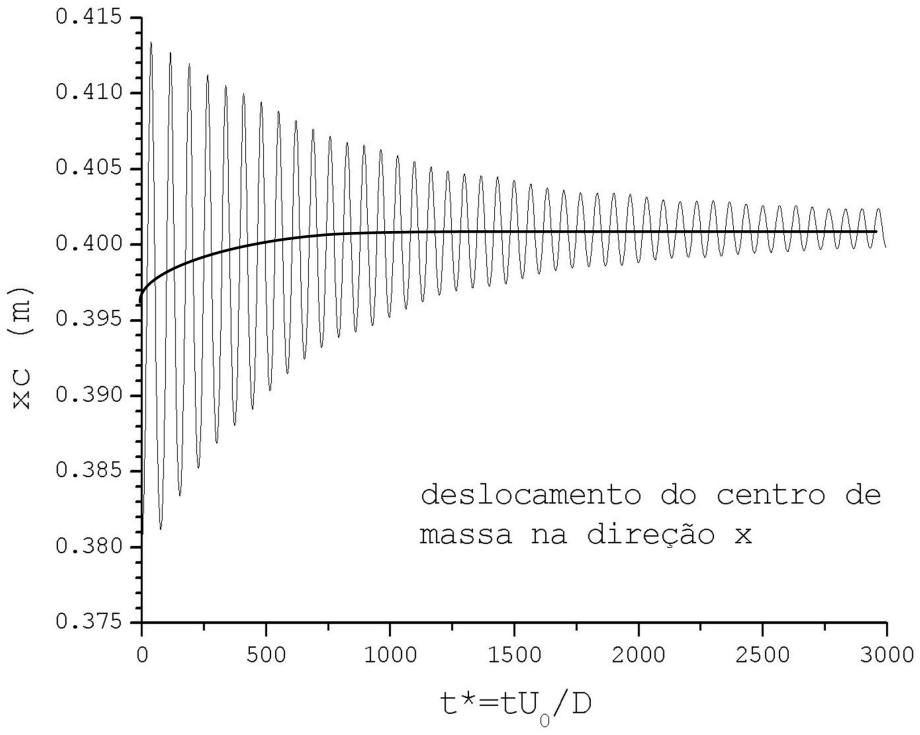


Figura 5.15 – Coordenada x do centro de massa em função do tempo adimensional.

Aos moldes do que foi realizado no capítulo anterior, foram inseridas sondas numéricas no domínio de cálculo de forma a monitorar a evolução temporal das flutuações das componentes das velocidades na esteira do escoamento. Entretanto, o esquema de disposição das estações foi alterado, pois se manteve apenas a estação A, localizada a 1,5D à jusante da parte posterior da esfera. Acrescentou-se, também, uma estação A*, que possui a mesma cota de A porém com liberdade de se mover acompanhando o movimento da esfera. Esta característica permitiu que a estação A* se mantivesse a uma distância constante 1,5D em relação à esfera. Na Fig. 5.17 encontra-se o desenho esquemático da disposição das sondas no escoamento. Assim como nos experimentos com a esfera estática, a disposição das sondas permaneceu com a sonda 1 na linha de centro da esfera, as sondas 2 e 3 deslocadas, respectivamente, -1,5D e +1,5D ao longo do eixo y e as sondas 4 e 5 dispostas ao longo do eixo z, deslocadas -1,5D e +1,5D, respectivamente, em relação à linha de centro. A freqüência de aquisição das sondas foi de 100Hz, com os passos de tempo sendo mantido constantes em $\Delta t = 10^{-3} s$. Além disso, a velocidade da corrente livre foi mantida em $U_0 = 0,4 m/s$.

Os resultados coletados para as sondas 2 e 3, monitorando a componente v, e os resultados para as sondas 4 e 5, monitorando a componente w, são apresentados na Fig. 5.18 (coluna da esquerda), com suas respectivas transformadas *FFT* (coluna à direita). É

interessante observar aqui a presença de freqüências mais altas na porção do domínio monitorada pela sonda 4. Enquanto as demais sondas apresentadas na Fig. 5.18 indicaram um pico de freqüência fundamental entre 1,47 Hz e 1,50 Hz, a sonda 4 detectou freqüências de 3,29 Hz. Porém, analisando-se as imagens das Figs. 5.8 e 5.9, pode-se perceber que esta região é atravessada por uma quantidade maior de estruturas de múltiplas escalas.

Essa hipótese é reforçada ao se notar o nível de energia associado às freqüências secundárias presentes no escoamento. É possível identificar uma delas que surge por volta de 13 Hz para todas as sondas. Para as sondas 2, 3 e 5, existe uma freqüência secundária de valor aproximado de 3 Hz (aproximadamente o mesmo valor da freqüência fundamental para a sonda 4) com intensidade semelhante à de 13 Hz. Para a sonda 4 a freqüência de 13 Hz adquire intensidade maior que nas outras sondas e, por sua vez, a freqüência menos energizada passa a ser aquela por volta de 1,5 Hz que era a mesma observada como principal nas demais sondas. Houve, então uma inversão nos valores das freqüências fundamentais no escoamento, quando se observam as diferentes sondas.

O número de *Strouhal*, $St = \frac{fD}{U_0}$, associado à estas freqüências foi calculado e assumiu os diferentes valores: 0,150 (para 1,50Hz), 0,149 (para 1,49 Hz), 0,147 (para 1,47 Hz) e 0,329 (para 3,29 Hz). Os três primeiros resultados são muito próximos dos apresentados na Fig. 4.54, extraída de Mittal e Najjar (1999), para esfera parada, que indicam aproximadamente St = *0,15.* Considerando-se os primeiros resultados como o número de *Strouhal* associado ao escoamento (St_1), o segundo número (St_2 = 0,329) apresentou um valor de 2 St_1 e, como pode ser visto na Fig. 5.18d, uma terceira freqüência começa a surgir de valor aproximado à 10 St_1. Estes valores indicam uma dinâmica do escoamento muito mais detalhada que não pode ser percebida ao se analisar qualitativamente as estruturas turbilhonares pela sua topologia.

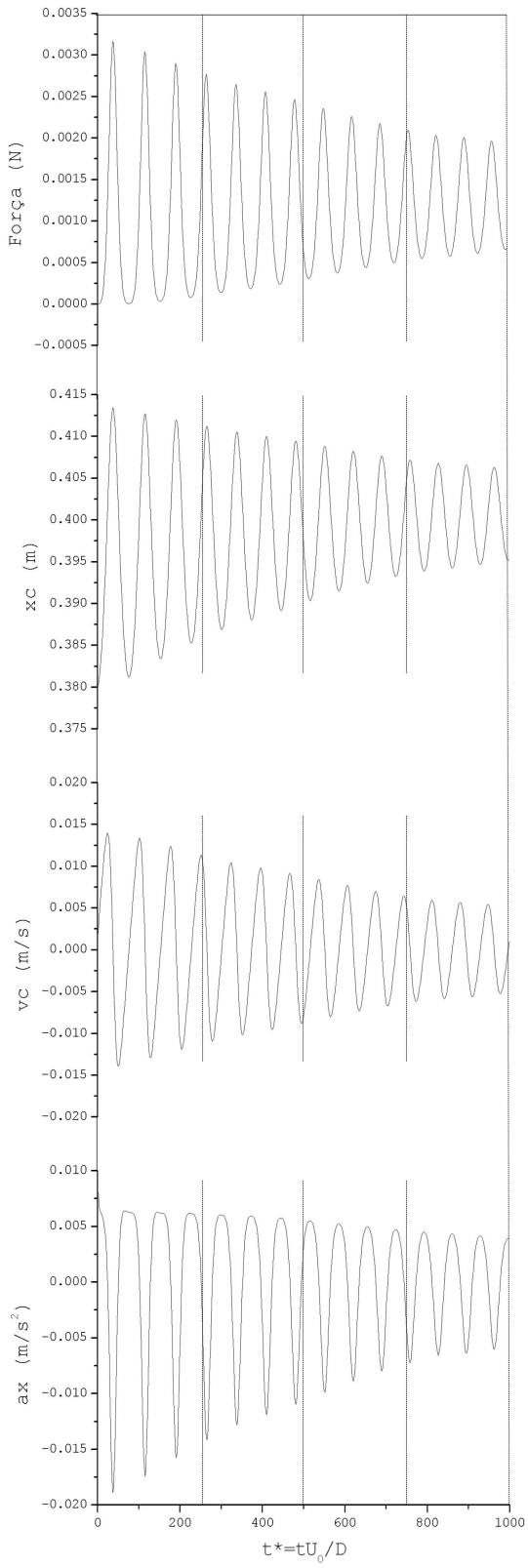


Figura 5.16 – Evolução dos valores da aceleração (ax), da velocidade (vx), deslocamento (xc) do centro de massa da esfera na direção do escoamento e força total nas molas.

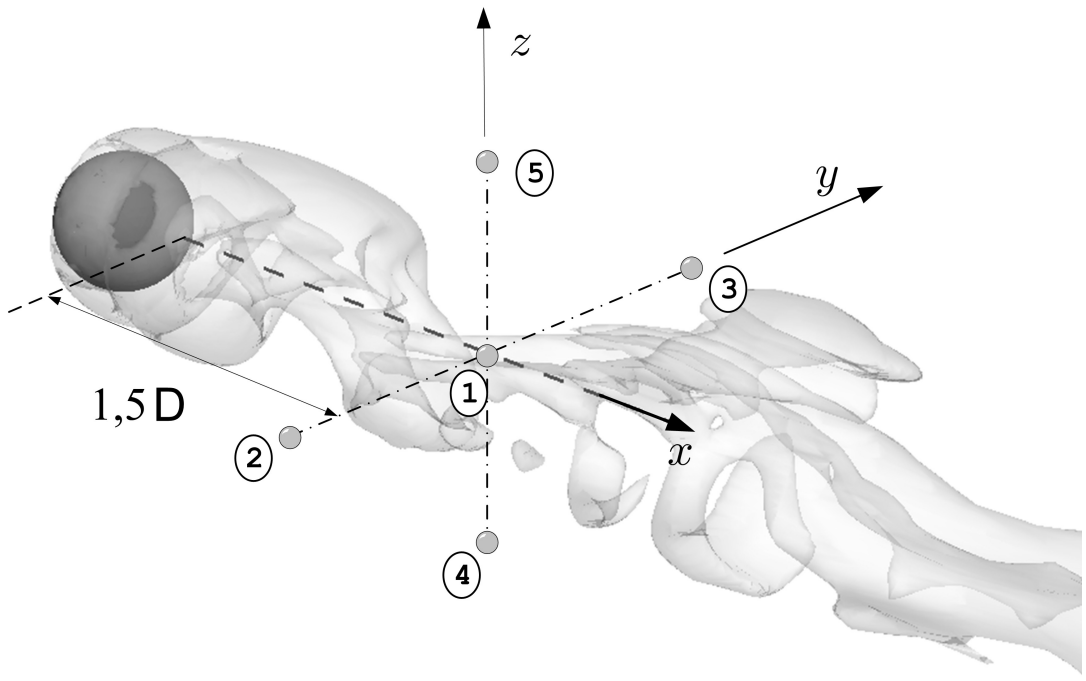


Figura 5.17 – Disposição das sondas numéricas ao longo da estação fixa A e da estação móvel A*.

Com a adoção de uma estação de sondas capaz de se mover com a esfera (estação A*), mantendo-se a uma distância permanente da mesma, procurou-se isolar os resultados das sondas numéricas dos efeitos provocados pela constante variação da velocidade relativa entre a esfera e o escoamento. Com efeito, esta estratégia capturou com mais clareza uma freqüência fundamental que aparecia apenas como secundária nas sondas estacionárias.

Assim, a partir dos resultados obtidos com a estação móvel A* (Fig. 5.19), pode-se perceber que os sinais das sondas das velocidades mostram o recrudescimento da freqüência mais alta de 13 Hz (10 St_1) registrada na Fig. 5.18d como secundária. Nos quadros da Fig. 5.19, pode-se notar o valor predominante de uma freqüência de ≈13 Hz, à exceção da Fig. 5.19d, cuja freqüência fundamental registrada foi de 19,85 Hz (12 St_1).

Por outro lado, os sinais registrados pelas sondas móveis estão influenciados pelo deslocamento do centro de massa da esfera pelos eixos y e z, o que afeta os resultados para as freqüências. Entretanto, os valores para as freqüências extraídas a partir dos históricos dos deslocamentos do centróide, mostrados na Fig. 5.20, indicam valores muito baixos e de pouca energia, o que minimiza os efeitos danosos provocados pelos movimentos transversais da esfera.

Além disso, as amplitudes dos deslocamentos nas direções transversais, como já mencionado, não ultrapassaram $1{,}4\cdot10^{-4}\,m$ na direção y e de $3{,}0\cdot10^{-5}\,m$ na direção z. Estes valores são, respectivamente, 14 e 67 vezes menores que as dimensões Δy e Δz dos volumes nessas regiões. Assim, na conversão do domínio contínuo para o domínio discreto (para a monitoração dos valores discretos das velocidades), sempre apontaram para a mesma coordenada discreta j e k para a coordenada y_i e z_i (posições atuais do centróide da esfera).

Procurou-se com o emprego de sondas móveis, mostrar as potencialidades da experimentação numérica na análise de problemas de fronteiras móveis. Como o problema estudado tem natureza inovadora, não se pretendeu evidenciar um fenômeno em particular com as sondas móveis, por desconhecer seus detalhes *a priori*.

Por fim, apresentam-se nas Figs. 5.21 a 5.23 resultados de trechos dos históricos de deslocamento do centróide da esfera. Estas figuras foram ajustadas para servirem não só para melhor analisar a dinâmica global do experimento mas, também, como comparação com a Fig. 5.3. Na Fig. 5.3, como comentado anteriormente, o deslocamento da esfera foi realizado pela imposição das componentes cartesianas da força exercida pelo fluido sobre ela durante uma simulação com o corpo estacionário.

A Fig. 5.21 mostra o trecho inicial ($0 < t^* < 300$) da simulação, a Fig. 5.22 apresenta um trecho intermediário ($1350 < t^* < 1650$) e a Fig. 5.23 a parte final do experimento ($2700 < t^* < 3000$). Cada um dos pontos discretos (representados por uma pequena esfera) indica uma coordenada x,y,z do centróide num determinado instante de tempo. As cores dos símbolos variam em função do tempo, indo do tom mais claro para o mais escuro à medida que este avança.

Observando-se os planos projetados (sub-índices *a, b* e *c*), nota-se que a dinâmica do movimento da esfera apresenta, inicialmente, uma amplitude de deslocamento quase uniforme nas duas direções coordenadas y e z, que representa um plano vertical normal à direção do escoamento. Em seguida, as amplitudes na direção y passam a serem maiores em relação às amplitudes da direção z. Uma das prováveis hipóteses para este comportamento pode ser o início mais caótico do escoamento, onde o processo de emissão de vórtices ainda não estava estabelecido, o que impulsionou deslocamentos iniciais de maiores amplitudes. Conseqüentemente, a energia acumulada nas molas tornou-se alta e, ao ser liberada provocou elevadas (mas uniformemente distribuídas) forças nas direções y e z sobre a esfera. Em seguida, os níveis energéticos no sistema esfera-molas foi sendo reduzido gradativamente, o que possibilitou que um outro detalhe passe a interferir na dinâmica: a constituição estrutural do modelo, onde os deslocamentos na direção y sofrem menor resistência do que os na direção z (onde os ângulos relativos às esferas são menores).

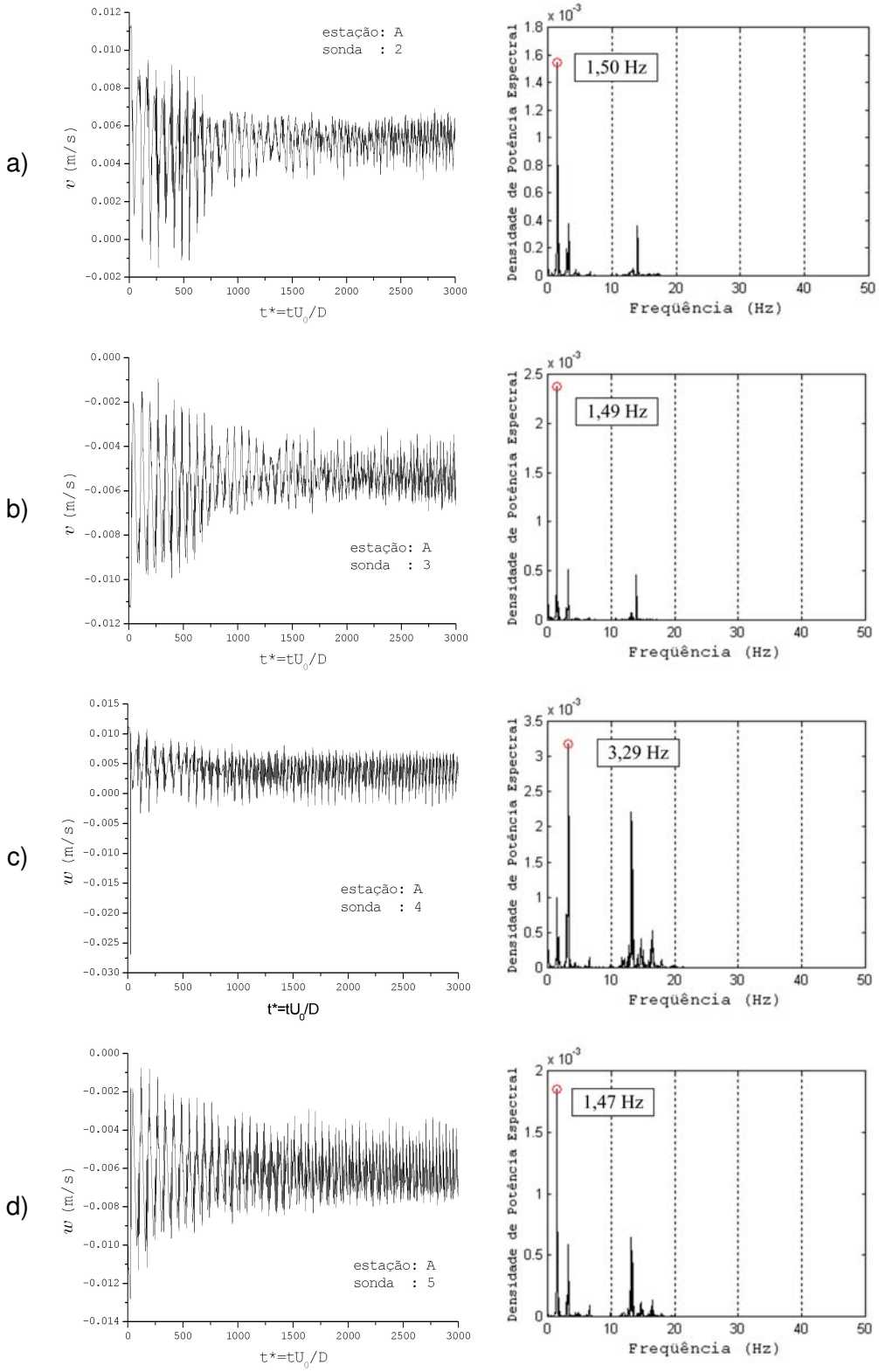


Figura 5.18 – Históricos dos sinais das componentes das velocidades (esq.) e *FFT* do sinal correspondente (dir.), para as sondas 2(a), 3(b), 4(c) e 5(d) posicionadas na estação fixa A.

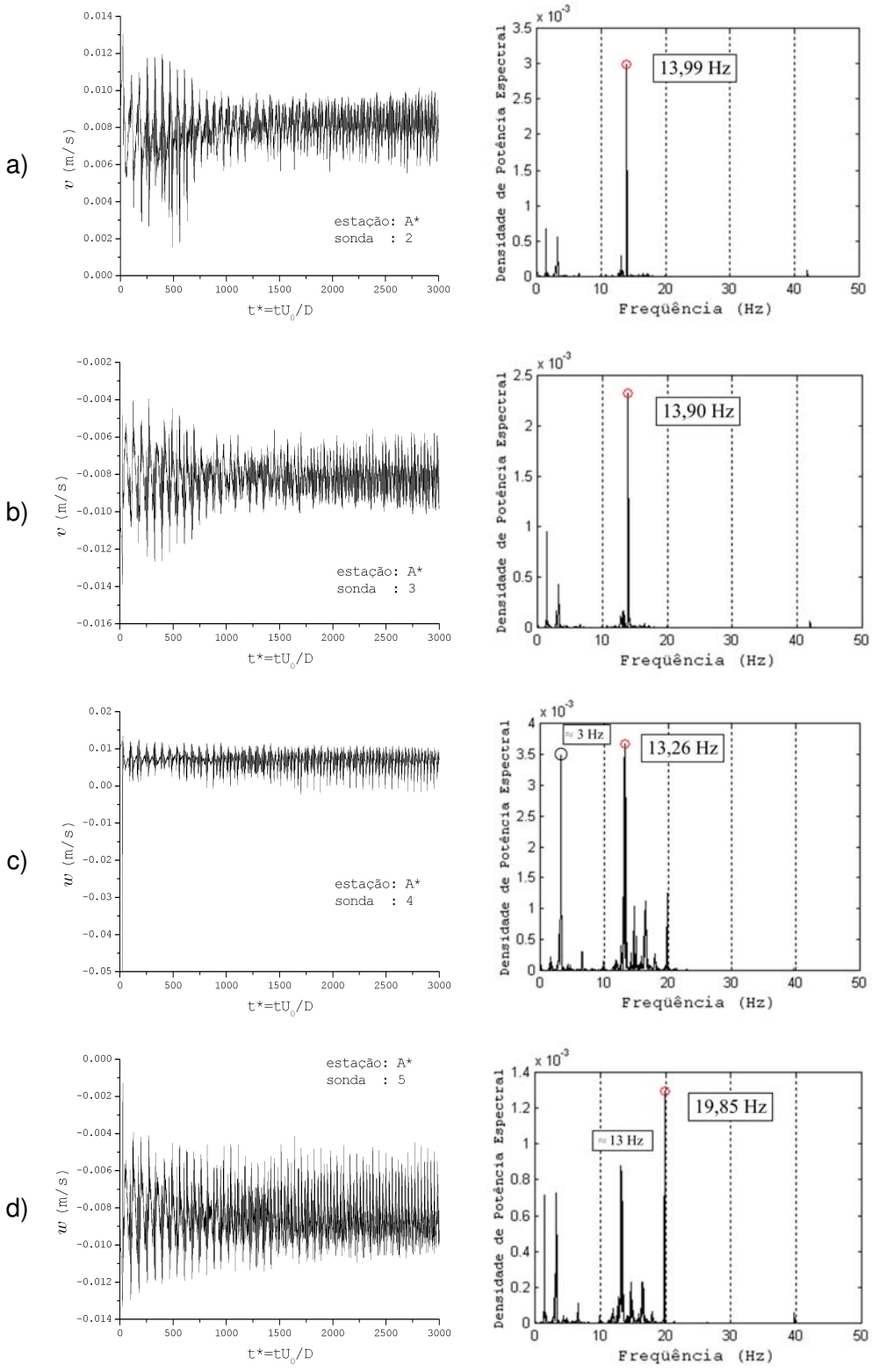
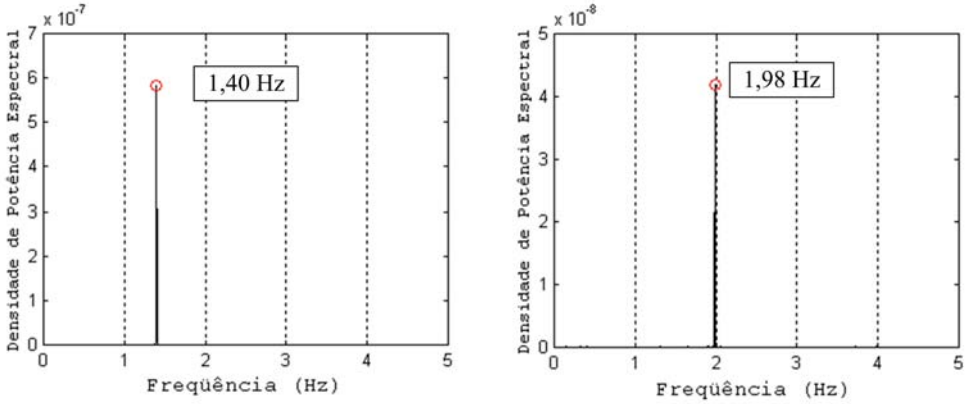


Figura 5.19 – Históricos dos sinais das componentes das velocidades (esq.) e *FFT* do sinal correspondente (dir.), para as sondas 2(a), 3(b), 4(c) e 5(d) posicionadas na estação móvel A*.



a) b)

Figura 5.20 – Resultados para a *FFT* dos sinais de deslocamentos do centro de massa na direção do eixo y (a) e do eixo z (b).

De uma forma geral, pode-se notar que a dinâmica de deslocamento experimentada pela esfera adquire um padrão mais ordenado à medida que a simulação avança no tempo. A amplitude dos deslocamentos foi bastante reduzida e o centróide da esfera passou a percorrer caminhos mais regulares. A Fig. 5.15, que representa todo o histórico do deslocamento do centro de massa na direção x, ajuda na compreensão global do processo. Pode-se perceber a diminuição gradual do deslocamento longitudinal da esfera, até que o mesmo atinja um padrão regular, oscilando em torno de uma média bem definida e num padrão temporalmente estacionário. Em particular, na Fig. 5.23b, plano alinhado com o escoamento, percebe-se um movimento helicoidal do centro de massa. Isto indica que a interação fluido-estrutura, para este número de Reynolds ($\mathrm{Re}_D = 400$), estabeleceu-se em um padrão relativamente organizado.

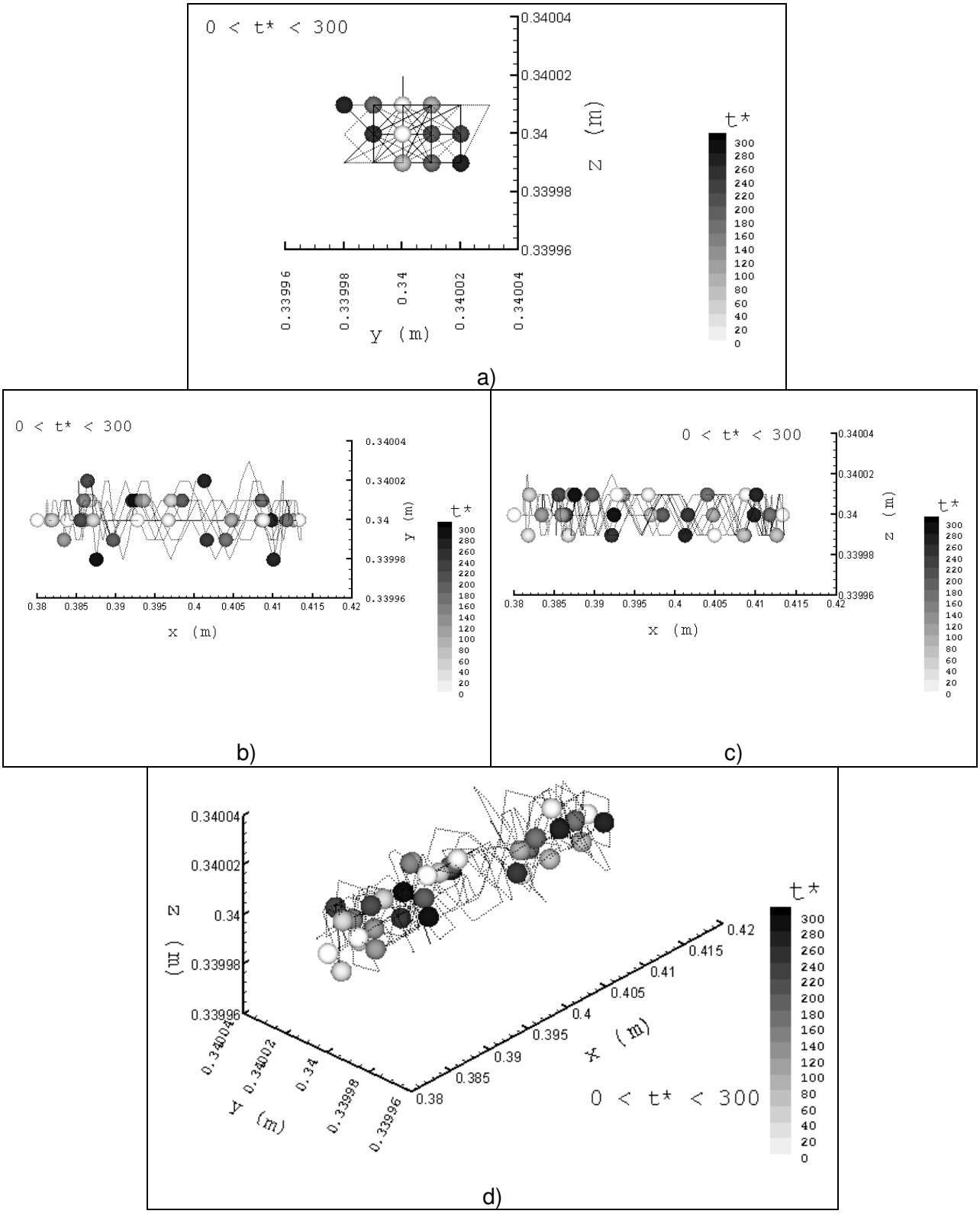


Figura 5.21 – Deslocamentos do centro de massa da esfera ao longo do plano xz (a), xy (b), yz (c) e em perspectiva (d), para os instantes de tempo $0 < t^* < 300$.

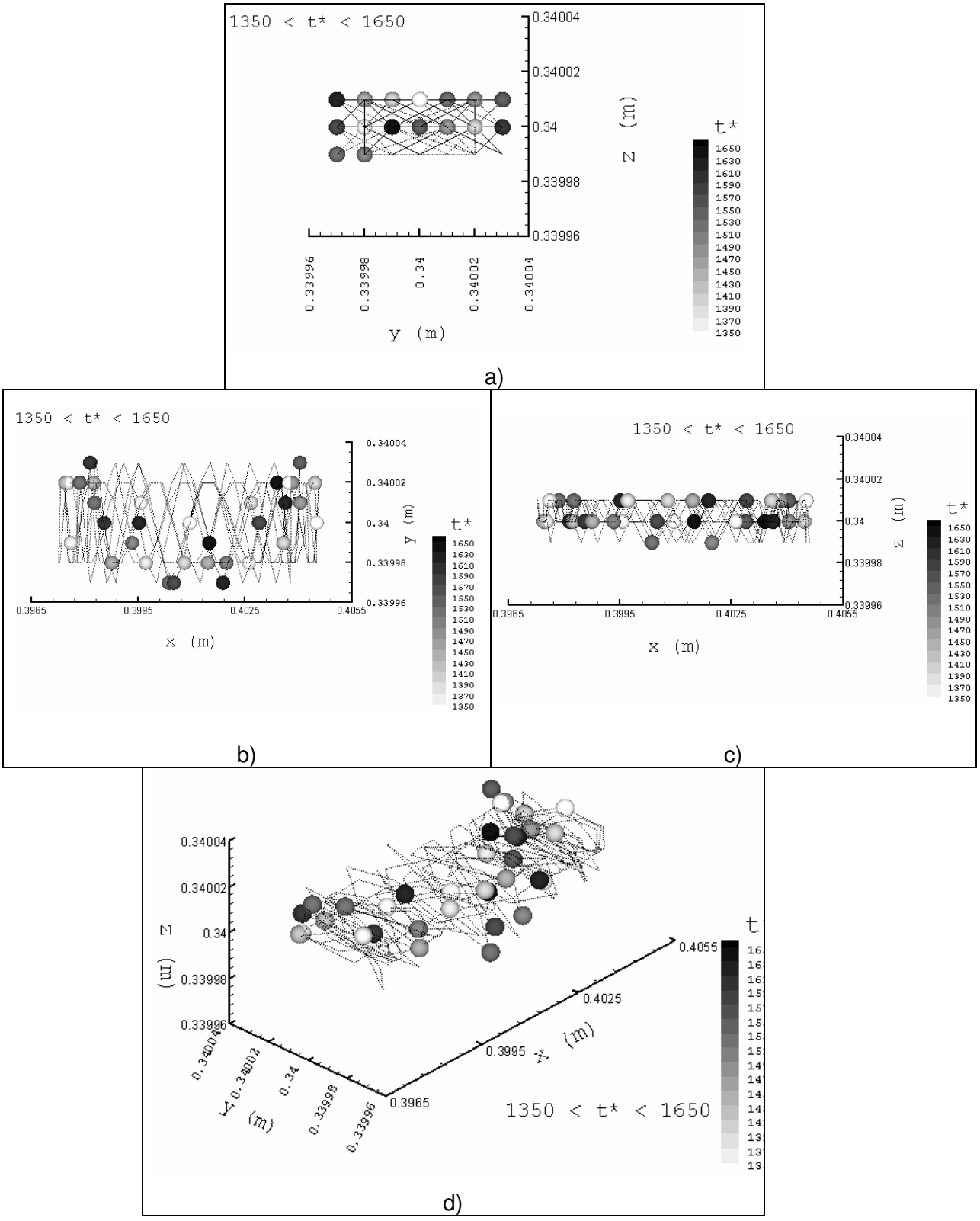


Figura 5.22 – Deslocamentos do centro de massa da esfera ao longo do plano xz (a), xy (b), yz (c) e em perspectiva (d), para os instantes de tempo $1350 < t^* < 1650$.

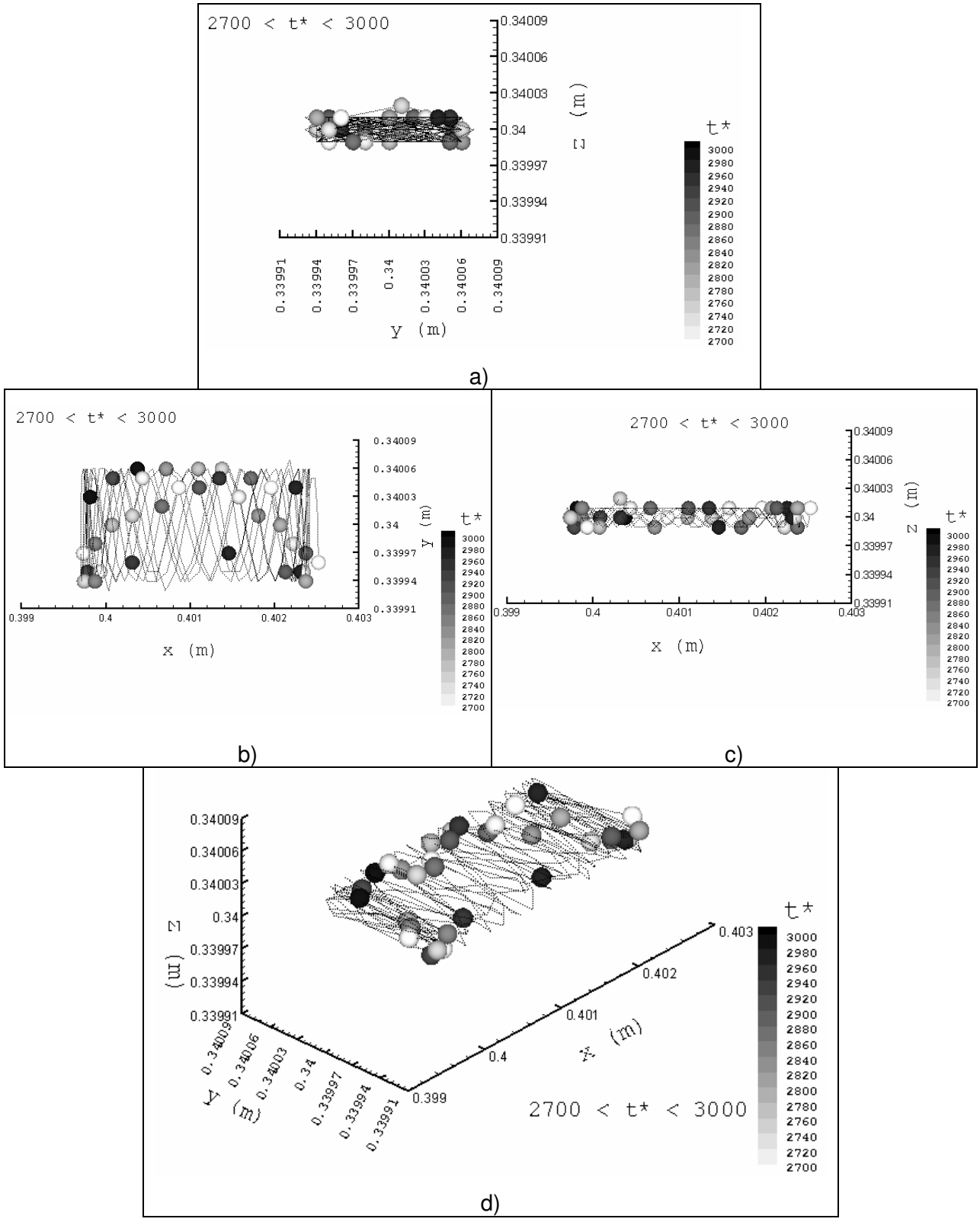


Figura 5.23 – Deslocamentos do centro de massa da esfera ao longo do plano xz (a), xy (b), yz (c) e em perspectiva (d), para os instantes de tempo $2700 < t^* < 3000$.

CAPÍTULO VI

Conclusões

No presente trabalho apresentou-se uma extensão do Modelo Físico Virtual, originalmente desenvolvido para domínios bidimensionais, para problemas tridimensionais. Para este propósito foi escrito um código computacional, com capacidade de processamento paralelo, para ser usado num cluster de computadores pessoais montado no LTCM no contexto do presente trabalho. O programa de base cartesiana foi validado tanto em sua capacidade de resolver os casos clássicos de escoamentos internos quanto em suas potencialidades de processamento distribuído.

Além disso, o código teve implementada a modelagem de turbulência pela Metodologia de Simulação de Grandes Escalas (LES) com modelo clássico de Smagorinsky e a equação da Energia, permitindo aumentar a gama de problemas físicos a serem simulados. Com relação à primeira implementação, a modelagem LES foi validada com casos de escoamento em degraus descendentes, obtendo-se bons resultados, tanto qualitativos quanto quantitativos, permitindo a captura de diversas estruturas turbilhonares presentes no escoamento. O uso do processamento paralelo foi igualmente bem sucedido, permitindo simular problemas empregando malhas refinadas. A equação da energia, embora implementada, não foi validada e contempla apenas a sua forma desacoplada das equações da quantidade de movimento.

Como primeiro caso teste para a metodologia de fronteira imersa, escolheu-se o escoamento ao redor de uma esfera estacionária a baixos números de Reynolds. Os resultados qualitativos obtidos apresentaram boa concordância com os resultados da literatura, com as linhas de corrente se desviando satisfatoriamente da superfície da esfera. As etapas do processo de transição do escoamento ao redor da esfera foram capturadas pelo código. De acordo com a literatura, a Re > *210-212* a bolha de recirculação à jusante da esfera torna-se assimétrica, dando início à emissão alterada de vórtices num padrão estacionário, que torna-se transiente a Re > *270-285*. Porém, as propriedades do escoamento mantêm a simetria com relação ao seu plano transversal. Os resultados, a Re = 3*00* mostram uma bolha de recirculação assimétrica, além da esteira dupla que se forma após a bolha de recirculação e a simetria planar. Ainda de acordo com a literatura, a partir de Re > *420* a simetria planar

desaparece e a $Re > 800$ o escoamento se torna completamente turbulento. Foi possível capturar as etapas do processo de transição dentro de uma faixa satisfatória de informações.

Os resultados para os números de *Strouhal* apresentaram algumas diferenças em relação à literatura. Entretanto, cabe salientar que mesmo os resultados consultados apresentaram discrepâncias entre si, dificultando estabelecer um parâmetro de comparação seguro. Por outro lado, os resultados para coeficientes de arrasto apresentaram ótima concordância com os da literatura. Grande parte dos bons resultados deveu-se à capacidade do método em manter a norma l_2 em valores razoavelmente baixos (nunca excedendo 10^{-2}).

Nos escoamentos a Reynolds mais elevados, sondas numéricas ajudaram a levantar os espectros de turbulência ao longo da esteira de vórtices evidenciando, quantitativamente, o processo de multiplicação de escalas das estruturas turbilhonares.

Devido às facilidades inerentes das metodologias de fronteira imersa, com respeito a deformações e movimentação da interface sólido/fluido, foi introduzido no código um modelo dinâmico composto de uma esfera sustentada por molas, como teste para um problema de interação fluido-estrutura. Procurou-se empregar a metodologia como ferramenta para ajudar a compreender, de forma qualitativa e quantitativa, a complexa relação mútua existente entre o processo de formação e emissão das estruturas turbilhonares e o balanço de forças no sistema dinâmico. É importante salientar que não se sabia, *a priori*, qual seria o comportamento deste sistema, pois não foi encontrado um similar, nem de natureza numérica, nem experimental, na literatura. Implementou-se, ainda, um conjunto de sondas numéricas capazes de acompanhar o movimento da esfera, buscando coletar informações adicionais a respeito do processo.

Dessa forma, o programa pode ser visto como uma ferramenta para futuros trabalhos envolvendo análises de escoamentos ao redor de geometrias complexas móveis, com facilidades de processamento paralelo. Como conclusão geral, com o presente trabalho pode-se avançar de forma significativa com o desenvolvimento de fronteira imersa, colocando-a como uma alternativa muito promissora para a análise numérica de problemas de interação fluido-estrutura.

Perspectivas Futuras

O presente trabalho deixa como perspectivas futuras os seguintes possíveis projetos:

- Realizar novos testes com o modelo de interação fluido-estrutura proposto, explorando diversos regimes de escoamento e configurações do modelo.
- Continuar o trabalho de implementação da modelagem da turbulência aplicada a escoamentos ao redor de geometrias imersas. Uma tese de doutorado já está em andamento com este tema a ser aplicado em domínios 2D.
- Implementar metodologias *multigrid* de aceleração da solução de sistemas lineares. Um módulo do presente código já foi escrito com esta finalidade, mas carece de testes adicionais.
- Estender as opções de condições de contorno nas fronteiras do domínio euleriano, ampliando as funcionalidades do programa.
- Implementar esquemas de discretização espacial compactos de ordens mais elevadas, buscando capturar um espectro mais amplo de estruturas no escoamento. Conseqüentemente, um estudo mais detalhado de sobrecarga na comunicação entre os processadores será necessário, devido ao *overlapping* maior entre os domínios.
- Estudar a possibilidade de se implicitar o termo da força de aceleração do Modelo Físico Virtual, buscando diminuir o valor da norma l_2 e, ao mesmo tempo, a dependência dos resultados das propriedades do escoamento com relação ao passo de tempo. Esta alteração poderá alterar o padrão heptadiagonal da matriz de coeficientes, o que exigirá a adoção de *solvers* capazes de lidar com sistemas esparsos.
- Estender a metodologia de paralelização também ao domínio lagrangiano, de forma a se ganhar maior autonomia no uso do *cluster*.
- Implementar o particionamento do domínio euleriano em topologias 2D e 3D de paralelização.
- Introduzir a transferência de calor ao modelo de fronteira imersa, aproveitando a equação da energia já implementada no atual código.
- Acrescentar um módulo de pré-processamento capaz de ajudar na construção das malhas do domínio euleriano e da importação de geometrias complexas em diferentes formatos, aproveitando os geradores de malhas de Elementos Finitos disponíveis.

- Estudar a viabilidade de se implementar, ao presente código, funcionalidades de malhas adaptativas, com balanço de carga em tempo de processamento. Com relação à primeira sugestão, uma tese de doutorado está em andamento, implementando malhas adaptativas para domínios bidimensionais.

Capítulo VII

Bibliografia

Armfield, S. e Street, R., 2004, “Modified fractional-step methods for the Navier-Stokes equations”, Australian and New Zealand Industrial and Applied Mathematics – ANZIAM 45.C362-C377.

Arruda, J., 2004, “Modelagem Matemática de Escoamentos Internos Forçados utilizando o Método da Fronteira Imersa e o Modelo Físico Virtual”, Tese de Doutorado, Universidade Federal de Uberlândia.

Arruda, J., Lima e Silva, A., Roma, A. e Silveira-Neto., A., 2004, “Simulação Numérica de Escoamentos sobre Cavidades Abertas Rasas utilizando o Método da Fronteira Imersa”, In Proceedings of the 12th Brazilian Congress of Thermal Sciences and Engineering – ENCIT.

Avancha, R. e Pletcher, R., 2002, “Large-Eddy Simulation of the Turbulent Flow Past a Backward-Facing Step with Heat Transfer and Property Variations”, International Journal of Heat and Fluid Flow 23, 601-614.

Baker, L. e Smith, B.J., 1996, “Parallel Programming”, McGraw-Hill.

Barkley, D., Gomes, M. e Henderson, R., 2002,”Three-dimensional Instability in Flow over a Backward-Facing Step”, Journal of Fluid Mechanics 473, 167-190.

Bathe, K. e Zhang, H., 2002, “A flow-condition-based interpolation Finite Element procedure for incompressible fluid flow”, Computer and Structures 80, 1267-1277.

Benhamadouche, S., Mahesh, K. e Constantinescu, G., 2002, “Collocated Finite-Volume scheme for Large-Eddy simulation on unstructured meshes”, In Center for Turbulence Research – Proceedings for the Summer Program 2002.

Campregher, R., 2002, “Simulação Numérica de Escoamentos Transicionais e Turbulentos ao redor de Geometrias Cartesianas”, Dissertação de Mestrado, Faculdade de Engenharia Mecânica – UNESP.

Campregher, R., Silveira-Neto, A, Marinho, W. e Mansur, S.S., 2004, “Numerical simulation of the Backward-Facing Step in a Beowulf-class cluster”, In Proceedings of the XXI ICTAM – International Congress of Theoretical and Applied Mechanics, Warsaw, Poland.

Campregher, R., Mansur, S.S. e Silveira-Neto, A., 2005, “Numerical simulation of the flow around a sphere using the Immerse Boundary Method for low Reynolds numbers”, In Proceedings of the DLES6 Workshop, Poitiers, France.

Chorin, A., 1967, “A numerical method for solving incompressible viscous flows problems”, Journal of Computational Physics 2, 12-26.

Comte, P., Silvestrini, J.H. e Bégou, P., 1998, “Streamwise vortices in Large-Eddy Simulations of mixing layers”, Eur. Journal Mech. B/Fluids 17 (4), 615-637.

Eaton, J. e Johnston, J., 1980, “Turbulent flow reattachment: an experimental study of the flow and structure behind backward-facing step”, Stanford University, Rep. MD-39.

Fadlun, E., Verzicco, R., Orlandi, P. e Mohd-Yusof, J., 2002, “Combined immersed-boundary Finite-Difference methods for three-dimensional complex flows simulations”, Journal of Computational Physics 161, 35-60.

Farhat,C., Lesoinne, M. e LeTallec, P., 1998, « Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces : Momentum and Energy conservation, optimal discretization and application to aeroeslaticity“, Comput. Methods Appl. Mech. Engrg. 157, 95-114.

Ferziger, J. e Perić, M, 2002, Computational Methods for Fluid Dynamics 3rd Ed.Springer Verlag, New York, USA.

Flynn, M.J., 1966, “Very high-speed computing systems”, Proc. IEEE 54 (12), 1901-1909.

Fornberg, B., 1988, “Steady viscous flow past a sphere at high Reynolds number”, Journal of Fluid Mechanics 190, 471-489.

Fortuna, A.O., 2000, “Técnicas computacionais para dinâmica dos fluidos”, Edusp.

Gilmanov, A., Sotiropoulos, F. e Balaras, E., 2003, “A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids”, Journal of Computational Physics 191, 660-669.

Glowinski, R., Pan, T. e Périaux, J., 1998, « Distributed Lagrange multiplier methods for incompressible viscous flow around moving rigid bodies”, Comput. Methods Appl. Mech. Engrg. 151, 181-194.

Goldstein, D., Handler, R. e Sirovitch, L., 1993, “Modeling a no-slip flow boundary with an external force field”, Journal of Computational Physics 150, 354-366.

Goldstein, D., Handler, R. e Sirovitch, L., 1993, “Direct numerical simulation of the turbulent flow over a modeled ribled covered surfaces”, Journal of Fluid Mechanics 302, 333-375.

Griffith, B. e Peskin, C., 2005, “On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems.”, Journal of Computational Physics 208, 75-105.

Guschin, V., Kostomarov, A., Matyushin, P. e Pavlyukova, E., 2002, “Direct numerical simulation of the transitional separated fluid flows around a sphere and a circular cylinder”, Journal of Wind Engineering and Industrial Aerodynamics 90, 341-358.

Haller, G., 2005, “An objective definition of a vortex”, Journal of Fluid Mechanics 525, 1-26.

Harlow, F. e Welsh, J., 1965, “Numerical calculation of time dependent viscous incompressible flow with free surface”, Phys. Fluids 8(4), 2182-2189.

Hayase, T., Humphrey, J. e Greif, R., 1992, “A consistently formulated QUICK scheme for fast and stable convergence using Finite-Volume iterative calculation procedures”, Journal of Computational Physics 98, 108-118.

Jeong, J. e Hussain, F., 1995, “On he identification of a vortex”, Journal of Fluid Mechanics 285, 69-94.

Johnson, T.A. e Patel, V., 1999, “Flow past a sphere up to a Reynolds number of 300”, Journal of Fluid Mechanics 378, 19-70.

Jovic, S. e Driver, D., 1994, “Backward-facing step measurement at low Reynolds number, Re = 5000”, NASA Technical Memorandum 108807.

Juric, D, 1996, “Computation of phase change”, PhD Thesis, Mechanical Engineering, University of Michigan, USA.

Karlsen, K., Lie, K-A, Natvig, J., Norhaug, H., e Dahle, H., 2001, “Operator splitting methods for systems of cconvection-diffusion equations: Nonlinear errors mechanisms and correction strategies”, Journal of Computational Physics 173, 636-663.

Kim, J., Kim, D. e Choi, H., 2001, “An immersed-boundary Finite-Volume method for simulations of flow in complex geometries”, Journal of Computational Physics 171, 132-150.

Kravchenko, A. G. e Moin, P., 1997, “On the effect of numerical errors in Large-Eddy Simulations of turbulent flows”, Journal of Computational Physics 131, 310-322.

Lai, M., 1998, “Simulations of the flow past an array of circular cylinders as a test of the Immersed Boundary Method”, PhD Dissertation, New York University.

Le, H, Moin, P. e Kim, J., 1997, “Direct numerical simulation of turbulent flow over a backward-facing step”, Journal of Fluid Mechanics 330, 349-374.

Lilek, Z. e Peric, M., 1995, “A fourth-order Finite-Volume method with colocated variable arrangement”, Computers and Fluids 24(3), 239-252.

Lima e Silva, A. L. F., 2002, “Desenvolvimento e implementação de uma nova metodologia para modelagem de escoamentos sobre geometrias complexas: método da fronteira imersa como Modelo Físico Virtual”, Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

Lima e Silva, A. L. F., Silveira-Neto, A. e Damasceno, J., 2003, “Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method”, Journal of Computational Physics 189, 351-370.

Lima e Silva, A. L. F., Silva, A. R. e Silveira-Neto, A., 2005, “Numerical simulation of two-dimensional complex flows over bluff bodies using the immersed boundary method”, Aceito para publicação no Journal of Fluid and Structures.

Mahmud, S., Das, P., Hyder, N., e Islam, A., 2002, “Free convection in an enclosure with vertical wavy walls”, Int. J. Therm. Sci. 41, 440-446.

Maliska, C.R., 1995, “Transferência de calor e mecânica dos fluidos computacional”, Rio de Janeiro, LTC Editora.

Marchi, C.H., 1993, “Esquemas de alta ordem para a solução de escoamentos de fluidos sem dispersão numérica”, Revista Brasileira de Ciências Mecânicas 15(3), 231-249.

Marinho, W., Campregher, R., e Silveira-Neto, A., 2004, “Three-dimensional heat transfer simulation as a benchmark for an in-house Beowulf-class cluster”, In Proceedings of the 10th Brazilian Congress of Thermal Sciences and Engineering – ENCIT.

Mittal, R. e Najjar, F.M., 1999, “Vortex dynamics in the sphere wake”, AIAA 99-3806.

Mohd-Yusof, J., 1997, “Combined immersed boundaries/B-splines methods for simulations of flows in complex geometries”, In CTR Annual Research Briefs, NASA Ames/Stanford University.

Muzaferija, S. e Peric, M., 1997, “Computational of free-surface flows using the Finite-Volume method and moving grids”, Numerical Heat Transfer, Part B 32, 369-384.

Oliveira, J., Lima e Silva, A., Guimarães, G. e Silveira-Neto, A., 2004, “Simulação numérica do escoamento a baixo Reynolds sobre o cilindro de diâmetro variável usando MFI/MFV”, In Proceedings of the 10th Brazilian Congress of Thermal Sciences and Engineering.

Oliveira, J., Lima e Silva, A., Souza, F., Guimarães, G. e Silveira-Neto, A., 2004, “Comparative analysis between different methodologies of turbulence modeling: URANS, DES e LES”, In 4º Escola de Primavera de Transição e Turbulência.

Patankar, S., 1980, “Numerical Heat Transfer and Fluid Flow”, Hemisphere Publishing Corporation, New York, USA.

Peric, M. Kessler, R. e Scheuerer, G., 1988, “Comparison of Finite-Volume numerical methods with staggered and collocated grids”, Computer and Fluids 16(4), 389-403.

Perron, S., Boivin, S. e Hérard, J., 2004, “A Finite-Volume method to solve the 3D Navier-Stokes equations on unstructured collocated meshes” Computer and Fluids 33, 1305-1333.

Peskin, C., 1972, “Flow patterns around heart valves: A numerical method”, Journal of Computational Physics 10, 252-271.

Peskin, C., 1977, “Numerical analysis of blood flow in the heart”, Journal of Computational Physics 25, 220-252.

Piller, M. e Stalio, E., 2004, “Finite-Volume compact schemes on staggered grids”, Journal of Computational Physics 197, 299-340.

Pitanga, M, 2002, “Construindo supercomputadores com Linux”, Brasport.

Ploumhans, P., Winckelmans, G., Salmon, J., Leonard e Warren, M., 2002, “Vortex methods for direct numerical simulation of three-dimensional bluff body flows: application to the sphere at Re = 300, 500, and 1000”, Journal of Computational Physics 178, 427-463.

Rhie, C e Chow, W., 1983, “Numerical study of the turbulent flow past an airfoil with trailing edge separation”, AIAA Journal, 21(11), 1525-1532.

Roma, A., Peskin, C. e Berger, M., 1999, “An adaptive version of the immersed boundary method”, Jornal of Computational Physics 153, 509-534.

Saiki, E. e Biringen, S., 1996, “Numerical simulation of a cylinder in uniform flow/; application of a virtual boundary method”, Journal of Computational Physics 123, 450-465.

Schneider, F.A. e Maliska, C.R., 2002, “Solução numérica de problemas convectivos-difusivos bidimensionais pelo método dos Volumes-Finitos usando malhas não estruturadas” IX Congresso Brasileiro de Engenharia e Ciências Térmicas – ENCIT.

Shirayama, S., 1992, “Flow past a sphere: topological transitions of the vorticity field”, AIAA Journal 30 (2), 349-358.

Silva Lopes, A. e Palma, J., 2002, “Numerical simulation of isotropic turbulence using a collocated approach and a non-orthogonal grid system”, Journal of Computational Physics 175, 713-738.

Silveira-Neto, A., 2003, “Introdução à Turbulência dos Fluidos”, Apostila do Curso de Pós-Graduação em Eng. Mecânica, LTCM/FEMEC/UFU.

Silveira-Neto, A., Grand, O., Metais, O. e Lesieur, M, 1993, “A numerical investigation of the coherent vortices in turbulence behind a backward-facing step”, Journal of Fluid Mechanics 256, 1-25.

Souza, A. M., 2005, “Análise da transição à turbulência em escoamentos de jatos circulares livres”, Tese de Doutorado, Universidade Federal de Uberlândia.

Souza, L., Mendonça, M., Medeiros, M., Kloker, M., 2002, “Three-dimensional code validation for transition phenomena”, III Escola de Transição e Turbulência – Florianópolis, SC.

Spalart, P.R., 1988, “Direct simulation of a boundary layer up to $Re_\theta = 1410$”, Journal of Fluid Mechanics 187, 61-98.

Spode, C., Campregher, R. e Silveira-Neto, A., 2005, “Parallel simulation of turbulent flow in a backward-facing step”, Aceito para publicação no 18th International Congress of Mechanical Engineering – COBEM, Ouro Preto, MG, Brasil.

Spode, C., Campregher, R. e Silveira-Neto, A., 2004, “Simulação numérica 3D do escoamento sobre um degrau descendente”, In Proceedings of the 14th Simpósio do Programa de Pós-Graduação em Engenharia Mecânica – POSMEC, Uberlândia, MG.

Subramanian, R. S., 2003, “Drag on a sphere”, Notas de aula do curso de Mecânica dos Fluidos no Departamento de Engenharia Química da Clarkson University, New York, http://www.clarkson.edu/subramanian/ch301/notes/index.htm. Acesso em: 2 set. 2005.

Tomboulides, A. G. e Orszag, S., 2000, “Numerical investigation of transitional and weak turbulent flow past a sphere”, Journal of Fluid Mechanics 416, 45-73.

Unverdi, S. e Tryggvason, G., 1992, “A front-tracking method for viscous, incompressible, multi-fluid flows”, Journal of Computational Physics 100, 25-37.

Van Doormal, J. e Raithby, G., 1984, “Enhancements of the simple method for predicting incompressible fluid flows”, Numerical Heat Transfer 7, 147-163.

Verteeg, H. e Malalasekera, W., 1995, “An introduction to computational fluid dynamics”, Prentice Hall.

Vilaça, A., Oliveira, O., Lima e Silva, A. e Silveira-Neto, A., 2004, “Modelagem matemática e simulação numérica do escoamento sobre uma partícula em queda livre”, Submetido ao XXXI ENEMP – Congresso Brasileiro de Sistemas Particulados.

White, F., 1991, “Viscous Fluid Flow”, McGraw-Hill, New York, USA.

Yu, Z., 2005, “ A DLM/FD method for fluid/flexible-body interactions”, Journal of Computational Physics 207, 1-27.

ANEXO I

Processamento Paralelo

Em um relatório preparado pelo governo britânico nos fins dos anos 40, concluiu-se que as necessidades computacionais de todo o país seriam satisfeitas por dois ou, talvez, três computadores. Naqueles tempos, os computadores eram empregados apenas para cálculos de balística, não sendo ainda utilizados para ciência e engenharia de uma forma mais ampla. Mesmo algum tempo após essa época, o potencial emprego de computadores ainda era subestimado: o prospecto inicial da *Cray Research* previa um mercado para, no máximo, 10 supercomputadores.

A maioria das simulações de problemas reais em engenharia necessita de domínios tridimensionais para serem bem representativas e, não raro, envolvem geometrias de topologias complexas. Assim, soluções numéricas dos fenômenos associados requerem, em grande parte, o emprego de enorme quantidade de células computacionais para serem atingidas com precisão aceitável. A tecnologia de processamento paralelo foi movida amplamente por suporte governamental, visando resolver grandes problemas numéricos. Existia um enorme interesse em processamento de sinais dos radares "*phased-arrays*" e de dados vetoriais de sonares (a tecnologia *phased* de radares permite vasculhar diferentes regiões simultaneamente, pois não possuem antenas móveis). Por este motivo, a primeira arquitetura de supercomputadores foi a de máquinas vetoriais (ou seja: *arrays*). Houve, também, um esforço no sentido de aumentar o fluxo de pesquisa simultânea em grandes bancos de dados. Durante muito tempo, a única opção disponível aos pesquisadores era o emprego de supercomputadores caros e, freqüentemente, de acesso restrito, devido a estratégias comerciais e/ou reservas de mercado. Felizmente, a evolução dos computadores pessoais (Personal Computers - PC) deu-se a taxas muito maiores do que os computadores de grande porte, estendendo o leque de opções em recursos. Cabe salientar, ainda, que a potência dos computadores está crescendo, desde 1945, a uma taxa de 10 vezes a cada 5 anos. Porém, mesmo com o crescente aumento da capacidade de processamento, comparável àquela disponível somente aos supercomputadores de poucos anos atrás, os microcomputadores atuais ainda não são capazes de armazenar e processar todos os dados necessários que são gerados em grande parte dos problemas reais.

Pode-se definir processamento paralelo como o emprego de mais de uma CPU (*Central Processing Unit* - Unidade de Processamento de Dados) trabalhando em conjunto para resolver um único problema. Esta definição, apesar de precisa, abrange desde o uso de dois computadores pessoais interligados por uma rede local, até milhares de processadores unidos formando um supercomputador paralelo. Porém, todo o embasamento teórico para o emprego de processamento paralelo já havia sido proposto anos antes da efetiva construção dos supercomputadores e dos clusters de PCs. Em seus artigos de 1966 e 1972, Michael Flynn propôs uma classificação para as arquiteturas de processamento paralelo, criando o que se conhece hoje por Taxonomia de Flynn (Flynn, 1966).

Segundo o autor, a configuração mais simples possível é a **Single Instruction** / **Single Data - SISD**, na qual um processamento serial convencional é realizado, por exemplo, em um PC convencional. *Single Instruction* significa que apenas uma instrução é realizada pelo processador por ciclo de *clock* e *Single Data* significa que existe apenas uma única entrada de dados a ser processada durante este mesmo ciclo, como ilustrado no exemplo da figura abaixo.

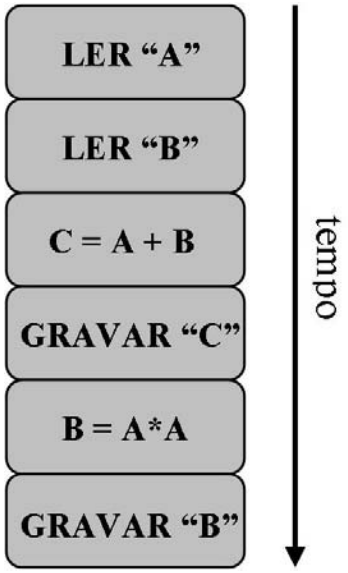


Figura A1.1 – Arquitetura Single Instruction / Single Data (SISD).

A arquitetura paralela **Single Instruction** / **Multiple Data - SIMD** pode ser entendida como um paralelismo de dados, onde uma única instrução é executada paralelamente utilizando vários dados, daí a expressão *Multiple Data*. Esta classificação abrange a tecnologia MMX (**M**ulti**M**edia e**X**tension) de alguns processadores modernos e também os processadores vetoriais do tipo CRAY.

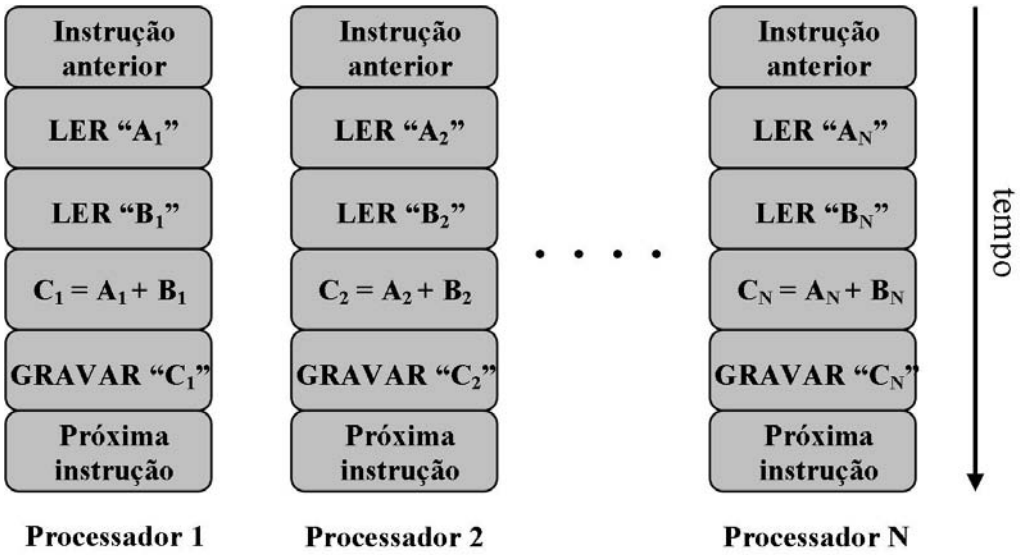


Figura A1.2 – Arquitetura Single Instruction / Multiple Data (SIMD).

Existe, também, uma configuração de cunho mais teórico, que seria a **Multiple Instruction** / **Single Data - MISD**, onde se encaixariam as máquinas capazes de realizar várias instruções (ou seja: *Multiple Data*) sobre um único dado. Não há registro de máquinas operando segundo esta arquitetura. Entretanto, poder-se-ia imaginar um procedimento de quebra de um determinado código criptográfico sendo realizado, ao mesmo tempo, por um grupo de processadores. Cabe salientar que este problema poderia ser resolvido, também, com processamento paralelo em outras arquiteturas.

Por fim, a arquitetura **Multiple Instruction** / **Multiple Data - MIMD** é caracterizada por ter cada processador agindo independentemente (*Multiple Instructions*) sobre dados diferentes (*Multiple Data*). Os processadores se comunicam usando uma rede que os permite compartilhar dados e sincronizar os cálculos. São raros os problemas nos quais não seja necessária nenhuma comunicação ou sincronismo entre os processadores. Um problema que apresente esta característica é denominado "*embarrassingly parallel*".

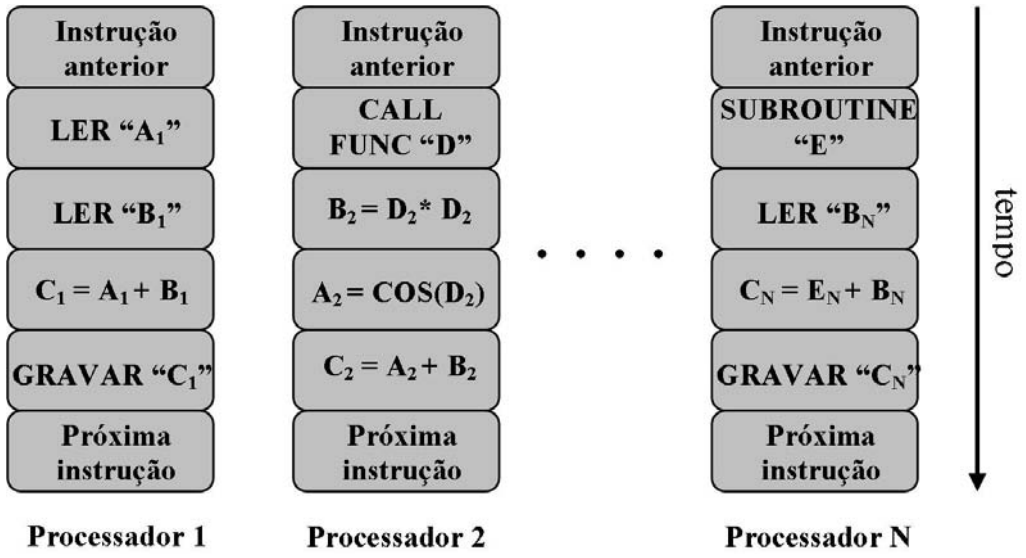


Figura A1.3 – Configuração Multiple Instruction / Multiple Data (MIMD).

É interessante comentar aqui que, com o advento de novas tecnologias de processadores, a Taxonomia de Flynn já não é capaz de descrever todas as arquiteturas possíveis. Além disso, a classificação MIMD acima engloba uma grande variedade de máquinas, sendo necessária uma triagem adicional que usa, para critério de seleção, a forma como a memória central está fisicamente organizada e o tipo de acesso que cada processador tem à esta memória. Assim, a arquitetura MIMD pode ser subdividida em Memória Distribuída, onde cada processador possui sua própria memória e realiza operações sobre ela e *Memória Compartilhada*, na qual existe uma única memória que é usada por todos os processadores.

A1.1 Clusters de computadores pessoais

A aproximação inicial em processamento paralelo era dividir, em teoria, um problema de tamanho **N** em um número também **N** de processadores (Baker e Smith, 1996). Infelizmente, esta estratégia mostrou-se equivocada ao se deparar com problemas reais do cotidiano. Estas idéias foram formuladas com base em máquinas idealizadas PRAM (*Parallel Random Access Machine*). Nestas máquinas, de memória compartilhada, todos os processadores têm acesso a um número infinito de memória. Além disso, todas as manipulações sobre a memória, tanto locais quanto remotas, assumem ter o custo de uma unidade. Entretanto, em computadores reais, o custo de acesso a uma memória remota é milhares de vezes maior que o acesso a uma memória local (gerando *overhead*). Esta dificuldade mudou o foco dos cientistas de *software* e de *hardware* para uma granulometria mais grosseira (ou seja, menos máquinas).

Durante a década de 90, a tecnologia empregada na troca de mensagens entre computadores adquiriu índices de confiabilidade e compatibilidade que permitiram o surgimento de agrupamentos (*clusters*) de computadores dedicados à execução de uma determinada tarefa. Tais características, aliadas à diminuição dos custos do equipamento, impulsionaram o estudo da viabilidade do emprego de clusters, popularizando a tecnologia e fomentando a pesquisa de novas arquiteturas e ferramentas de paralelização. O mercado atual de PCs é maior que o mercado de *workstations*, empurrando os preços dos equipamentos para baixo e, ao mesmo tempo, aumentando sua performance.

Deve-se atentar, ainda, para a possibilidade de construção de clusters empregando tanto máquinas homogêneas (quando todas possuem as mesmas características e estão interligadas pela mesma rede) quanto máquinas heterogêneas. Evidentemente, existem vantagens e desvantagens em se empregar cada uma das configurações. Planejar a distribuição de carga em um ambiente homogêneo requer menos preocupação com a quantidade de carga que cada nó deve receber. *Clusters* homogêneos tendem a serem mais rápidos que os heterogêneos, pois não há a necessidade de conversão de dados. Entretanto, mais cedo ou mais tarde, no momento em que um cluster homogêneo for ampliado, certamente ele se tornará heterogêneo, pois encontrar máquinas exatamente iguais é muito difícil, devido ao avanço tecnológico atual.

A quantidade de domínios que vêm se beneficiando do emprego de processamento paralelo, e aqueles que poderiam se beneficiar, cresce constantemente. Alguns exemplos a serem citados incluem os servidores de internet (onde *clusters* podem distribuir a carga e aumentar a capacidade de resposta), bancos de dados (redução do tempo nas pesquisas intensivas), computação gráfica (diminuição do tempo de renderização de imagens), inteligência artificial (reconhecimento de padrões de voz, visão artificial, etc.), engenharia genética, astrofísica (simulação da formação de galáxias e dinâmica estelar), meteorologia, pesquisa militar (simulação de explosões nucleares, processamento de sinais de radares, geração automática de mapas, etc.) e diversas outras aplicações.

Algumas vantagens de se empregar clusters de computadores são:

- Capacidade de se resolver problemas complexos, diminuindo o tempo de resolução;
- Possibilidade de adicionar novos componentes à medida que cresce a demanda por poder computacional;
- Aumento da confiabilidade do cluster em caso de falha em algum equipamento;
- Excelente relação custo/benefício;
- Uso de software livre e de hardware convencional, caracterizando independência de fornecedores.

Pode-se ainda, dividi-los em clusters **classe I**, que são inteiramente construídos a partir de componentes de *hardware* e de *software* disponíveis no mercado comum e empregando tecnologias padronizadas (como IDE, SCSI e *Ethernet*) e, também, os *clusters* **classe II**, que podem empregar programas e equipamentos feitos sob encomenda, visando aumentar sua performance.

A1.2 Cluster *Beowulf*

Beowulf foi um dos primeiros poemas da língua inglesa escrito, provavelmente, por volta do ano 1000 DC e conta a história de um herói (*Beowulf*), dotado de grande coragem e força, que enfrentou um monstro chamado *Grendel*. A aventura se passa numa terra onde, hoje, se encontra o sul da Suécia e, mesmo muito antes de ser escrita, já era conhecida e cantada pelos menestréis.

O projeto original de um cluster *Beowulf* teve início ao final de 1993 no CESDIS (*Center of Excellence in Space Data and Information Science*), um centro de pesquisa localizado no *Goddard Space Fligh Center* in Greenbelt, Maryland (EUA) e financiado em parte pela divisão ESS (*Earth and Space Sciences*) da NASA. Naquela época, os pesquisadores Donald Becker e Thomas Sterling começaram a esboçar um protótipo de cluster que pudesse ser montado a baixo custo e com componentes comuns. No ano seguinte, o projeto foi executado com a montagem do primeiro *cluster* composto de 16 processadores DX4, conectados por uma rede Ethernet de 10MBits/s. Este primeiro conjunto custou US$ 40.000,00 e atingiu a marca de 70 megaflops (o que não era pouco, quando comparado aos pequenos supercomputadores disponíveis). O sucesso deste trabalho logo se espalhou por outras unidades da NASA e para outras instituições de ensino e pesquisa pelos EUA e em todo o mundo.

É interessante comentar que *Beowulf* representa mais uma "filosofia" do que um tipo particular de *cluster*. Um *cluster* para ser considerado de classe *Beowulf* deve atender à algumas características diferenciadas (Pitanga, 2002):

- Nenhum de seus componentes deve ser feito sob encomenda, todos devem ser adquiridos no comércio convencional;
- Independência de fornecedores de *hardware* e *software*;
- Periféricos de fácil atualização, de forma que, para aumentar a capacidade do *cluster*, basta acrescentar mais processadores e/ou fazer um *upgrade*;
- *Software* livre e de código aberto;
- Uso de ferramentas distribuídas livremente com nenhuma ou mínimas alterações;
- Retorno à comunidade do projeto, onde as melhorias obtidas por um determinado grupo possam ser partilhadas por todos.

Um esquema de um cluster de classe *Beowulf* empregado no presente trabalho pode ser visto na Fig. A1.4. Atualmente, conta-se com 10 máquinas ligadas por uma rede de 1GBps sendo que, cada uma, conta com um processador Intel® Pentium IV de 2.8GHz, 1.5GBytes de memória DDR (*Double Data Rate*) RAM, 80GBytes de HD e rodam o sistema operacional Linux. O conjunto conta, ainda, com uma unidade KVM (*Keyboard*, *Video*, *Mouse*) para auxiliar no gerenciamento do equipamento e No-breaks como fonte ininterrupta de energia.

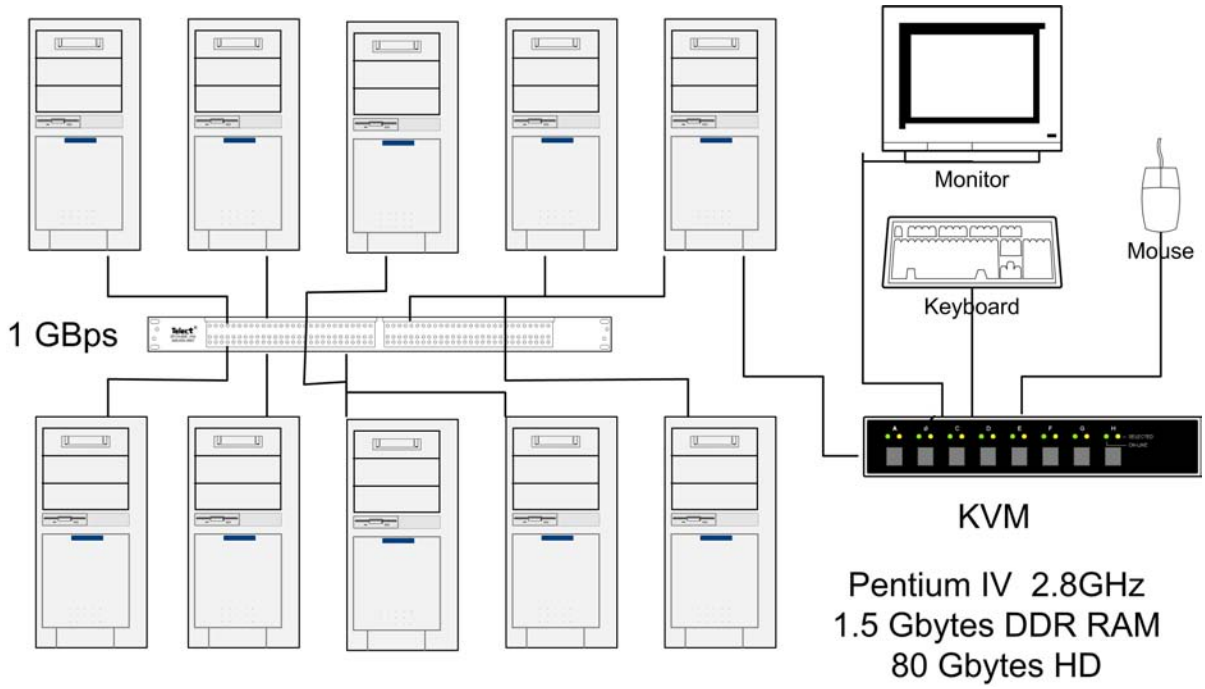


Figura A1.4 – Desenho esquemático representando o cluster *Beowulf* empregado no presente trabalho.

A1.3 Metodologias de Paralelização

A forma como o problema contido no domínio original é subdivido entre os diversos subdomínios é uma decisão que pode refletir de maneira decisiva no desempenho do programa paralelo. Alguns tipos de problemas não apresentam grandes dificuldades na paralelização, pois não há a necessidade de se trocar informações entre os subdomínios. A avaliação de uma integral ou, de outra forma, o cálculo da área sob uma curva, ao ser executado paralelamente é um exemplo clássico desta classe de problema, pois os resultados obtidos em cada processador são independentes uns dos outros, sendo agrupados somente ao final dos seus resultados individuais, para obter a área total (e, é claro, o resultado da integral). Tais tipos de problemas são conhecidos como *embarrassingly parallel*.

Entretanto, para a grande maioria dos casos, os problemas são resolvidos ao mesmo tempo em todos os processadores, mas não de forma independente, ou seja, para se obter a solução em um subdomínio, é necessário utilizar as informações armazenadas nos outros subdomínios. Esta troca de informações ou, mais especificamente, troca de mensagens é uma tarefa fundamental no sucesso de um processamento paralelo demandando enorme tempo no projeto de um código numérico.

Pode-se efetuar a divisão (também conhecida como particionamento) do problema a partir da decomposição do domínio original (*domain decomposition*) em outros subdomínios menores e/ou a divisão das tarefas, ou seja, das instruções do código a serem efetuadas no programa original (*functional decomposition*). A primeira alternativa é a mais empregada em sistemas de memória distribuída, enquanto que a segunda é mais utilizada em sistemas de memória compartilhada. Entretanto é possível, e comumente utilizado pelos programadores, empregar ambas abordagens a um mesmo problema. Porém, somente a decomposição do domínio foi abordada no presente trabalho.

Para uma análise do efeito da subdivisão do domínio sobre o desempenho do algoritmo, considerou-se um domínio cúbico tendo um número i de células em cada aresta. Pode-se dizer, assim, que o cubo possui volume $V = i^3$ e área superficial $A = 6\ i^2$. Deduz-se, ainda, que os novos subdomínios, originados da divisão do domínio original em N partes, possuirão volumes $V_i = i^3 / N$ e área superficial de $A_N = 6\ i^2 / N^{2/A.}$ Grosso modo, a carga de processamento devido à cada processador é equivalente ao volume. Por sua vez, o fluxo de dados (comunicação) entre os subdomínios é proporcional à área superficial. O gráfico da Fig. A1.5 mostra a taxa de diminuição dos volumes e das áreas dos subdomínios.

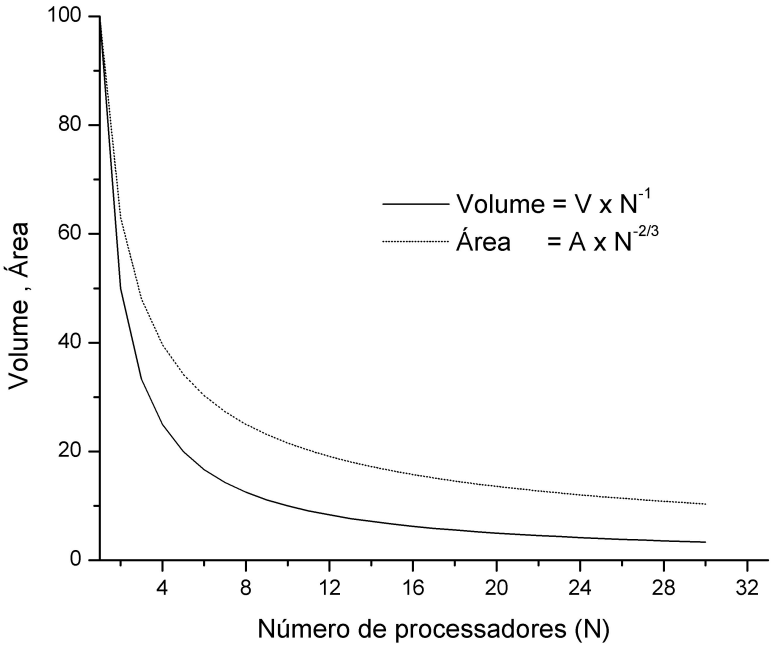


Figura A1.5 – Relação entre a diminuição do volume e da área em função do número de processadores.

É possível perceber que, à medida que mais processadores são adicionados, o volume decresce mais rapidamente do que a área superficial em cada subdomínio. Tal característica sugere que a dificuldade no particionamento pode ser devida mais à comunicação entre os processadores do que aos cálculos *per se*, o que pode se agravar, caso a estratégia de divisão do domínio não seja bem feita. Com efeito, a decisão sobre qual a topologia ideal de paralelização, ou seja, a quantidade e a forma dos subdomínios empregados na decomposição, não é uma tarefa trivial. Com relação à quantidade de volumes em cada subdomínio (que se associa diretamente ao balanço de carga) é possível escolher a divisão de acordo com a dimensão do domínio ou com a quantidade de volumes contidos nele. Em domínios discretizados por malhas regulares, não há diferença no balanço de carga, quaisquer que sejam as opções. Entretanto, para malhas irregulares, uma divisão baseada nas dimensões do domínio original resulta em subdomínios dotados de quantidades diferentes de volumes. Algumas vezes, este efeito é desejável (como em certas configurações de *clusters* heterogêneos), mas, na maioria das situações, este desbalanço é difícil de ser contornado e foi, no presente trabalho, preterido em relação à divisão por volumes.

As categorias de comunicação entre sub-domínios podem ser divididas em comunicação local / global, estruturada / não-estruturada, estática / dinâmica e sincronizada / não-sincronizada. Na comunicação *local*, cada processo se comunica apenas com seus processos "vizinhos", enquanto que na comunicação *global* exige-se uma comunicação com vários

processos ao mesmo tempo. Na forma *estruturada*, os dados a serem trocados formam um conjunto regular, como uma grade ou árvore por exemplo, e no modelo *não-estruturado*, os dados podem formar padrões completamente arbitrários.

Entende-se como comunicação estática quando as identidades dos dados a serem trocados não variam ao longo do tempo. Por outro lado, quando as estruturas se alteram constantemente ao longo da simulação, temos uma comunicação *dinâmica*. Em uma comunicação *sincronizada*, as operações de troca entre os processadores são executadas de uma maneira coordenada entre as partes envolvidas, onde cada uma das partes não realiza sua operação sem a contrapartida da outra. No modo *não-sincronizado*, um processador pode receber (ou enviar) informações sem que haja a cooperação do outro.

Não é uma tarefa fácil determinar a performance de um programa paralelo, devido às diversas configurações de *hardware* e *software* possíveis. Entretanto, de forma a padronizar as relações de desempenho entre os diferentes algoritmos paralelizados, algumas definições se fazem necessárias, como, por exemplo, o *speedup*, a escalabilidade e o desempenho.

O **speedup** pode ser entendido como a razão entre o tempo de processamento para um único processador e o tempo para **N** processadores. Entretanto, cabe salientar que todo programa paralelo possui uma parte serial, impossível de ser paralelizada. Esta parcela serial do programa e o *speedup* estão intimamente ligados pela *lei de Amhdal*. Segundo esta lei, se um programa possui uma parte serial (e todo o programa a possui) que representa *1/*S do tempo total de execução, diz-se que o *speedup* máximo que pode ser atingido por este mesmo programa é S. A título de exemplo, se um determinado programa possui 5% de seu código serial (o que já é um tanto difícil de se obter) podemos dizer, a partir da lei de *Amhdal*, que o *speedup* máximo a ser atingido é 20. Entretanto, na maioria dos problemas práticos, esta relação torna-se muito simplista para poder representar todas as variantes do processo, de forma que o *speedup* pode atingir valores bem maiores que S. Uma expressão matemática prática para o *speedup* poder ser:

$$S(N) = \frac{T_S}{T_N} \tag{A1.1}$$

onde T_S é o tempo gasto por um programa rodando em um só processador e T_N é o tempo gasto por este mesmo programa rodando em **N** processadores.

A **eficiência** (E) pode ser definida como sendo o quanto de *speedup* é obtido à medida que novos processadores são adicionados. Uma expressão para o seu cálculo é dada por:

$$E(N) = \frac{S(N)}{N} \tag{A1.2}$$

Costuma-se, ainda, empregar o termo **desempenho**, que nada mais é do que $E(N)x100\%$. Pode-se dizer que o valor ideal de eficiência seja 1 ou, então, um desempenho de *100*%.

Uma curva característica de *speedup* em função do número de processadores pode ser vista na Fig. A1.6, retirada de Marinho *et al.* (2004). Nesta figura, a reta a 45° representa o *speedup* ideal, ou seja, aquele que aumenta na proporção que mais máquinas são adicionadas ao processamento. Na situação apresentada não foi possível observar o comportamento do código para um número maior de processadores. Porém, sabe-se que, à medida que eles são incluídos nos cálculos, a curva apresenta um padrão aproximadamente parabólico. Após o *speedup* atingir o máximo possível, seu valor começa a regredir, de forma que o acréscimo de novos processadores pode prejudicar o desempenho do código. Na Fig. A1.7, os resultados da eficiência do código computacional para o mesmo problema pode ser observada. A reta paralela ao eixo das abscissas e que cruza o eixo de eficiência no valor igual a 1, representa o desempenho de *100*%, considerado ideal.

A capacidade que os programas possuem de aumentar seu *speedup* à medida que mais processadores são adicionados é definida como **escalabilidade**. Esta propriedade é muito significativa a ponto de se tornar peça de propaganda das empresas de softwares comerciais, ao dizerem que seus produtos possuem a escalabilidade de centenas, ou mesmo milhares, de processadores. A escalabilidade perfeita, ou seja a capacidade de manter a eficiência constante, não importando o número de processadores que são adicionados, é o "Santo Graal" do processamento paralelo.

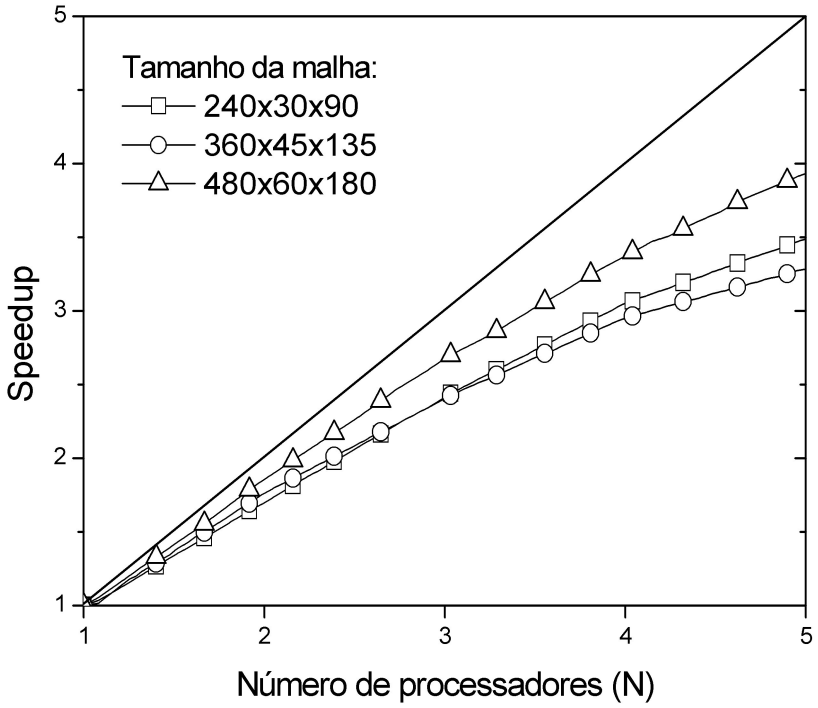


Figura A1.6 – Resultados de *speedup* para três malhas de refinamentos diferentes.

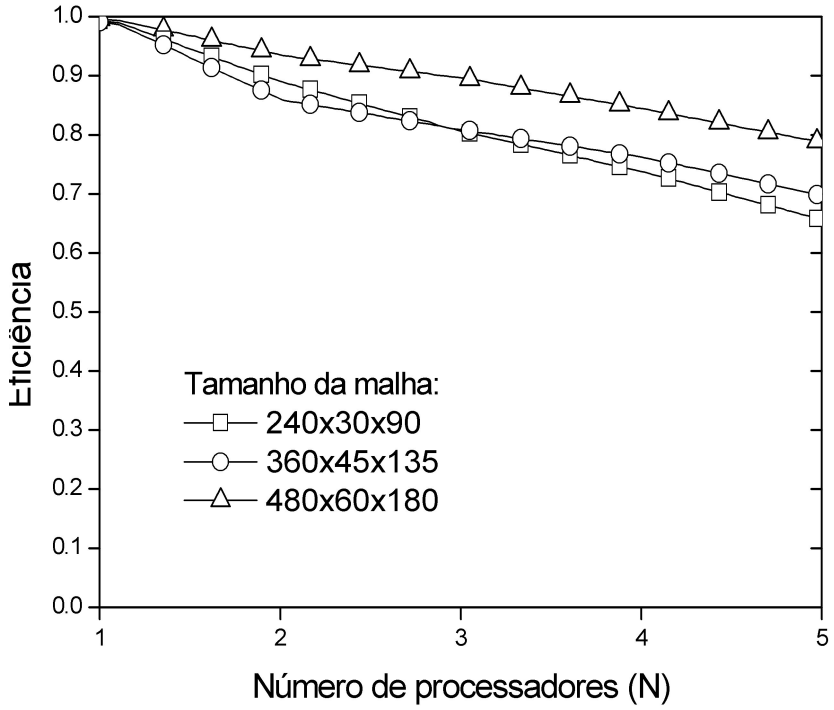


Figura A1.7 – Resultados da eficiência obtidos em três malhas diferentes.

A1.4 Paralelização do presente problema

Para o presente trabalho, o particionamento do domínio computacional foi grandemente auxiliado pelo fato das variáveis, no interior das células elementares, serem posicionadas num arranjo co-localizado. Nesta configuração, como mencionado, o domínio original é subdividido em sub-domínios menores, de forma a se particionar a carga de trabalho do modo mais igualitário possível entre os processadores. Assim, a comunicação entre os subdomínios torna-se necessária para que o problema avance, satisfazendo às condições de contorno exigidas. Cada processador executa basicamente o mesmo código, porém sobre uma base de dados diferente. A topologia apresentada pela interface de comunicação entre os subdomínios pode variar de unidimensional a tridimensional, conforme mostrada na Fig. A1.8. Na topologia unidimensional, a troca de mensagens é realizada em uma única direção, que é estendida a duas direções na topologia bidimensional e três para o caso tridimensional, sempre relacionado à um volume localizado na interface. A título de exemplo, tome-se o estêncil gerado na interface de troca de mensagens entre os subdomínios: dividindo-os empregando-se a topologia unidimensional, implicaria no intercâmbio de informações ao longo de um único eixo coordenado. No caso bidimensional, os dois eixos do estêncil trocariam mensagens e, por fim, no caso tridimensional, os três eixos sofreriam intercâmbio de dados. A relação de diminuição da área superficial referente à escolha da topologia está apresentada na Fig. A1.9, aos moldes do que foi feito na Fig. A1.5. Cabe salientar que o volume computacional no interior de cada subdomínio permanece como sendo $V_i = i^3 / N$ independentemente do tipo de topologia escolhida.

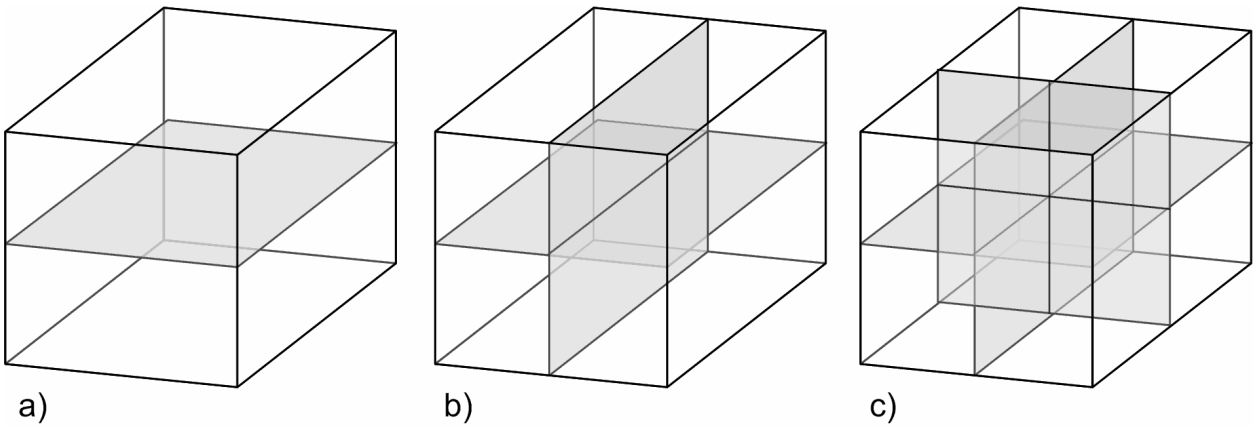


Figura A1.8 – Topologias possíveis para a troca de mensagens: *a)* unidimensional, *b)* bidimensional e *c)* tridimensional.

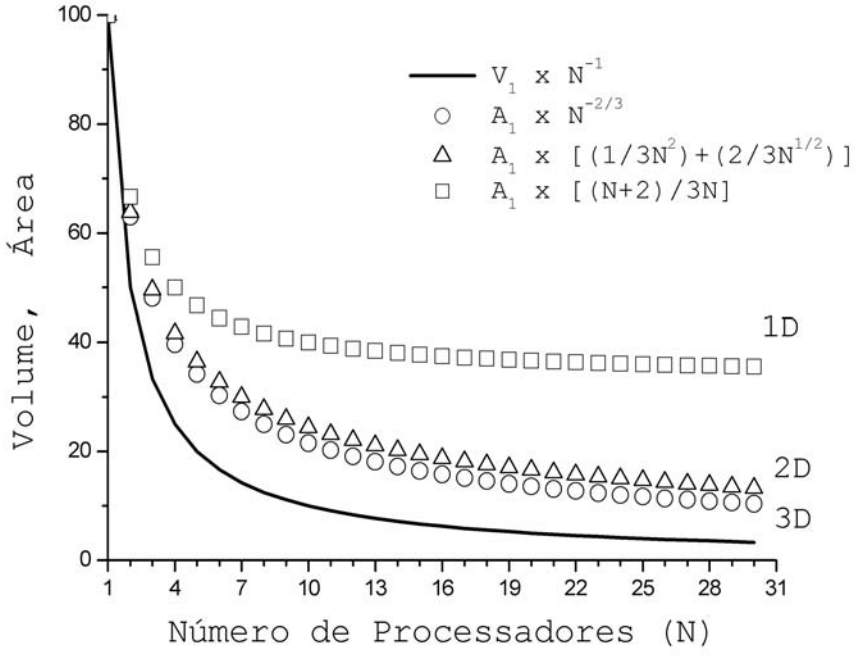


Figura A1.9 – Relação entre a diminuição do volume computacional e da área de troca de mensagens para as três topologias possíveis de particionamento.

É possível observar, com relação à Fig. A1.9, que o particionamento unidimensional atinge a sua capacidade máxima de diminuição da área superficial de troca de mensagens entre os subdomínios adjacentes muito antes das demais topologias. Evidentemente, esta característica, num problema de grande porte, implica num desempenho inferior do cluster. No contexto do presente trabalho, a escolha pela partição unidimensional não comprometeu sua performance, principalmente por se empregarem poucos processadores nos cálculos. A vantagem mais clara de se empregar particionamento unidimensional reside na implementação bem mais simplificada do código.

Foi empregada, ainda, uma discretização dos termos espaciais de segunda ordem, o que exige um estêncil de apenas uma célula vizinha, de cada lado do ponto de interesse, nos três eixos coordenados. Assim, na interface entre os subdomínios, fez-se necessária a sobreposição (*overlapping*) de apenas um plano. Por outro lado, aproximações de ordem mais elevada, exigiriam um número maior de planos e, conseqüentemente, um fluxo maior de troca de mensagens e possível sobrecarga (*overhead*) da rede. Na Fig. A1.10 abaixo, pode-se observar que os volumes internos ao domínio, como os volumes 2 e n - *1* que foram calculados e estão atualizados, são copiados aos volumes n e 1, respectivamente, agindo como condição de contorno aos domínios Ω_1 e Ω_2.

Decomposições de domínios em topologias de dimensões maiores que a unitária são, em geral, mais trabalhosas para o programador, pois demandam um controle mais sofisticado e cuidadoso da seleção e agrupamento dos pacotes a serem enviados, bem como o controle e manipulação dos pacotes recebidos. Por outro lado, existe um evidente acréscimo da área superficial (A) de troca de mensagens, melhorando a sua relação com o volume (V) do subdomínio e aumentando a escalabilidade do algoritmo. Pelo pequeno porte do cluster empregado neste trabalho e pelas características geométricas do domínio computacional, a escolha pela topologia unidimensional tornou-se a mais conveniente. Na Fig. A1.11 é possível observar, conjuntamente, as sobreposições entre os domínios e a topologia unidimensional empregada.

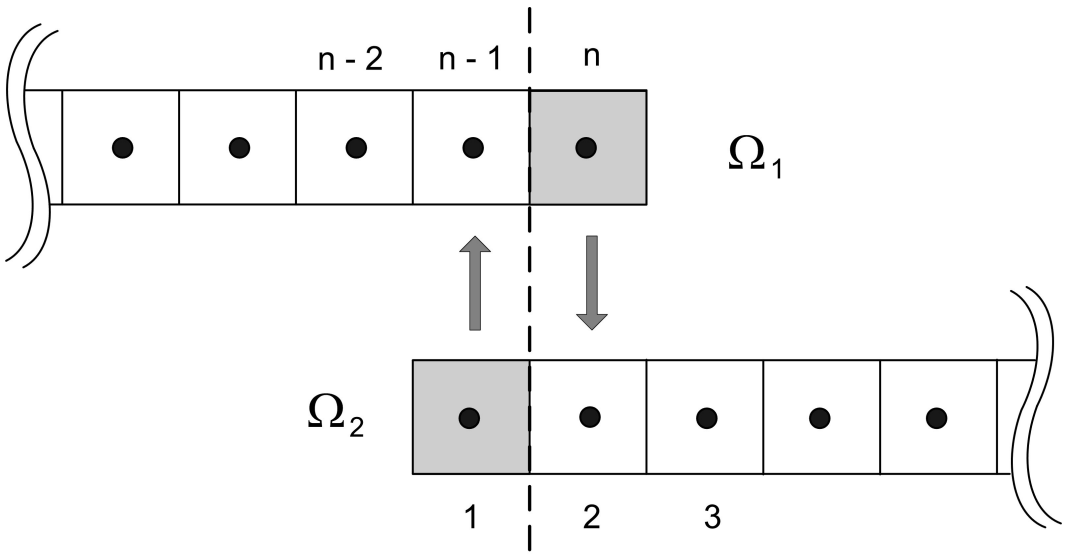


Figura A1.10 – Sobreposição necessária para aproximações espaciais de segunda ordem.

Pelas características de sincronismo, exigidas pela discretização implícita no tempo, a troca de mensagens entre os subdomínios deve ser realizada a cada interação para solução do sistema linear e a cada atualização as condições de contorno. A título de exemplo, o algoritmo na Fig. A1.12 esquematiza os instantes em que a troca é realizada (quadros em cinza) durante um ciclo completo do código desenvolvido no presente trabalho. Nesta figura, as abreviaturas U,V,W e T correspondem, respectivamente, às três componentes do vetor velocidade e à temperatura. Em caso de falha na comunicação ou perda de sincronia devida a erros de implementação, o resultado imediato é uma possível descontinuidade nos resultados numéricos, pois o tempo pode ter transcorrido de forma diferente para os subdomínios.

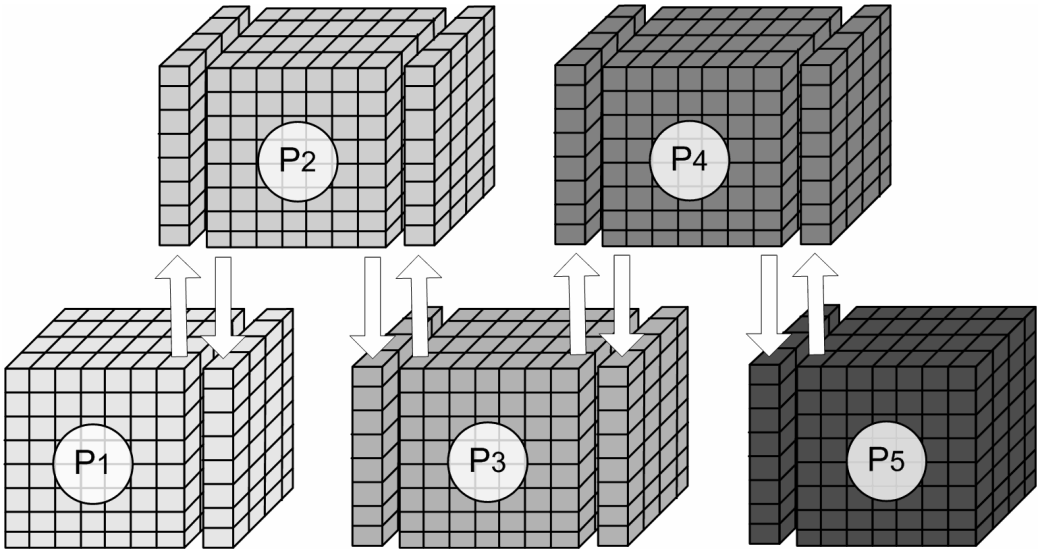


Figura A1.11 – Comunicação entre os subdomínios usando uma topologia unidimensional de divisão.

A1.5 Bibliotecas de paralelização

A comunicação entre os domínios, juntamente com a quantidade de cálculos processados em cada um deles, requer um cuidado especial, pois afeta não somente o desempenho da simulação mas, também, a integridade dos resultados. Infelizmente, ainda não existe um software que seja capaz de paralelizar automaticamente, de forma eficiente e com segurança, todas as situações possíveis encontradas em processamento paralelo. Programas de maior desempenho ainda necessitam da intervenção do programador para se chegar ao resultado desejado.

Dentre os diversos detalhes a serem explorados está a forma como os dados trafegarão entre os subdomínios. Em sistemas de processamento distribuído, esta tarefa fica a cargo de ferramentas que promovem a troca de mensagens. Felizmente, encontra-se disponível e de forma gratuita na *internet* algumas excelentes bibliotecas que podem ajudar nesta tarefa. Podem-se citar, como mais populares, a PVM (*Parallel Virtual Machine*) e a MPI (*Message Passing Interface*) sendo esta a empregada no presente trabalho.

A biblioteca PVM foi desenvolvida em 1989 por um grupo de universidades e centros de pesquisa nos EUA em um projeto chamado *Heterogeneous Network Project*. Como proposta inicial, o objetivo dos pesquisadores era desenvolver uma biblioteca que permitisse a comunicação entre máquinas de natureza heterogênea sob o conceito de troca de mensagens (Pitanga, 2002). De forma geral, o sistema PVM consiste de uma aplicação que, uma vez inicializada, é responsável por comunicar-se com as demais máquinas da rede, formando a

rede virtual e de um conjunto de bibliotecas próprias (em torno de 38) para realizar a criação e destruição de processos, troca de mensagens e sincronização de tarefas.

Assim, pode-se dizer que a principal característica da PVM é poder construir uma rede heterogênea com eficiência, sendo controlada por uma única máquina virtual. Além disso, permite criar e destruir processos em tempo de execução, sendo possível remover ou adicionar máquinas durante a simulação. Finalmente, a PVM possui notável tolerância à falhas, pois caso a máquina virtual não encontre comunicação com um nó da rede, ela automaticamente redireciona as tarefas para um outro nó, sem interromper o processo. Entretanto, toda esta facilidade na construção da rede virtual implica num custo alto em termos de sobrecarga de comunicação (*overhead*), prejudicando o desempenho do código o que tende a piorar, consideravelmente, em sistemas com grande número de processadores. Por este motivo, dificilmente encontram-se *clusters* com várias máquinas utilizando PVM. Com o intuito de ser padrão em ambientes de memória distribuída, foi lançada em novembro de 1992 a primeira versão da biblioteca de paralelização MPI. Seu desenvolvimento foi realizado como um trabalho conjunto de cerca de 80 pesquisadores provenientes de, aproximadamente, 40 organizações industriais, acadêmicas e governamentais, principalmente americanas e européias. A biblioteca fornece um conjunto de rotinas para comunicação entre processos de memória, possui em torno de 125 funções (embora seja possível escrever vários códigos com apenas 6 delas) além de ser portável para diversas plataformas. Suporta, ainda, linguagens C/C++ e Fortran 77/90 nas quais suas bibliotecas são compiladas e *linkadas*.

Para a construção do programa empregado neste trabalho, foi escolhida a implementação MPICH (*MPICHamaleon*) do padrão MPI-2. Esta biblioteca é disponível gratuitamente na *internet* e foi desenvolvida pelo *Argonne National Laboratory* (ANL) e pela *Mississipi State University* (MSU), ambos nos EUA. Sua primeira versão foi lançada em 1993 e, atualmente, encontra-se na versão 1.2.6, datada de agosto de 2004. Além de ser considerada uma das únicas versões que combinam portabilidade, interoperabilidade e alta performance, possui ainda capacidade de lidar com máquinas heterogêneas e SMP (*Symmetric Multi Processors*), ou seja, máquinas individuais com dois ou mais processadores. Outras vantagens importantes são a disponibilidade de diversos recursos, que podem ser explorados visando melhorar o rendimento dos códigos e a sua excelente portabilidade. Finalmente, pode ser dito que o MPI apresenta-se como um padrão para processamento paralelo, o que garante constante manutenção e atualização.

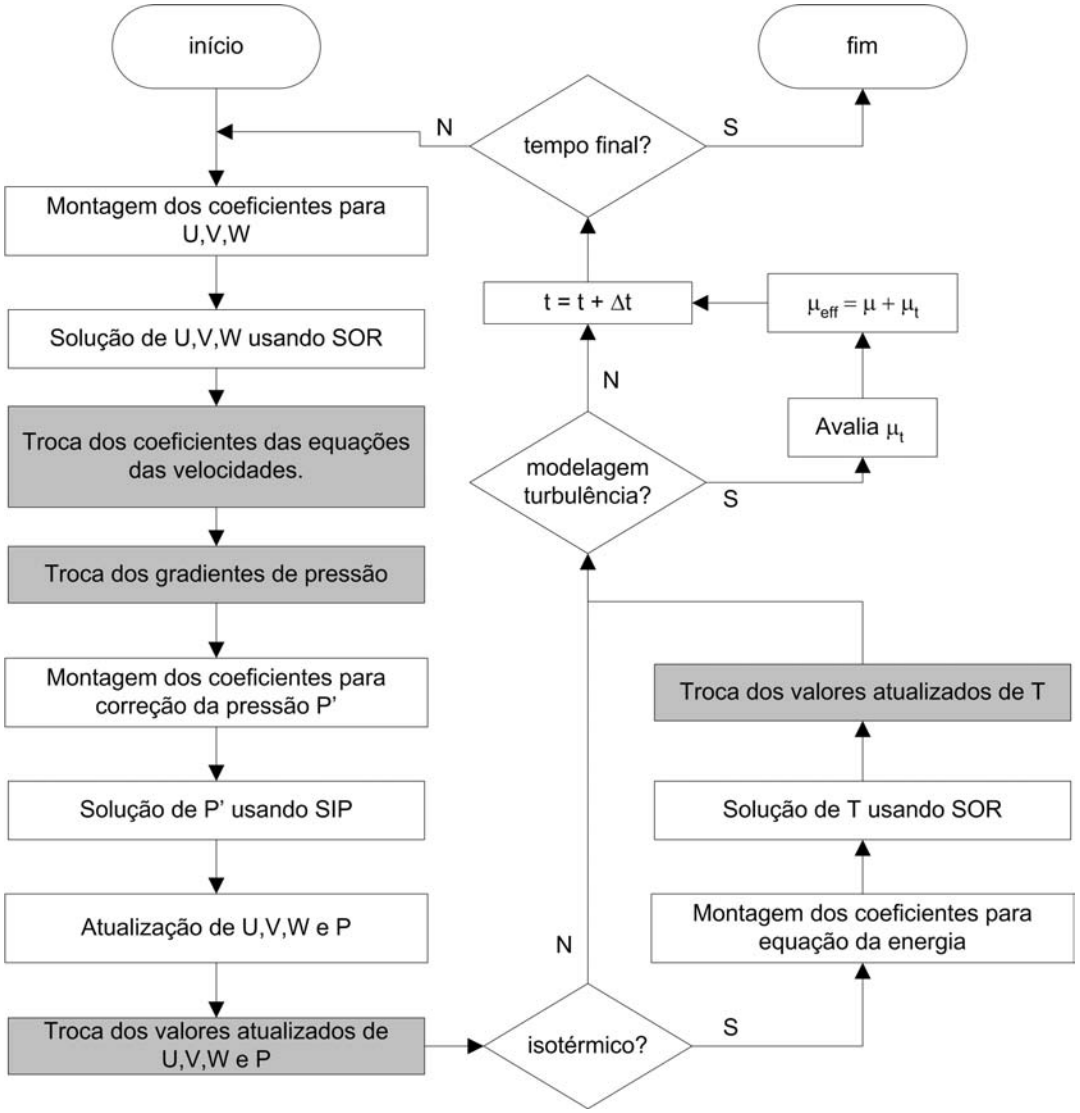


Figura A1.12 – Troca de mensagens durante um ciclo iterativo.

ANEXO II

Equação da Energia

No presente trabalho, a equação da energia foi também implementada para uso em trabalhos futuros, inclusive introduzindo o processo de transferência de calor ao Modelo Físico Virtual. Dessa forma, procurar-se-á apresentar a sua discretização completa a título informativo.

A equação de conservação da energia total do sistema pode ser obtida de maneira similar à empregada nas equações de conservação da massa e quantidade de movimento no Capítulo 3. A Primeira Lei da Termodinâmica relaciona a taxa de variação da energia com o fluxo líquido de calor que atravessa a fronteira do sistema, mais o trabalho líquido realizado sobre o volume de controle, na forma:

$$dE = dQ + dW \,. \tag{A2.1}$$

Ou, como uma variação temporal:

$$\frac{DE}{Dt} = \frac{DQ}{Dt} + \frac{DW}{Dt} \,. \tag{A2.2}$$

O calor transferido pode ser avaliado usando a lei de Fourier:

$$\frac{DQ}{Dt} = -\nabla \cdot q = \nabla \cdot \left(k \nabla T \right) \tag{A2.3}$$

e o trabalho realizado sobre o elemento de fluido (White, 1991):

$$\frac{DW}{Dt} = \nabla \cdot \left(\mathbf{v} \cdot \mathbf{T} \right) . \tag{A2.4}$$

Substituindo-se Φ por E (E = energia total do sistema) na Eq. (3.2), aplicando-se o teorema da divergência sobre as Eqs. (A2.3) e (A2.4) e usando-as juntamente com a Eq. (A2.2), obtém-se a equação da conservação da energia (ou, simplesmente, equação da energia):

$$\frac{\partial}{\partial t}\int_{\Omega}\rho h d\Omega + \int_{S}\rho h \mathbf{v}\cdot\mathbf{n} dS = \int_{S} k\nabla T\cdot\mathbf{n} dS + \int_{\Omega} q_e d\Omega \,, \tag{A2.5}$$

onde $h = c_P T$ é a entalpia. O valor do termo-fonte q_e deve ser considerado a partir das simplificações feitas no modelo. Sabe-se que o termo da energia corresponde à soma da componente de energia cinética do fluido, da energia interna e a da energia potencial gravitacional (Fortuna, 2000). Além disso, podem-se incluir fontes de geração de calor como, por exemplo, as provenientes de combustão. Para a maioria das aplicações, é possível desconsiderar os efeitos provocados por todas as componentes citadas anteriormente. Caso se deseje incluí-las, o termo q_e completo pode apresentar-se da forma (Ferziger e Peric, 2002):

$$\int_{\Omega} q_e d\Omega = \int_{\Omega}\left(\mathbf{v}\cdot\nabla p + \mathbf{S}:\nabla\mathbf{v}\right) d\Omega + \frac{\partial}{\partial t}\int_{\Omega}\rho d\Omega \,, \tag{A2.6}$$

onde $\mathbf{S}$ é a parte viscosa do tensor tensão $\mathbf{T}$, definida como:

$$\mathbf{S} = \left(-\frac{2}{3}\mu\nabla\cdot\mathbf{v}\right)\mathbf{I} + \mu\left[\nabla\mathbf{v} + \left(\nabla\mathbf{v}\right)^{\mathbf{T}}\right]. \tag{A2.7}$$

A forma mais simplificada da Eq. (A2.5) pode ser obtida fazendo $q_e = 0$ e considerando o fluido com calor específico (c_P) constante, resultando na equação de transporte advectivo/difusivo para a temperatura:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho T d\Omega + \int_{S}\rho T\mathbf{v}\cdot\mathbf{n} dS = \int_{S}\frac{k}{c_P}\nabla T\cdot\mathbf{n} dS \,. \tag{A2.8}$$

Seguindo-se os mesmos procedimentos adotados anteriormente, pode-se obter a equação de balanço, na forma integral, para uma propriedade ϕ qualquer, como sendo:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\phi d\Omega+\int_{S}\rho\phi\mathbf{v}\cdot\mathbf{n}dS=\int_{S}\Gamma\nabla\phi\cdot\mathbf{n}dS+\int_{\Omega}q_{\phi}d\Omega \tag{A2.9}$$

onde q_{ϕ} é o termo de geração ou destruição de ϕ e Γ é a difusividade desta propriedade.

Como comentado anteriormente, a Eq. (A2.5) pode ser usada para avaliar o transporte de energia interna do sistema. Integrando-a no tempo e no espaço, segundo as técnicas já apresentadas no Capítulo 3, tem-se:

$$\left(\frac{3\rho T_P^n-4\rho T_P^{n-1}+\rho T_P^{n-2}}{2\Delta t}\right)\Delta x\Delta y\Delta z+\left(\rho_e u_e T_e-\rho_w u_w T_w\right)^n\Delta y\Delta z+\left(\rho_n u_n T_n-\rho_s u_s T_s\right)^n\Delta x\Delta z+$$
$$\left(\rho_t u_t T_t-\rho_b u_b T_b\right)^n\Delta x\Delta y=\left[\left(\frac{k}{c_P}\frac{\partial T}{\partial x}\right)_e-\left(\frac{k}{c_P}\frac{\partial T}{\partial x}\right)_w\right]^n\Delta y\Delta z+ \tag{A2.10}$$
$$\left[\left(\frac{k}{c_P}\frac{\partial T}{\partial y}\right)_n-\left(\frac{k}{c_P}\frac{\partial T}{\partial y}\right)_s\right]^n\Delta x\Delta z+\left[\left(\frac{k}{c_P}\frac{\partial T}{\partial z}\right)_t-\left(\frac{k}{c_P}\frac{\partial T}{\partial z}\right)_b\right]^n\Delta x\Delta y+q_e\Delta x\Delta y\Delta z\,.$$

Aproximando-se a temperatura nas interfaces por diferenças centrais e substituindo na equação acima, tem-se a equação discretizada para a temperatura:

$$A_P T_P=A_E T_E+A_W T_W+A_N T_N+A_S T_S+A_T T_T+A_B T_B+B \tag{A2.11a}$$

onde:

$$A_I=\max\left(-Flux_i,0\right)+Diff_i \qquad I=E,N,T \qquad i=e,n,t \tag{A2.11b}$$

$$A_I=\max\left(Flux_i,0\right)+Diff_i, \qquad I=W,S,B \qquad i=w,s,b \tag{A2.11c}$$

$$Flux_i=\rho_i u_i S_i, \qquad i=e,w,n,s,t,b \tag{A2.11d}$$

$$Diff_i=\frac{k_i S_i}{c_{Pi}\delta_i}, \qquad i=e,w,n,s,t,b \tag{A2.11e}$$

$$A_P=\sum_I A_I+B_P\Delta V\,, \qquad I=E,W,N,S,T,B \tag{A2.11f}$$

$$B = q_e \Delta V \,, \tag{A2.11g}$$

Como na discretização da equação do balanço da quantidade de movimento, todos os termos que compõem os coeficientes A_I foram calculados com base em propriedades conhecidas no instante de tempo anterior (ou seja, em n-1). O termo-fonte B pode abrigar, caso desejado, a função dissipação (Eq. A2.6) que, após devida manipulação (White, 1991), assume a forma :

$$q_e = 2\mu\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\right] + \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)^2$$

$$+\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)^2 + \mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)^2 \tag{A2.12}$$

onde λ é a segunda viscosidade que, para gases, pode assumir o valor $\lambda = -\frac{2\mu}{3}$ (hipótese de Stokes) e para líquidos ou escoamentos incompressíveis, onde $\nabla \cdot \mathbf{v} = 0$, o último termo do lado direito da equação acima se anula.