis nunca consegue ser pequeno o suficiente, o que indica que dentro da iteração de Picard a solução desestabiliza. Ou seja, a solução numérica que originalmente parecia estável na realidade começa a divergir dentro da iteração de Picard, mas é interrompida por causa do erro grande considerado. Para que a solução seja estável para vários valores do erro máximo, é necessário um incremento de tempo menor ainda, a fim de reduzir o quociente $\Delta t/\Delta L$.

O ideal para refinamento inicial da malha uniforme seria um valor bem alto, para poder suportar a diferença brusca na onda de choque. O inconveniente é que o gasto de memória seria extremamente elevado (já que não ocorre desrefinamento) e, para esse método, seria necessário um incremento de tempo extremamente pequeno, o que é inviável pela demora.

A respeito do refinamento adaptativo, conseguimos implementá-lo, considerando a diferença de pressão entre células vizinhas (Δp) como critério de refinamento. Malhas adaptativas funcionaram apenas para valores baixos dos níveis de refinamento, a saber: nível 1 como refinamento inicial, nível 3 como máximo ao refinar, nível 2 como mínimo ao desrefinar. Esta configuração faz a simulação ir do nível 1 ao 3 sem irregularidades, mas a malha é refinada até que ela toda se torna uniforme de nível 3. Valores maiores para o refinamento máximo seriam o ideal para suportar a onda de choque, mas isto logo faz o programa divergir mesmo com nível 4 para refinamento máximo.

3. Conclusão

Esse método desenvolvido com formulação explícita para as iterações de Picard certamente não é robusto. Mesmo para malhas uniformes ele impõe critérios de convergência bem restritos, além de divergir em malhas adaptativas, análogo à uma malha adaptativa forçada em um esquema numérico de diferenças finitas. Os resultados obtidos insinuam que uma formulação implícita na iteração de Picard poderia ser bem mais eficiente, a fim de obter independência do fator $\Delta t/\Delta L$ que fez até então uma grande diferença na simulação, impondo muitas dificuldades na escolha do incremento e do tamanho inicial das células. O próximo passo é desenvolver esse esquema implícito e considerá-lo para todos os próximos problemas semelhantes.