IMPLEMENTAÇÃO COMPUTACIONAL EM
AMBIENTE PARALELO DE MEMÓRIA DISTRIBUÍDA
PARA ANÁLISE ACOPLADA DE SISTEMAS OFFSHORE

Luciano Donizetti Franco

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

_____
Prof. Breno Pinheiro Jacob, D.Sc.

_____
Prof. Nelson Francisco Favilla Ebecken, D.Sc.

_____
Dr. Marcos Donato Auler da Silva Ferreira, Ph.D.

_____
Prof[a]. Myrian Christina de Aragão Costa, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
ABRIL DE 2004

FRANCO, LUCIANO DONIZETTI

Implementação Computacional em Ambiente Paralelo de Memória Distribuída para Análise Acoplada de Sistemas Offshore [Rio de Janeiro] 2004

XII, 133 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Civil, 2004)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas Offshore

2. Análise Acoplada

3. Computação Paralela

I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

A Deus por tudo.

Aos meus familiares, em especial aos meus avós, pelo carinho e incentivo durante toda a minha vida.

Ao Professor Breno Pinheiro Jacob pela valiosa orientação, apoio e incentivo durante todo o mestrado.

Aos professores do Programa de Engenharia Civil da COPPE, em especial a Gilberto Ellwanger e Sagrilo pelos ensinamentos dentro da área offshore e a Myrian C. A. Costa pelos ensinamentos dentro da área de computação paralela.

Aos meus amigos e colegas de trabalho do LAMCSO (Laboratório de Métodos Computacionais em Sistemas Offshore) do Programa de Engenharia Civil COPPE/UFRJ.

Aos amigos Eduardo Macedo, Francisco da Silva, Cris Anderson, Petrônio e Antônio pelo apoio, solidariedade e amizade demonstrada nestes dois anos de convívio.

Aos amigos (as) do curso de mestrado pela amizade.

Aos eternos e inestimáveis amigos (as) da FEG/UNESP pelo incentivo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

IMPLEMENTAÇÃO COMPUTACIONAL EM AMBIENTE PARALELO DE MEMÓRIA DISTRIBUÍDA PARA ANÁLISE ACOPLADA DE SISTEMAS OFFSHORE

Luciano Donizetti Franco

Abril/2004

Orientador: Breno Pinheiro Jacob

Programa: Engenharia Civil

Este trabalho tem como objetivo a implementação em ambiente de processamento paralelo de memória distribuída de um programa orientado para a análise de unidades flutuantes ancoradas, considerando o acoplamento do casco com as linhas de ancoragem e risers, permitindo obter simultaneamente os movimentos da unidade flutuante e a resposta estrutural das linhas. Acima de tudo busca-se apresentar uma solução eficiente para minimizar o tempo de processamento gasto neste tipo de análise, através da paralelização do código seqüencial empregando bibliotecas de comunicação MPI. A implementação paralela foi realizada para operar em ambientes paralelos com memória distribuída como *Cluster de PC's*. Os resultados obtidos comprovam a eficiência da técnica de paralelização adotada.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

COMPUTER IMPLEMENTATION IN DISTRIBUTED MEMORY PARALLEL ENVIRONMENT FOR COUPLED ANALYSIS OF OFFSHORE SYSTEMS

Luciano Donizetti Franco

April/2004

Advisor: Breno Pinheiro Jacob

Department: Civil Engineering

This work presents an implementation in parallel environment of distributed memory of a program used for coupled analysis of anchored flotation units (offshore system) to obtain the movements of the flotation unit and the structural response of the lines simultaneously. The main objective is to show an efficient solution to minimize the processing time consumed in this analysis type, through the parallelization of the sequential code using MPI. The parallel implementation was made to operate in parallel environment of distributed memory as PC's Cluster. The results achieved show efficiency of the parallelization method adopted.

ÍNDICE

1 INTRODUÇÃO

1.1 OBJETIVOS

O presente trabalho tem como objetivo efetuar a implementação em um ambiente de processamento paralelo de memória distribuída de um programa orientado à análise acoplada de unidades flutuantes ancoradas, empregadas na exploração de petróleo (*sistemas offshore*). Este programa considera modelos hidrodinâmicos para o casco e modelos de elementos finitos para as linhas de ancoragem e risers, permitindo realizar análises que consideram o acoplamento entre os comportamentos do *sistema offshore* e assim obter simultaneamente os movimentos da unidade flutuante e a resposta estrutural das linhas.

As análises acopladas contribuem para que seja possível prever com mais precisão o comportamento de estruturas *offshore*, analisando-se possíveis problemas e soluções antes mesmo de serem instaladas ou de ir a campo avaliá-las. No entanto, este tipo de análise exige uma carga computacional muito grande.

Diante de tal problema busca-se neste trabalho apresentar uma solução eficiente para minimizar o tempo de processamento das análises, através da paralelização do código empregando bibliotecas de comunicação MPI. Como ponto de partida para estas atividades, foi tomado o código seqüencial do programa Prosim [17]. A principal motivação da paralelização é contribuir para o desenvolvimento de análises acopladas de unidades flutuantes.

1.2 HISTÓRICO E MOTIVAÇÃO

O inicio da exploração de petróleo ocorreu em 1889 no Texas (EUA), lugar onde é conhecido como “o berço do Petróleo no mundo”. Já a industrial *offshore* teve inicio no final do século passado na costa da Califórnia. As primeiras construções para exploração de petróleo no mar eram constituídas por estruturas rústicas de madeira e tinham lâminas d’água em torno de cinco metros. Só em 1947 foram construídas as duas primeiras plataformas metálicas pela empresa Superior Oil Company. Estas estruturas ficaram conhecidas como Jaquetas e mudaram o rumo da construção offshore, pois possibilitaram a instalação mais rápida, uma vez que, eram construídas em canteiros e instaladas por guindastes.

No Brasil, inicialmente, a extração de petróleo no mar também era realizada por Jaquetas, pois as lâminas d’águas eram rasas com profundidades variando de 100 m a 500 m. Com a descoberta de campos petrolíferos em água cada vez mais profundas, houve a necessidade de buscar soluções que possibilitassem a exploração destes campos. Com isto surgiram diferentes concepções de plataformas. Pode-se dividir estas concepções segundo sua geração.

A primeira e a segunda geração de plataformas consistiram de plataformas fixas, como, jaquetas e plataformas de gravidade. O sistema de produção flutuante pode ser considerado como sendo a terceira geração. Por plataforma flutuante entende-se por uma estrutura complacente que é mantida em sua posição através de um sistema de ancoragem. As estruturas complacentes caracterizam-se por apresentarem grandes deslocamentos sob a ação das cargas ambientais. Estas plataformas representam uma mudança na filosofia de exploração de petróleo permitindo atingir profundidades nunca antes imaginadas. Mas vale salientar que o desenvolvimento de novos materiais e concepções como risers flexíveis e sistemas de ancoragem incluindo cabos de poliéster contribuíram muito para que se chegasse a tais profundidades.

Com o avanço dessas tecnologias houve também a necessidade de avançar no estado de arte de análise e projeto de unidades flutuantes. Atualmente, a análise de sistemas offshore tende cada vez mais para o desenvolvimento de sistemas

computacionais baseados em métodos numéricos integrados para a análise dinâmica não-linear do sistema estrutural de suporte à plataforma e de suas linhas de ancoragem e risers, considerando automaticamente o *acoplamento* em cada passo do processo de integração no tempo das equações não-lineares de equilíbrio. No entanto, este tipo de análise exige mais recursos computacionais, por possuir um elevado número de graus de liberdade.

Para atender ao aumento na demanda de processamento computacional pode-se contar com o desenvolvimento de *hardware* disponível no mercado, e tirar partido dos recursos de computação de alto desempenho, considerando modernas arquiteturas para processamento vetorial e paralelo. Porem, alguns desses recursos apresentam custos elevados tais como máquinas com arquitetura para processamento vetorial. Uma alternativa que apresenta melhor custo-benefício pode ser encontrada na arquitetura com memória distribuída como por exemplo os clusters de PC's.

Cluster de PC's é um conjunto de máquinas independentes interligadas em rede e configuradas para executar cooperativamente um dado conjunto de serviços computacionais. Para montar um cluster é preciso basicamente ter vários computadores interligados, uma plataforma para manipulação de processos paralelos (LAM, MPICH), um sistema operacional (Linux,Windows), uma linguagem de programação (Fortran, C++) e uma biblioteca de comunicação (MPI, PVM, MPL).

Este recurso computacional se desenvolveu em função do aumento da capacidade computacional dos PC's pessoais, aliado a evolução de técnicas eficientes de conexão. Vale destacar também o papel importante das bibliotecas de comunicação no desenvolvimento destes sistemas. Atualmente, a evolução deste tipo de sistema vem sendo estimulada pelo desempenho alcançado em diversas aplicações, nas mais diversas áreas, e pelo baixo custo total das instalações.

1.3 ORGANIZAÇÃO DA TESE

Inicialmente, o Capítulo 2 apresenta uma descrição sucinta dos componentes básicos de sistemas *offshore*, incluindo os tipos de plataformas, de sistemas de ancoragem e de risers.

No Capítulo 3 apresentam-se os fundamentos da formulação do programa Prosim, incluindo aspectos de modelagem numérica do casco das unidades flutuantes e das linhas de ancoragem e risers.

Já no Capítulo 4 é abordada a questão da programação paralela, foco principal deste trabalho. Neste capítulo é apresentado também o programa simples de comunicação paralelo utilizando MPI.

No Capítulo 5 é ilustrada a análise do código seqüencial realizada para chegar até o trecho paralelizado. Neste Capítulo é ilustrada também a estratégia de implementação adotada. Em seguida no Capítulo 6 são apresentados os resultados de ganho de performance obtidos com a técnica de paralelização adotada, juntamente com comentários e conclusões preliminares.

Por fim, no Capítulo 7 conclui-se este trabalho relatando as conclusões mais importantes do resultado da análise de performance do código paralelizado.

2 SISTEMAS OFFSHORE

2.1 INTRODUÇÃO

Este capítulo traz uma descrição de forma sucinta dos componentes básicos de *sistemas offshore*, tais como sistemas flutuantes, ancoragem e risers. Esta breve descrição serve para que o leitor compreenda melhor a dimensão real do tipo de problema encontrado na tarefa de simular computacionalmente uma unidade flutuante de produção ou perfuração ancorada em alto mar, se familiarizando com denominações e conceitos básicos da indústria *offshore*. Uma descrição mais detalhada pode ser encontrada nas bibliografias referentes ao assunto, por exemplo, a descrição de cada tipo de plataforma pode ser encontrada em Chakrabarti [5].

2.2 SISTEMAS FLUTUANTES

2.2.1 Plataforma Semi-Submersível

As semi-submersíveis (Figura 2.1) são plataformas com estruturas flutuantes largamente empregadas para produção, completação e perfuração. Consistem de dois flutuadores compartimentados em tanques com finalidades de oferecer lastro e flutuação à plataforma. Estes flutuadores são denominados de “pontoons”, os quais apóiam as colunas, também chamadas de pernas, e que por sua vez sustentam os conveses. Sua profundidade pode alterar através do bombeio de água para o tanque de lastro.

As Semi-submersíveis podem ser empregadas tanto em produção quando perfuração. As plataformas Semi-submersíveis de perfuração são geralmente denominadas de MODU (*Moblie Offshore Drilling Unit*).

Figura 2.1 - Vista Geral de uma Plataforma Semi-submersível.

2.2.2 SPAR-BUOY

O sistema Spar consiste de um único cilindro vertical de aço de grande diâmetro, ancorado, operando com um calado de profundidade constante de cerca de 200 metros, o que gera apenas pequenos movimentos verticais e, conseqüentemente, possibilita a adoção de risers rígidos de produção. Neste tipo de plataforma há utilização de supressores de vórtices em torno do cilindro com o objetivo de inibir vibrações induzidas pelo fenômeno de *vortex shedding* decorrente principalmente pelas correntes marinhas.

2.2.3 Navios FPSO's

Os navios FPSO's (*Floating Production Storage and Offloading*) surgiram para atender os desafios de escoamento da produção de petróleo em águas profundas. Consistem em uma unidade estacionária de produção que utiliza um navio ancorado, o qual suporta o seu convés uma planta de processo, armazenamento do óleo produzido e permite o escoamento da produção para outro navio, chamado de aliviador.

Não é desejável que os navios recebam condições ambientais severas de través (perpendicular ao eixo popa-proa), visto que a área de contado exposta às forças de arrasto devidas às ações da onda, vento e corrente é muito grande. Para evitar esta condição é empregado um equipamento chamado de *turret* (Figura 2.2), que é constituído de rolamentos que permitem o navio girar e ficar alinhado com as condições ambientais extremas (*weathervane*). Dependendo do ponto de equilíbrio em função da intensidade das forças de onda, vento e correnteza, o navio receberá mar de proa.

Figura 2.2 - Vista Geral de um Navio FPSO.

2.2.4 TLP

A TLP (*tension leg platform*) consiste numa estrutura similar à semi-submersível, sendo mantida na locação através de tirantes (pernas) que são ancorados no fundo através de estacas e tracionadas no topo pela força resultante entre peso e empuxo (*restauração hidrostática*). Esta tração deve ser mantida ao longo de todo seu comprimento a fim de evitar a desconexão no fundo do mar. Seu casco é semelhante ao casco da plataforma Semi-submersível.

A TLP permite que o uso da completação dos poços seja do tipo 'seca' em semi-submersível, pois restringe os movimentos verticais. Isto facilita o controle e intervenções nos poços. Desta forma torna-se desnecessária a utilização de embarcações com posicionamento dinâmico para a intervenção nos poços, o que ocorre quando é utilizada a completação 'molhada' em que as árvores de natal ficam no fundo do mar.

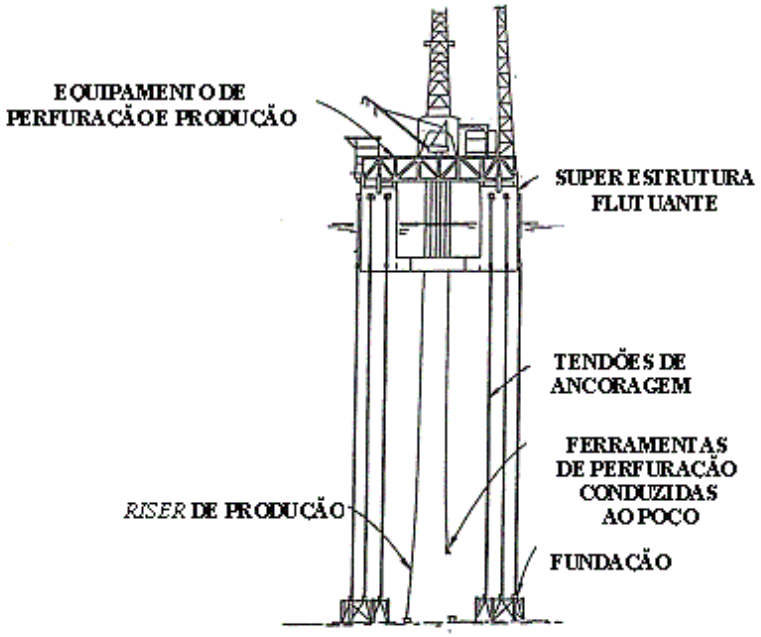


Figura 2.3 - Vista Geral de uma TLP.

2.3 LINHAS DE ANCORAGEM

As linhas de ancoragem têm a função estrutural de fornecer forças de restauração para manter em posição os sistemas flutuantes tais como plataformas semi-submersíveis ou navios. Para oferecer a força de restauração necessária são dispostas em catenária (ancoragem convencional) ou utilizadas como linhas retesadas (*taut-leg*) ou tendões.

2.3.1 Ancoragem Convencional

Denomina-se ancoragem convencional a ancoragem em catenária (Figura 2.4). Esta técnica de ancoragem é utilizada em operações de produção ou perfuração. A ancoragem em catenária mantem a unidade flutuante em uma locação através da força de restauração das linhas. As linhas ancoradas são presas ao fundo do mar por âncoras de resistência horizontal. A força de restauração está relacionada com vários parâmetros, um deles é o raio de ancoragem, um dos principais problemas deste tipo de ancoragem.

Para atender os critérios de projeto para passeio das unidades flutuantes ancoradas (10% da lâmina d'água) tem-se a necessidade de ter um raio de ancoragem razoavelmente grande. Conseqüentemente em um campo de exploração de petróleo isto gera um congestionamento de linhas de unidades próximas, interferindo diretamente no posicionamento das mesmas, juntamente com equipamentos submarinos.

Semi-submersíveis e Navios FPSO's podem ser ancorados com sistema convencional. Os navios FPSO's com ancoragem convencional utilizam o sistema SPM (*Single Point Mooring*). Este sistema de ancoragem é composto por um ponto simples de ancoragem do tipo *Turret* interno ou externo. O *Turret* permite que o navio gire livremente ao redor das linhas de ancoragem e risers e fique orientado na direção das cargas ambientais, reduzindo a atuação destas na estruturas.

2.3.2 Taut-Leg

A ancoragem Taut-Leg (Figura 2.4) é constituída por linhas retesadas com um ângulo de topo de aproximadamente 45° com a vertical. Conseqüentemente, tem-se uma projeção horizontal menor do que a ancoragem convencional, com relação a mesma ordem de grandeza da lâmina d'água.

As linhas da ancoragem Taut-Leg são constituídas nas suas extremidades por cabos de aço ou amarras e no seu trecho intermediário por cabo de poliéster. Esta configuração de linha pode ser a mesma adotada para ancoragem convencional.

As linhas da ancoragem Taut-Leg são fixas na suas extremidades inferiores por meio de estacas de sucção, VLA (âncoras com resistência vertical) ou estacas de fundeio. A ancoragem Taut-Leg é geralmente empregada em plataformas Semi-submersíveis e navios FPSO's.

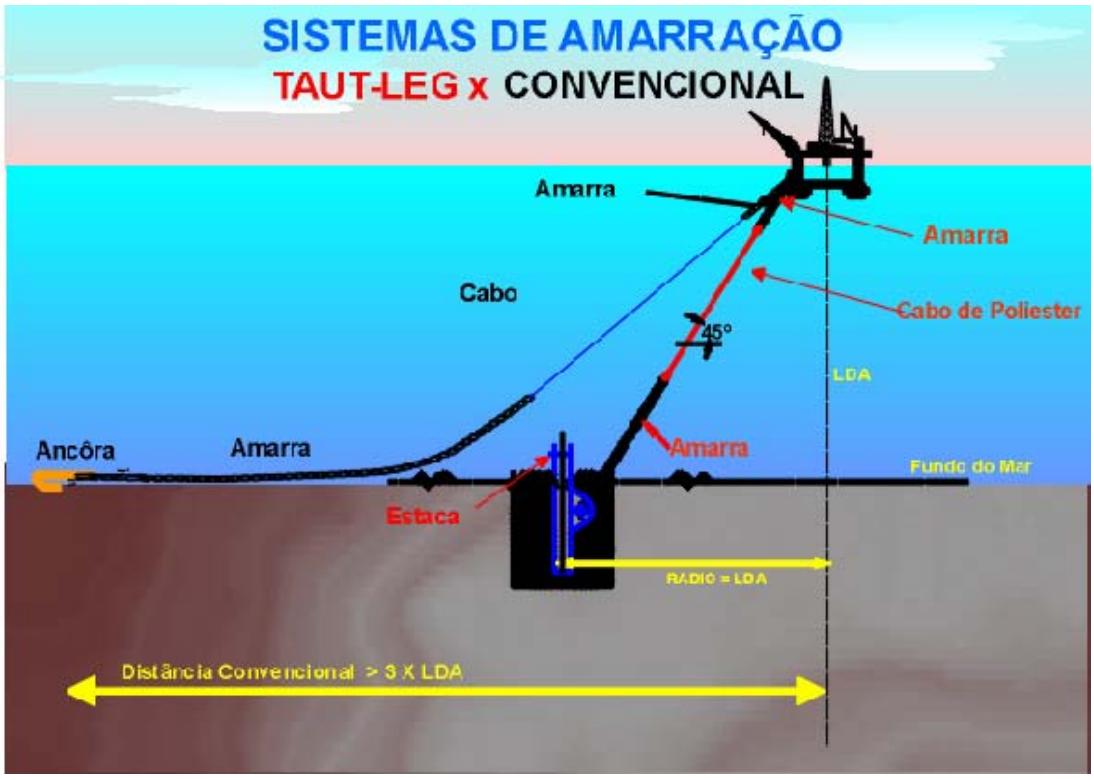


Figura 2.4 - Sistema de Ancoragem Taut-Leg x Convencional.

2.3.3 Tendões

Os tendões podem ser de cabo de aço ou material sintético, proporcionando alta rigidez no plano vertical e baixa rigidez no plano horizontal. A força de restauração no plano horizontal é fornecida pela componente horizontal da força de tração nos tendões. Para tendões de pequenos diâmetros (d ≅ 0.25 m), os efeitos de flexão podem ser desprezados enquanto que para grandes diâmetros (d ≅ 1.00 m) os efeitos de flexão devem ser considerados.

Este tipo de ancoragem baseia-se na utilização de tendões verticais, que precisam estar sempre tracionados devido ao excesso de empuxo proveniente da parte submersa da embarcação. Este tipo de ancoragem é usado principalmente em plataformas tipo TLP (*Tension Leg Plataform*), mas também pode ser adotado por bóias, monobóias, entre outras.

2.3.4 Dicas

Atualmente a PETROBRAS vem empregando o sistema DICAS [7] de ancoragem, o qual consiste basicamente por um sistema de amarração disperso com diferentes resistências na proa e na popa do navio. A diferença básica entre o sistema DICAS e um SPM é com relação ao alinhamento. No sistema SPM o navio se alinha completamente com a direção das cargas ambientais enquanto que no sistema DICAS isto ocorre parcialmente. O sistema DICAS por dispensar o "turret" é um sistema mais simples sob o ponto de vista de construção.

2.4 RISERS

Os risers são linhas que transportam os fluídos, resultante do processo de produção (óleo, gás, lama e água), da cabeça do poço de petróleo no fundo do mar até a

plataforma de exploração na superfície. Existem dois tipos de risers: riser flexível e riser rígido.

2.4.1 Riser Flexível

Risers flexíveis são mangotes especiais empregados em atividades de produção em plataformas baseadas em sistemas flutuantes. São compostos por camadas entrelaçadas (Figura 2.5). A principal vantagem do riser flexível é a característica de ser acentuadamente "complacente", pois podem acompanhar sem problemas todos os movimentos do sistema flutuante.

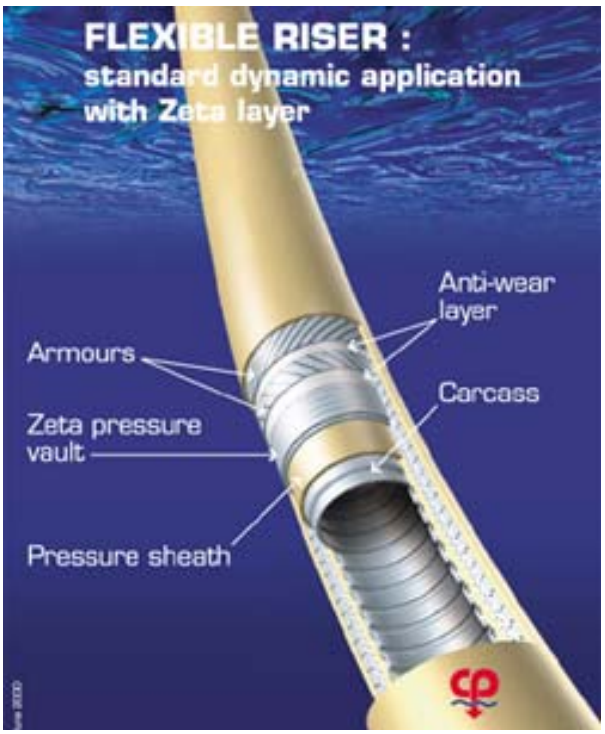


Figura 2.5 - Riser flexível.

Os risers flexíveis podem assumir diferentes configurações em catenária como "Steep Wave" e "Lazy Wave" (Figura 2.6). Estas configurações possuem seções intermediária com flutuadores, cujo empuxo alivia o peso suportado pelo sistema flutuante e quando sob solicitação laterais contribui com momentos restauradores.

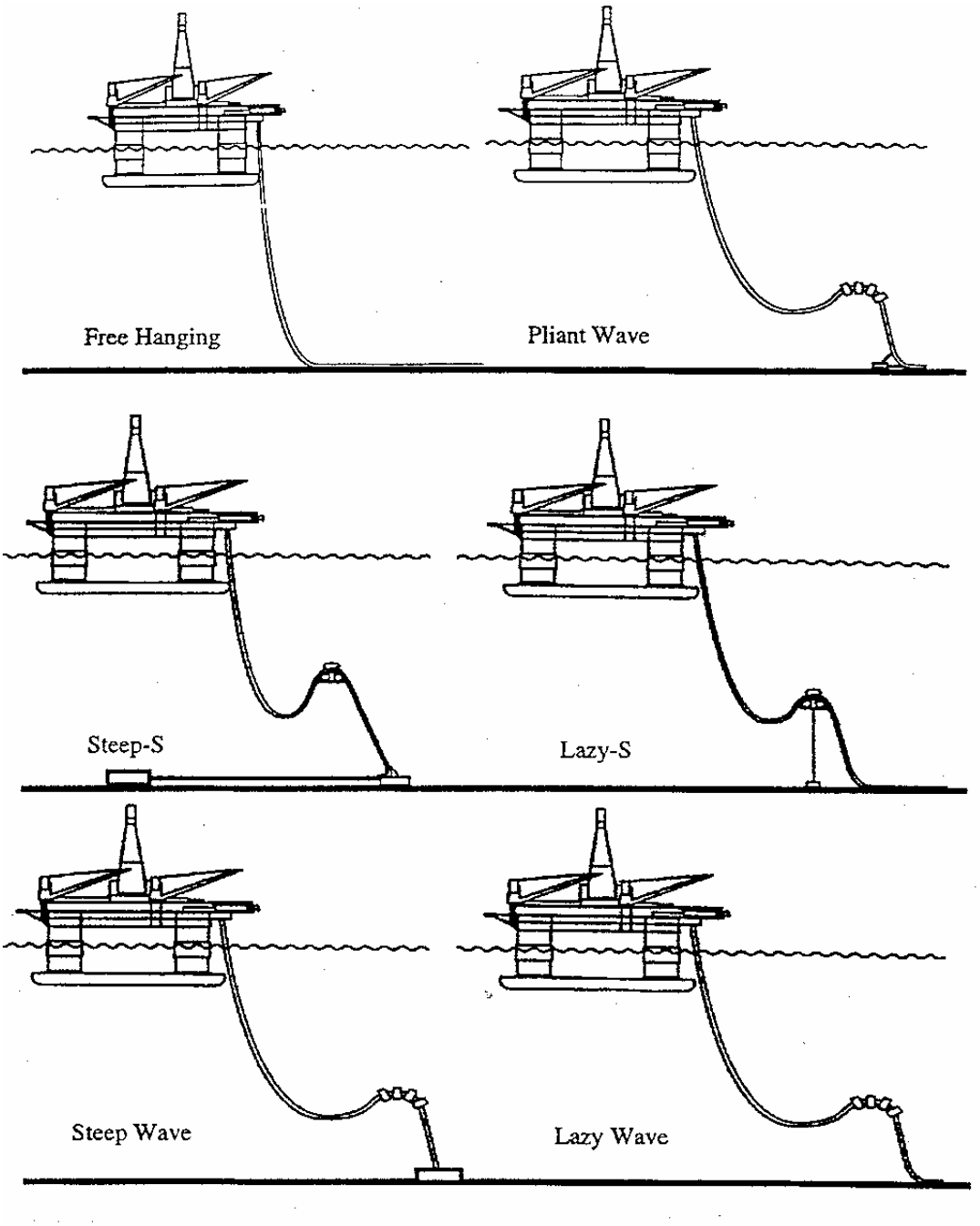


Figura 2.6 – Configuração dos Risers Flexível.

2.4.2 Riser Rígido

Os risers rígidos são compostos por uma série de tubos de aço (Figura 2.7) acoplados uns aos outros. Dispostos em uma configuração vertical podem ser utilizados em atividades de perfuração e também de produção em plataformas que apresentam deslocamentos relativamente menores como TLP. Risers rígidos também podem ser dispostos em configurações em catenária, compondo a concepção conhecida como SCR (Steel Catenary's Riser).

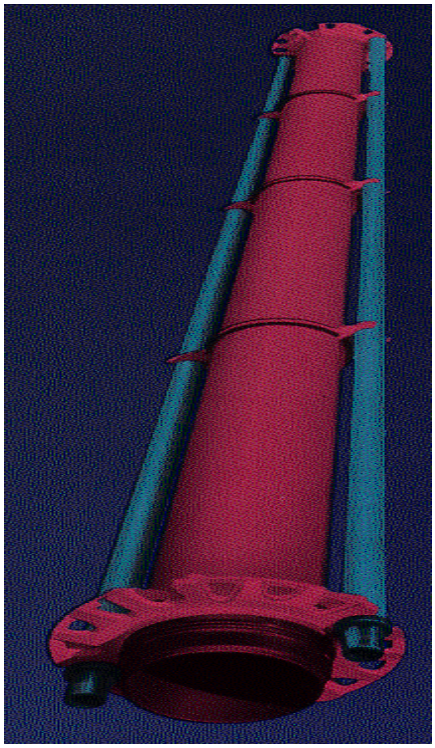


Figura 2.7 - Riser Rígido.

3 FORMULAÇÃO MATEMÁTICA

3.1 INTRODUÇÃO

Neste capítulo apresentar-se-á de forma sucinta a formulação matemática considerada no programa Prosim para a modelagem numérica das unidades flutuantes, bem como as formulações utilizadas na modelagem das linhas de ancoragem e risers. Serão revisados apenas os conceitos básicos envolvidos na análise dinâmica não-linear no domínio do tempo desses componentes de sistemas *offshore*. Maiores detalhes sobre a formulação do programa Prosim podem ser encontrados em [24]. Abordagens mais detalhadas sobre aspectos da formulação também podem ser encontradas na literatura de análise dinâmica e elementos finitos [2, 6 e 4] ou *offshore* [5].

O programa Prosim foi desenvolvido de forma cooperativa por pesquisadores do LAMCSO (Laboratório de Métodos Computacionais e Sistemas Offshore) do Programa de Engenharia Civil da COPPE/UFRJ e do CENPES/Petrobras e desde 1999 vem sendo utilizado em diversos projetos na Petrobras. É orientado para a análise de unidades flutuantes ancoradas, considerando o acoplamento do casco com as linhas de ancoragem e risers, permitindo obter simultaneamente os movimentos da unidade flutuante e a resposta estrutural das linhas.

Sua formulação se baseia no conceito da *análise acoplada*, incorporando um modelo hidrodinâmico para representar o casco da unidade flutuante e um modelo de elementos finitos para representar as linhas. A cada instante do processo de integração no tempo das equações de movimento do casco efetua-se uma ou mais análises não-lineares dinâmicas de cada linha sob a ação da onda, correnteza, peso próprio e das componentes de movimentos do casco.

Modelo Hidrodinâmico do Casco

O modelo hidrodinâmico considera o casco da unidade flutuante representado através de elementos cilíndricos. A formulação das equações de movimento do casco considera diversos efeitos não-lineares, incluindo os devidos aos movimentos de grande amplitude, os incorporados no cálculo das cargas ambientais (por exemplo os termos quadráticos de arrasto da fórmula de Morison), e os efeitos não-lineares associados às linhas de ancoragem e risers. Para integração no tempo das equações de movimento são empregados os métodos de Newmark ou de Runge-Kutta.

As cargas devido à atuação da onda e correnteza são determinadas por um modelo híbrido, proposto originalmente em [22], que combina a formulação de Morison com parcelas de carga de segunda ordem calculadas a partir de resultados da aplicação da Teoria Potencial.

A formulação de Morison original é adequada para membros que podem ser representados por elementos unifilares com diâmetro pequenos em relação ao comprimento das ondas, de modo que as ondas incidentes não sejam perturbadas.

Já o modelo híbrido empregado no programa Prosim permite representar a difração e radiação das ondas incidentes em membros com dimensões relativamente maiores. Com esta formulação as forcas de deriva lenta, bem como o amortecimento dependente da freqüência das ondas é incorporado através da leitura e processamento de coeficientes gerados por um programa baseado no modelo de difração/radiação da Teoria Potencial, tal como o WAMIT.

O uso desta formulação híbrida faz com que o programa Prosim seja adequado para a análise de unidades flutuante compostas por membros cilíndricos de pequenos ou grandes diâmetros, tais como TLP's, plataformas semi-submersíveis, SPAR-buoys e monobóias.

Modelo Estrutural das Linhas de Ancoragem e Risers

O comportamento dinâmico das linhas de ancoragem e risers são descritos matematicamente por um problema de valor inicial e de contorno (PVI/C), constituído por um sistema de equações diferenciais parciais (EDP) hiperbólicas denominadas *equações de movimento* ou *equações de equilíbrio dinâmico*. O modelo matemático contínuo é discretizado no espaço pelo Método dos Elementos Finitos, e no tempo pelo algoritmo de integração implícito de Newmark (Regra Trapezoidal). Efeitos não-lineares são tratados pelo método de Newton-Raphson.

3.2 MODELO DO CASCO

3.2.1 Equação de Movimento

A seguir apresenta-se a formulação das equações de movimento de grande amplitude. Inicialmente, vale a pena rever algumas terminologias empregadas para definir os movimentos de uma unidade flutuante, usuais na Engenharia Naval, mas pouco familiares para engenheiros civis:

- *Surge* – Movimento na direção do eixo longitudinal da unidade flutuante;
- *Sway* – Movimento na direção do eixo transversal da unidade flutuante;
- *Heave* – Movimento vertical da unidade flutuante;
- *Roll* – Movimento de rotação em torno do eixo longitudinal da unidade flutuante (passando pelo CG);
- *Pitch* – Movimento de rotação em torno do eixo transversal da unidade flutuante (passando pelo CG);
- *Yaw* – Movimento de rotação no plano horizontal.

Para o estudo de movimento da unidade flutuante, serão utilizados dois diferentes sistemas de coordenadas, como mostra a figura abaixo:

- *Sistema Estrutural* – Sistema de eixos (OXYZ) que acompanha os movimentos da unidade flutuante, com origem no centro de gravidade (ou a meia nau, na quilha);
- *Sistema Global* - Sistema de eixos (oxyz) fixo no espaço na posição inicial de (OXYZ) no nível do mar;

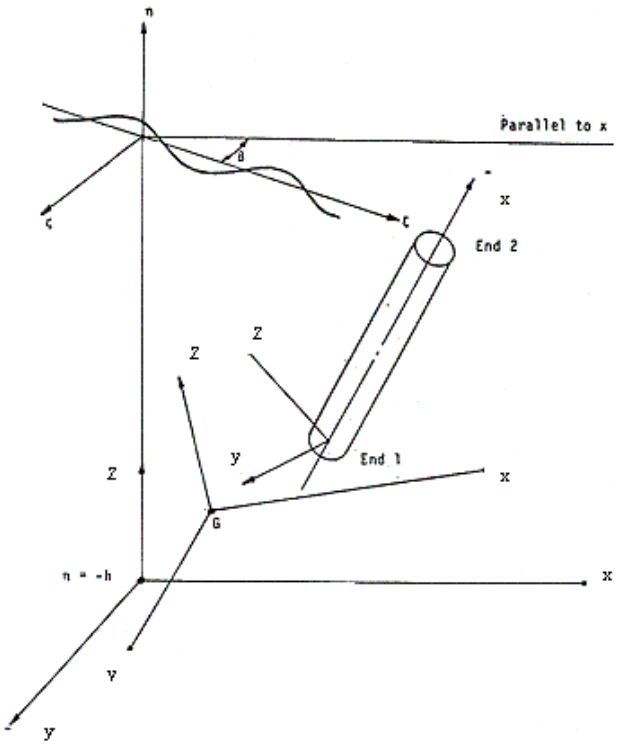


Figura 3.1 - Sistemas de Referência Local.

As equações de movimento de corpo rígido expressam a posição e movimento de OXYZ em relação a oxyz considerando-se as propriedades do corpo e as forças externas causadas por ondas, vento, corrente, sistema ancoragem e risers.

O movimento do corpo flutuante pode ser expresso como sendo o somatório da translação da origem do sistema de coordenadas do corpo, e a rotação em torno de um eixo passando através deste ponto. O movimento translacional é expresso pelas coordenadas de origem do corpo, dependentes do tempo, medidas nas direções xyz e denotadas por $x_1(t)$, $x_2(t)$ e $x_3(t)$. Com o objetivo de se expressar a posição rotacional relativa dos dois sistemas de coordenadas, torna-se necessário a utilização dos ângulos de Euller, γ, α e β, os quais representam respectivamente a rotação em torno do eixo OX (roll), a rotação em torno de OY (pitch) e rotação em torno de OZ (yow).

A transformação geral rotacional de coordenadas, relacionando o sistema de coordenadas globais fixa no espaço, com o sistema de coordenadas imóveis (fixas na plataforma) pode ser expresso pela equação:

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{bmatrix} c\beta c\alpha & s\beta c\alpha & -s\alpha \\ -s\beta c\gamma+c\beta s\alpha s\gamma & c\beta c\gamma+s\beta s\alpha s\gamma & c\alpha s\gamma \\ s\beta s\gamma+c\beta s\alpha c\gamma & -c\beta s\gamma+s\beta s\alpha c\gamma & c\alpha c\gamma \end{bmatrix} \begin{Bmatrix} x - x_1 \\ y - x_2 \\ z - x_3 \end{Bmatrix} \qquad (3.1)$$

onde $s\theta$ = sen θ , $c\theta$ = cos θ , $s\beta$ = sen β, $c\beta$ = cos β, $s\alpha$= sen α e $c\alpha$ = cos α.

A equação 3.1 pode ser reescrita na seguinte forma:

$$X = \mathbf{A} (\mathbf{x} - \mathbf{x}_1) \qquad (3.2)$$

onde **X** é o vetor cujas componentes são (X, Y, Z); **x** e $\mathbf{x}_1$ (coordenadas do centro de gravidade) são similarmente definidas em termos das coordenadas globais do ponto em consideração e as translações das coordenadas de origem, e **A** é a matriz de rotação de coordenadas (3x3).

Observa-se que **A** é uma transformação ortogonal, sua inversa é igual à sua transposta, e a transformação inversa será dada por:

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{A}^T\mathbf{X} \qquad (3.3)$$

Considerando que $\boldsymbol{\omega}$ seja o vetor de velocidades angulares do corpo expresso em termos de ω_1, ω_2, ω_3 em torno dos eixos xyz, a relação entre ω e a derivada no tempo dos ângulos de Euler é dada pela equação 3.4:

$$\boldsymbol{\omega} = \mathbf{B}\,\dot{\boldsymbol{\theta}} \tag{3.4}$$

onde $\dot{\theta} = \{\dot{\gamma}, \dot{\alpha}, \dot{\beta}\}$ e B é definido em 3.5:

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & -sen\ \alpha \\ 0 & cos\ \gamma & sen\ \gamma.cos\ \alpha \\ 0 & -sen\ \gamma & cos\ \gamma.cos\ \alpha \end{bmatrix} \tag{3.5}$$

A matriz **B**, em geral, é uma matriz quadrada e não singular, portanto a sua inversa existe e assim a transformação inversa de 3.4 pode ser reescrita por:

$$\dot{\boldsymbol{\theta}} = \mathbf{B}^{-1}\,\boldsymbol{\omega} \tag{3.6}$$

A segunda lei de Newton para movimentos translacionais e rotacionais é dada pela equação 3.7:

$$\begin{aligned} \mathbf{f} &= \frac{d}{dt}(\,\mathbf{M}\,\mathbf{v}\,) \\ \mathbf{m} &= \frac{d}{dt}(\,\mathbf{I}\,\boldsymbol{\omega}) \end{aligned} \tag{3.7}$$

onde **v** e $\boldsymbol{\omega}$ são os vetores de velocidade translacional e rotacional, **f** e **m** são os vetores de força e momento externos, **M** e **I** são matrizes compostas da massa do corpo, e seus momentos e produtos de inércia de acordo com as equações 3.8 e 3.9:

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \tag{3.8}$$

$$\mathbf{I} = \begin{bmatrix} I_{11} & -J_{12} & -J_{13} \\ -J_{21} & I_{22} & -J_{23} \\ -J_{31} & -J_{32} & I_{33} \end{bmatrix} \tag{3.9}$$

onde:

- m é a massa da plataforma;
- I_{ii} o momento de inércia em torno dos eixos i, dado por:

  $I_{ii} = \int (x_j^2 + x_k^2)\ dm \qquad j,k \neq i$

- J_{ij} = ij-ésimo produto de inércia, dado por:

  $J_{ij} = \int x_i x_j\ dm \qquad i \neq j$

  $J_{ij} = J_{ji}$.

O primeiro termo da equação 3.7 representa as derivadas no tempo do momento e pode ser reescrita por:

$$\mathbf{f} = \mathbf{M}\ \frac{\mathrm{d}}{\mathrm{dt}}(\mathbf{v}) \tag{3.10}$$

onde:

$$\mathbf{v} = \frac{d}{dt}(\mathbf{x}) \tag{3.11}$$

e $\mathbf{x} = \{x_1, y_1, z_1\}$ são as coordenadas do centro de gravidade do corpo em relação ao sistema de referência *oxyz*.

É conveniente também que a equação de momento angular seja avaliada em relação ao sistema de referência móvel, que permanece fixo no corpo, visto que a matriz de inércia é constante neste sistema. A derivada no tempo do momento angular é, portanto, avaliada num sistema de coordenadas que está girando, assim o segundo termo da equação 3.7 torna-se:

$$\mathbf{m} = \mathbf{I}\,\frac{d}{dt}(\boldsymbol{\omega}) + \boldsymbol{\omega}(\mathbf{I}\,\boldsymbol{\omega}) \tag{3.12}$$

As equações 3.10, 3.11, 3.12 e 3.6 podem ser reescritas como:

$$\begin{aligned}
\frac{d}{dt}(\mathbf{v}) &= \mathbf{M}^{-1}\,\mathbf{f} \\
\frac{d}{dt}(\mathbf{x}) &= \mathbf{v} \\
\frac{d}{dt}(\boldsymbol{\omega}) &= \mathbf{I}^{-1}\,[\,\mathbf{m} - \boldsymbol{\omega}(\mathbf{I}\,\boldsymbol{\omega})\,] \\
\frac{d}{dt}(\boldsymbol{\theta}) &= \mathbf{B}^{-1}\,\boldsymbol{\omega}
\end{aligned} \tag{3.13}$$

3.2.2 Solução das Equações de Movimento

Nota-se que e necessário uma modificação na equação 3.13 por causa da variação no tempo do termo de massa adicionada. Percebe-se que a força, **f**, e o momento, **m**, podem ser escritos como sendo a soma de uma parcela dependente da aceleração (termo de massa adicionada) e uma independente da aceleração. Rearranjadas as equações 3.13 tem-se:

$$\begin{aligned} \mathbf{M}\ \frac{d}{dt}(\mathbf{v}) &= -\mathbf{A}\frac{d}{dt}(\mathbf{v}) - \mathbf{B}\ \frac{d}{dt}(\boldsymbol{\omega}) + \mathbf{f_1} \\ \mathbf{I}\ \frac{d}{dt}(\boldsymbol{\omega}) &= -\mathbf{C}\frac{d}{dt}(\mathbf{v}) - \mathbf{D}\ \frac{d}{dt}(\boldsymbol{\omega}) + \mathbf{m_1} - \boldsymbol{\omega}(\mathbf{I}\,\boldsymbol{\omega}) \end{aligned} \tag{3.14}$$

onde:

- **A** e **D** são as matrizes da massa adicionada no presente tempo de integração;
- **B** e **C** são os termos cruzados de massa adicionada;
- $\mathbf{f_1}$ e $\mathbf{m_1}$ são as parcelas dos termos de força e momento que dependem da posição, velocidade e tempo, mas são independentes da aceleração.

As equações 3.14 são mais uma vez rearranjadas:

$$\begin{aligned} (\mathbf{M}+\mathbf{A})\frac{d}{dt}(\mathbf{v}) + \mathbf{B}\ \frac{d}{dt}(\boldsymbol{\omega}) &= \mathbf{f_1} \\ \mathbf{C}\ \frac{d}{dt}(\mathbf{v}) + (\mathbf{I}+\mathbf{D})\frac{d}{dt}(\boldsymbol{\omega}) &= \mathbf{m_1} - \boldsymbol{\omega}(\mathbf{I}\,\boldsymbol{\omega}) \end{aligned} \tag{3.15}$$

A equação 3.15 pode ser reescrita por:

$$\begin{bmatrix} \mathbf{M}+\mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{I}+\mathbf{D} \end{bmatrix} \begin{Bmatrix} \frac{d}{dt}(\mathbf{v}) \\ \frac{d}{dt}(\boldsymbol{\omega}) \end{Bmatrix} = \begin{Bmatrix} \mathbf{f_1} \\ \mathbf{m_1} - \boldsymbol{\omega}(\mathbf{I}\,\boldsymbol{\omega}) \end{Bmatrix} \tag{3.16}$$

Da equação 3.16 pode-se definir a matriz de massa global, dada por:

$$\overline{\mathbf{A}} = \begin{bmatrix} \mathbf{M} + \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{I} + \mathbf{D} \end{bmatrix} \tag{3.17}$$

A matriz de massa global, $\overline{\mathbf{A}}$, é uma matriz simétrica e, em geral, não singular, assim a sua inversa pode ser expressa, numa forma particionada, dada por:

$$\overline{\mathbf{A}}^{-1} = \begin{bmatrix} \overline{A}_{11} & \overline{A}_{12} \\ \overline{A}_{21} & \overline{A}_{22} \end{bmatrix} \tag{3.18}$$

Multiplicando os dois lados da equação 3.16 por $\overline{\mathbf{A}}^{-1}$, obtém-se:

$$\begin{aligned} \frac{d}{dt}(\mathbf{v}) &= \overline{\mathbf{A}}_{11}\ \mathbf{f}_1 + \overline{\mathbf{A}}_{12}\ [\ \mathbf{m}_1 - \boldsymbol{\omega}(\mathbf{I}\,\boldsymbol{\omega})\ ] \\ \frac{d}{dt}(\boldsymbol{\omega}) &= \overline{\mathbf{A}}_{21}\ \mathbf{f}_1 + \overline{\mathbf{A}}_{22}\ [\ \mathbf{m}_1 - \boldsymbol{\omega}\ (\mathbf{I}\,\boldsymbol{\omega})\ ] \end{aligned} \tag{3.19}$$

Existem alguns métodos numéricos que podem ser utilizados para a integração no tempo de um sistema de equações diferenciais acopladas da forma, $\frac{dy}{dt} = f(y,t)$, que como pode ser visto é similar às equações 3.19. No Prosim emprega-se o método de Runge-Kutta de quarta ordem para a integração das equações de movimento, e recentemente foi implementado também o método de Newmark.

As equações 3.19 encontram-se numa forma apropriada para a aplicação do algoritmo de Runge-Kutta, uma vez que o lado direito destas equações não possui mais termos com derivadas. O método de Runge-Kutta emprega uma função de extrapolação do tipo polinomial, expressando os coeficientes do polinômio em termos dos valores estimados das derivadas em pontos intermediários no interior e no contorno do intervalo de integração. Por exemplo, considerando *N* equações diferenciais ordinárias de 1ª ordem, tem-se:

$$\frac{dx_1}{dt} = f_i = (x_1, x_2, \ldots, x_N, t),\ i = 1, 2, \ldots, N \tag{3.20}$$

As soluções das equações 3.20 podem ser extrapoladas em um intervalo de tempo $t_1 \le t \le t_1 + h$ por uma aproximação polinomial de quarta ordem:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 \tag{3.21}$$

As derivadas desta aproximação polinomial nos pontos t = 0, t = h e no ponto médio, t = h/2, do intervalo de extrapolação são:

$$\begin{aligned} \dot{x}(0) &= a_1 \\ \dot{x}\left(\frac{h}{2}\right) &= a_1 + a_2 h + \frac{3}{4} a_3 h^2 + \frac{1}{2} a_4 h^3 \\ \dot{x}(h) &= a_1 + 2\, a_2 h + 3\, a_3 h^2 + 4\, a_4 h^3 \end{aligned} \tag{3.22}$$

O incremento em x de intervalo h é dado pela substituição em 3.21:

$$x(h) - x(0) = h\,(a_1 + a_2 h + a_3 h^2 + a_4 h^3) \tag{3.23}$$

Isto pode ser mostrado pela substituição:

$$\frac{1}{6}\left[\, \dot{x}(0) + 4\, \dot{x}\left(\frac{h}{2}\right) + \dot{x}(h) \,\right] = a_1 + a_2 h + a_3 h^2 + a_4 h^3 \tag{3.24}$$

Substituindo 3.23 em 3.24, a extrapolação apresenta-se:

$$x(h) = x(0) + \frac{h}{6}\left[\, \dot{x}(0) + 4\, \dot{x}\left(\frac{h}{2}\right) + \dot{x}(h) \,\right] \tag{3.25}$$

Ou, em termos mais gerais:

$$x(t+h) = x(t) + \frac{h}{6}\ [\ \dot{x}\ (t) + 4\ \dot{x}\left(t+\frac{h}{2}\right) + \dot{x}\ (t+h)\ ] \tag{3.26}$$

Os termos que aparecem entre colchetes são derivadas de x(t) em 3 pontos do intervalo de extrapolação. Observando-se que a função é desconhecida neste intervalo torna-se necessário estimar-se valores para os termos entre colchetes. Uma estimativa é obtida pela modificação de 3.26:

$$x(t+h) = x(t) + \frac{h}{6}\ [\ \dot{x}_1 + 2\dot{x}_2 + 2\dot{x}_3 + \dot{x}_4] \tag{3.27}$$

Nesta expressão, as derivadas são calculadas no início do tempo presente utilizando-se valores de variáveis do intervalo de tempo anterior. Assim, duas estimativas são obtidas para o ponto médio do intervalo de tempo e uma estimativa de derivada é obtida no final do intervalo de tempo. As expressões para estas estimativas são apresentadas a seguir:

$$\begin{aligned}
&x_1 = x(t) \quad \text{(no início do intervalo)}\\
&\dot{x}_1 = f\ (x_1,t)\\
&x_2 = x_1 + \frac{h}{2}\ f(x_1,t) = x_1 + \frac{h}{2}\ \dot{x}_1\\
&\dot{x}_2 = f\ (x_2,t + \frac{h}{2});\ \text{(Primeira estimativa de derivada em } \frac{h}{2}\text{)}\\
&x_3 = x_1 + \frac{h}{2}\ \dot{x}_2\\
&\dot{x}_3 = f\ (x_3,t + \frac{h}{2});\ \text{(Segunda estimativa de derivada em } \frac{h}{2}\text{)}\\
&x_4 = x_1 + h\ \dot{x}_3\\
&\dot{x}_4 = f\ (x_4,t + h)
\end{aligned} \tag{3.28}$$

O início de um método de integração deste tipo necessita apenas de valores de variáveis no tempo t = 0. Porém, algumas vezes, efeitos transientes associados às condições iniciais apresentam-se e podem persistir por vários intervalos de tempo. É possível reduzir-se o efeito do transiente inicial fazendo com que o carregamento externo seja aplicado à estrutura suavemente através de uma função rampa durante um determinado período de tempo inicial. No caso específico do PROSIM, esta função é dada por:

$$c(t) = \frac{1}{2}\left(1 - \cos\frac{\pi t}{t_0}\right); \text{ para } t \leq t_0 \qquad (3.29)$$

$$1.0; \text{ para } t > t_0$$

3.3 MODELO DAS LINHAS

3.3.1 Introdução

O comportamento dinâmico das linhas é descrito matematicamente por um problema de valor inicial e de contorno (PVI/C), constituído por um sistema de equações diferenciais parciais (EDP) hiperbólicas denominadas equações de movimento ou equações de equilíbrio dinâmico. Na montagem deste sistema são incorporadas também as equações constitutivas, as quais relacionam tensão x deformação, e as equações deformação x deslocamento. Introduzindo-se também as condições de contorno e condições iniciais ter-se-á um sistema com solução única. Devido à complexidade deste tipo de problema adotam-se métodos numéricos que discretizam as EDP no espaço e no tempo. O processo usual consiste em efetuar as discretizações de forma independente, realizando primeiro a discretização espacial das EDP pelo Método dos Elementos Finitos, transformando as EDP em EDO (Equações Diferenciais Ordinárias), e em seguida a discretização no tempo. Os itens a seguir descrevem resumidamente as etapas deste procedimento de solução.

3.3.2 Discretização Espacial por Elementos Finitos

No programa Prosim esta discretização espacial é feita pelo Método dos Elementos Finitos, empregando elementos de treliça espacial e pórtico espacial. Os elementos de treliça possuem dois nós com três graus de liberdade translacionais por nó, como indicado na Figura 3.2. Como este tipo de elemento não possui graus de liberdade angulares conseqüentemente não é possível fornecer rigidez flexional a estes elementos. Por este motivo estes elementos representam bem linhas para as quais a rigidez à flexão pode ser desconsiderada tais como linhas de ancoragem e cabos umbilicais. Devido à menor quantidade de graus de liberdade por nó, os elementos de treliça usualmente requerem um tempo de processamento inferior ao utilizado pelos elementos de pórtico.

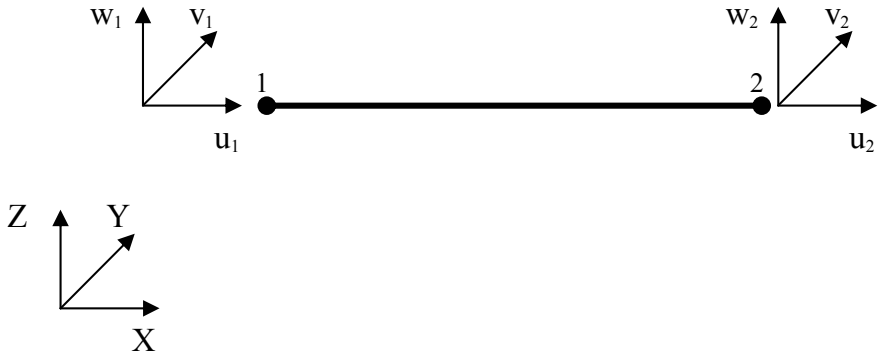


Figura 3.2 – Elemento de Treliça Espacial.

Os elementos de pórtico possuem dois nós com seis graus de liberdade por nó (Figura 3.3). Este elemento permite considerar a rigidez à flexão das linhas e representam bem linhas cuja rigidez à flexão é representativa tais como risers rígidos e risers flexíveis.

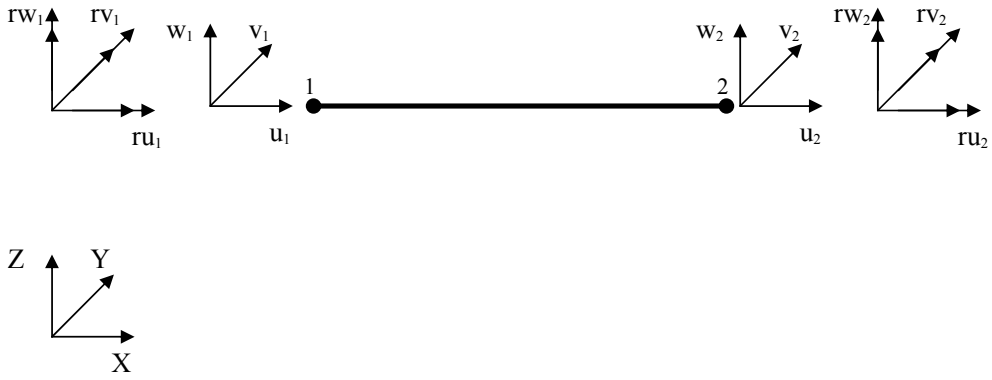


Figura 3.3 – Elemento de Pórtico Espacial.

3.3.3 Equações de Movimento para Problemas Lineares

O segundo passo do procedimento de semidiscretização consiste em efetuar a discretização no tempo das EDO, obtendo-se a resposta dinâmica através de um algoritmo de integração. Vale a pena salientar que a solução numérica exposta neste item refere-se a problemas dinâmicos lineares. Mais à frente será exposto o procedimento para solução de problemas dinâmicos não-lineares.

As equações de movimento para problemas lineares são as seguintes:

$$\mathbf{M}\,\ddot{\mathbf{u}}(t) + \mathbf{C}\,\dot{\mathbf{u}}(t) + \mathbf{K}\,\mathbf{u}(t) = \mathbf{F}(t) \tag{3.30}$$

onde **M**, **C** e **K** representam, respectivamente, as matrizes de massa, amortecimento e rigidez, **F** é o vetor de forças externas e $\ddot{\mathbf{u}}$, $\dot{\mathbf{u}}$ e **u** são respectivamente os vetores de acelerações, velocidades e deslocamentos nodais. Para obtenção da solução considera-se que as condições de contorno estejam incorporadas no processo de discretização espacial e que as condições iniciais são dadas por:

$$\mathbf{u}(0) = \mathbf{u}_0 \;\; ; \;\; \dot{\mathbf{u}}(0) = \mathbf{v}_0 \tag{3.31}$$

onde $\mathbf{u}_0$ e $\mathbf{v}_0$ são, respectivamente, vetores contendo os valores conhecidos para os deslocamentos e velocidades nodais no instante t = 0.

Para solucionar a equação 3.30 emprega-se um algoritmo de integração direta no domínio do tempo. A integração direta se baseia essencialmente em duas idéias:

- A primeira é satisfazer a equação de equilíbrio dinâmico a cada passo de tempo integrado;

- A segunda é assumir que a variação dos deslocamentos, velocidades e acelerações a cada passo de tempo são definidas por operadores que caracterizam o método de integração.

Assim, para a discretização das equações (3.30) no tempo utiliza-se um algoritmo de integração, que fornece aproximações $\mathbf{a}_{n+1}$, $\mathbf{v}_{n+1}$ e $\mathbf{d}_{n+1}$ para os valores exatos $\ddot{\mathbf{u}}(t_{n+1})$, $\dot{\mathbf{u}}(t_{n+1})$, $\mathbf{u}(t_{n+1})$ a partir de aproximações obtidas em instantes anteriores. Substituindo estas aproximações na equação (3.30), obtemos a forma discretizada no tempo das equações de movimento, dada por:

$$\mathbf{M}\,\mathbf{a}_{n+1} + \mathbf{C}\,\mathbf{v}_{n+1} + \mathbf{K}\,\mathbf{d}_{n+1} = \mathbf{F}_{n+1} \tag{3.32}$$

3.3.4 Solução das Equações de Movimento

Para o problema em questão considerar-se-á os algoritmos de Newmark [14], cujos operadores são dados por:

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t\,\mathbf{v}_n + \frac{\Delta t^2}{2}\,[(1-2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}] \tag{3.33}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t[(1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1}] \tag{3.34}$$

A família de Newmark engloba uma série de algoritmos, cada um definido pelos parâmetros ajustáveis γ e β, tais como Diferença Central (método explícito com $\gamma = 0.5$ e $\beta = 0.0$) e Regra Trapezoidal (método implícito com $\gamma = 0.25$ e $\beta = 0.5$). O programa Prosim emprega a Regra Trapezoidal na integração de movimento das linhas e também na integração das equações de movimento do casco.

A implementação mais usual para a solução de problemas de dinâmica estrutural consiste em reescrever os operadores de Newmark em termos de acelerações e velocidades, e substituí-los na equação do movimento. Com isso, chegamos à seguinte expressão, em termos de deslocamentos, para o sistema efetivo:

$$\left[\frac{1}{\beta\,\Delta t^2}\mathbf{M} + \frac{\gamma}{\beta\,\Delta t}\mathbf{C} + \mathbf{K}\right]\mathbf{d}_{n+1} =$$

$$= \mathbf{F}_{n+1} + \mathbf{M}\left[\frac{1}{\beta\,\Delta t^2}\mathbf{d}_n + \frac{1}{\beta\,\Delta t}\mathbf{v}_n + \left(\frac{1}{2\beta} - 1\right)\mathbf{a}_n\right] +$$

$$+ \mathbf{C}\left[\frac{\gamma}{\beta\,\Delta t}\mathbf{d}_n - \left(1 - \frac{\gamma}{\beta}\right)\mathbf{v}_n - \left(1 - \frac{\gamma}{2\beta}\right)\Delta t\,\mathbf{a}_n\right] \qquad (3.35)$$

ou

$$\hat{\mathbf{A}}\,\mathbf{d}_{n+1} = \hat{\mathbf{b}} \qquad (3.36)$$

onde $\hat{\mathbf{A}}$ é a "matriz efetiva", e $\hat{\mathbf{b}}$ é o "vetor de cargas efetivo".

A grande vantagem de empregar este método de integração é a utilização de um intervalo de tempo Δt para a integração, relativamente elevado, maior do que o valor exigido pelos métodos explícitos, onde a estabilidade é garantida somente quando se usa um intervalo de tempo superior ao $\Delta\, t_{cr}$ definido por:

$$\Delta t_{cr} \cong \frac{T_n(\min)}{\pi} \qquad (3.37)$$

onde T_n (min) é o menor período natural do sistema analisado.

Apesar do método implícito ser incondicionalmente estável, deve-se ter o cuidado de se escolher um intervalo de integração de forma a não comprometer a precisão das respostas. Em geral, recomenda-se que o intervalo de integração dos métodos implícitos esteja na ordem de um décimo do menor período natural da estrutura excitada pelo carregamento, a fim de se garantir a precisão dos resultados.

3.3.5 Solução da Equação de Movimento Não-Linear

Generalizando para problemas não-lineares as equações de movimento semidiscretas podem ser expressas da forma:

$$\mathbf{M}\,\ddot{\mathbf{u}}(t) + \mathbf{R}(u) = \mathbf{F}(u,t) \tag{3.38}$$

onde **M** a matriz de massa, **R**(u) é o resíduo e **F**(u,t) a resultante da força externa aplicada ao sistema.

Vale a pena salientar que as não-linearidades estão embutidas nas parcelas R(u) e F(u,t). A primeira parcela inclui efeitos geométricos e/ou de materiais com comportamento elástico não-linear. A segunda parcela considera não-linearidade devido à variação das cargas externas com a geometria, caracterizando carregamentos não-conservativos.

Em problemas de engenharia offshore a não-linearidade se apresenta principalmente nos grandes deslocamentos das unidades flutuantes ancoradas, além das parcelas quadráticas de velocidade relativa fluído-estrutura devido aos carregamentos ambientais, e também no contato variável das linhas com o solo.

Para demonstrar a solução do sistema de EDO não-linear reescreve-se a EDO de forma discretizada, dada por:

$$\mathbf{M}\,\mathbf{a}_{n+1} + \mathbf{R}(\mathbf{d}_{n+1}) = \mathbf{F}_{n+1}(\mathbf{d}_{n+1}) \tag{3.39}$$

O problema pode ser assumido localmente linear, no entorno de uma configuração deformada. Esta linearização consiste em tomar a seguinte aproximação para as parcelas não-lineares, através de uma série de Taylor com termos de ordem superior truncados:

$$\mathbf{R}(\mathbf{d}_{n+1}) = \mathbf{R}(\mathbf{d}_n) + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{d}}\right|_{\mathbf{d}_n} \Delta\mathbf{d} \tag{3.40}$$

$$\mathbf{F}_{n+1}(\mathbf{d}_{n+1}) = \mathbf{F}_{n+1}(\mathbf{d}_n) + \left.\frac{\partial \mathbf{F}_{n+1}}{\partial \mathbf{d}}\right|_{\mathbf{d}_n} \Delta\mathbf{d} \tag{3.41}$$

onde $\Delta\mathbf{d} = \mathbf{d}_{n+1} - \mathbf{d}_n$.

A última parcela de 3.41 define a variação das cargas externas com a geometria. Esta parcela usualmente só é levada em conta quando se exige um tratamento muito rigoroso de carregamento não-conservativo, já que compõe uma matriz não-simétrica e não será considerada nos desenvolvimentos posteriores.

A matriz de rigidez tangente é definida como:

$$\mathbf{K}_T = \frac{\partial \mathbf{R}}{\partial \mathbf{d}} \tag{3.42}$$

Substituindo as equações 340, 3.41 e 3.42 em 3.39, tem-se:

$$\mathbf{M}\,\mathbf{a}_{n+1} + \mathbf{K}_T\,\Delta\mathbf{d} = \mathbf{F}_{n+1}(\mathbf{d}_n) - \mathbf{R}(\mathbf{d}_n) \tag{3.43}$$

onde:

- $\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta\mathbf{d}$;
- $\mathbf{R}(\mathbf{d}_n)$ são os esforços elásticos resistentes calculados com os deslocamentos do intervalo anterior.

Observa-se que as equações 3.43 não garantem mais o equilíbrio dinâmico ao fim do intervalo de tempo t_{n+1}, devido às linearizações assumidas no problema. Para que

o equilíbrio seja alcançado empregam-se técnicas iterativas semelhantes às utilizadas para problemas estáticos, baseados na formulação incremental-iterativa.

FORMULAÇÃO INCREMENTAL-ITERATIVA

Geralmente, uma análise dinâmica não linear não necessita de estratégias iterativas mais caras ou elaboradas. Em geral emprega-se o método de Newton-Raphson Modificado (NRM) no qual a matriz tangente é mantida constante ao longo do ciclo iterativo, e é reavaliada apenas ao início de instantes de tempo escolhidos arbitrariamente. Isto é possível porque os intervalos de tempo que são requeridos para a integração com precisão adequada dos modos de vibração que dominam a resposta dinâmica geralmente representam incrementos de carga menores que os utilizados em casos estáticos, e a matriz efetiva é melhor condicionada que a matriz de rigidez original. Conseqüentemente, a solução iterativa de um problema dinâmico não-linear tende a convergir mais rapidamente do que um problema estático.

A formulação do Método de Newton-Raphson baseia-se em adotar a linearização (equação 3.40) e iterar com matrizes tangentes como a dada por (equação 3.42). No método Newton-Raphson padrão NRP a matriz tangente é reavaliada em todas as iterações. No entanto, os custos de montagem e decomposição associados não compensam os ganhos com a convergência do processo.

Já no método de Newton-Raphson modificado NRM a matriz de rigidez tangente $\mathbf{K}_T$ é calculada ao início de cada intervalo de tempo e mantida constante ao longo do ciclo iterativo, podendo ainda ser mantida constante ao longo de um certo número de intervalos de tempo. Portanto, por oferecer um custo computacional menor o NRM [16] é usualmente mais utilizado.

Para a implementação do algoritmo de Newmark com a formulação NRM devem ser colocadas de uma forma incremental iterativa (I-I) as equações de movimento não-lineares discretizadas no tempo. Considerando-se que $\Delta\Delta\mathbf{d}_{n+1}^{(k)}$

representa a variação dos deslocamentos incrementais obtida a cada iteração do ciclo de verificação do equilíbrio, a expressão I-I escreve-se na forma:

$$\mathbf{M}\,\mathbf{a}_{n+1}^{(k)} + \mathbf{K}_T\,\Delta\Delta\mathbf{d}_{n+1}^{(k)} = \mathbf{F}_{n+1}(\mathbf{d}_n) - \mathbf{R}(\mathbf{d}_{n+1}^{(k-1)}, \Delta\Delta\mathbf{d}_{n+1}^{(k-1)}) \tag{3.44}$$

$$\Delta\mathbf{d}^{(k)} = \Delta\mathbf{d}^{(k-1)} + \Delta\Delta\mathbf{d}_{n+1}^{(k)} \tag{3.45}$$

$$\mathbf{d}_{n+1}^{(k)} = \mathbf{d}_{n+1}^{(k-1)} + \Delta\Delta\mathbf{d}_{n+1}^{(k)} \tag{3.46}$$

Ao longo do ciclo iterativo o vetor de cargas externas $\mathbf{F}_{n+1}(\mathbf{d}_n)$ é mantido constante, mas ao início de cada intervalo de tempo ele é reavaliado.

3.4 CARREGAMENTOS AMBIENTAIS

Os carregamentos ambientais considerados em uma estrutura offshore são onda, correnteza e vento. A onda e correnteza induzem as forças hidrodinâmicas atuantes na estrutura. Para obtê-las, primeiro calculam-se as velocidades e acelerações das partículas fluidas, em seguida essas grandezas são transformadas em forças atuantes sobre os membros da estrutura.

3.4.1 Onda

Um estado de mar geral, irregular, é representado pela superposição linear de ondas senoidais, cada uma com valores associados de freqüência, amplitude e fase. As velocidades, acelerações e pressões em um determinado ponto são obtidas somando-se os valores calculados para cada componente de onda.

De modo geral, emprega-se a teoria linear de Airy [5] para o cálculo das velocidades e acelerações das partículas fluidas devido à onda. Esta teoria é baseada na hipótese de que a elevação da crista da onda é pequena se comparada ao comprimento

da onda ou à profundidade. As condições de contorno são satisfeitas apenas até o nível de águas tranqüilas. Para aplicações em que a altura da onda é significativa, o efeito de alteração da posição da superfície livre sobre a força total induzida pela onda se torna muito importante. Desta forma, adota-se algum tipo de aproximação a partir da teoria original. Entre os tipos de aproximações existentes, destaca-se a extrapolação hiperbólica, linear e os métodos de 'stretching' de Wheeler [25] a qual é o utilizada pelo Prosim.

3.4.2 Correnteza

A correnteza é definida através de um perfil poligonal, em que são fornecidos valores de velocidade e ângulos de incidência.

Normalmente considera-se que, durante a análise dinâmica, a velocidade da correnteza permanece constante. Em condições ambientais de correntezas com valores elevados de velocidade, existe um efeito dinâmico que não é considerado na formulação global, devido às vibrações induzidas por vórtices.

3.4.3 Vento

No cálculo da força de vento sobre a área exposta do casco utiliza-se um modelo simplificado, onde a força de vento atuando no centro de pressão é calculada, em cada intervalo de integração, pela a seguinte equação:

$$F_w = \frac{\rho}{2} U_w^2 A_w \tag{3.47}$$

onde ρ é a densidade do ar, U_w a velocidade total do vento, A_w o produto da área exposta pelo coeficiente de forma (arrasto).

A velocidade do vento, U_w, consiste na soma de uma parcela estática com uma parcela variável no tempo, que é proveniente da simulação a partir de um espectro de vento. O espectro considerado nas análises realizadas pelo Prosim é o proposto pela API [1].

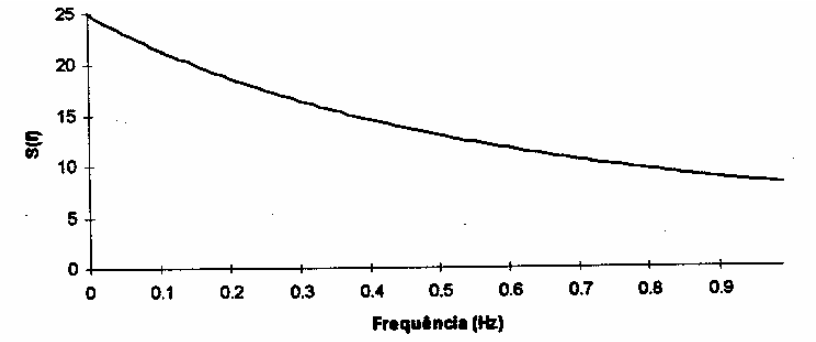


Figura 3.4 - Espectro de Vento da API.

3.4.4 Carga Hidrodinâmica nas Linhas: Formulação de Morison

Para calcular a carga hidrodinâmica induzida pela onda e atuante nos elementos resultantes da discretização das linhas de ancoragem e risers pelo Método dos Elementos Finitos, emprega-se a fórmula de Morison [5]. Para isto, assume-se que o tamanho médio dos elementos sejam grande quando comparado com as dimensões transversais da seção.

A formulação de Morison é bastante difundida para o cálculo das forças fluidas em membros esbeltos, onde os efeitos viscosos preponderam sobre os inerciais. Usualmente, define-se como um corpo esbelto aquele cuja relação D/l < 5, onde D é a dimensão transversal do membro e l o comprimento de onda.

O modelo de Morison pode ser utilizado conforme a seguinte fórmula :

$$f = C_d \frac{1}{2} \frac{\rho}{g} d | v_f - v_e | (v_f - v_e) + C_m \frac{\pi}{4} d^2 \frac{\rho}{g} a_f - C_a \frac{\pi}{4} d^2 \frac{\rho}{g} a_e \qquad (3.48)$$

onde f é força por unidade de comprimento num ponto do membro estrutural, C_d o coeficiente de arrasto, ρ o peso específico da água, g a constante gravitacional, d o diâmetro hidrodinâmico, v_f o vetor de velocidades do fluido na direção normal ao membro estrutural, v_e o vetor de velocidades da estrutura na direção normal ao membro estrutural, C_m o coeficiente de inércia, a_f o vetor de acelerações do fluido na direção normal ao membro estrutural, C_a o coeficiente de massa adicionada e a_e o vetor de acelerações da estrutura na direção normal ao membro estrutural. Os coeficientes de arrasto e inércia são parâmetros obtidos empiricamente, sendo especificados como invariantes ao longo da análise.

As velocidades e acelerações do fluido são calculadas somando-se as contribuições devido à onda (calculadas por exemplo pela teoria linear de Airy [5]) e correnteza. Após a soma vetorial das velocidades de correnteza com as velocidades de onda, realiza-se sua projeção na direção normal ao elemento, e em seguida se aplica a expressão de Morison.

O termo $C_a \frac{\pi}{4} d^2 \frac{\rho}{g} a_e$ corresponde a massa adicionada do membro e pode ser somado a massa estrutural. A massa interna, que também é adicionada a massa da estrutura, correspondente à massa do fluido interno de um riser.

As forças resultantes são calculadas apenas nas extremidades dos elementos, considerando-se uma variação linear ao longo do comprimento do elemento. Para elementos próximos à superfície livre, parcialmente molhados, uma das forças é avaliada no ponto de interseção onda elemento. Depois de avaliada a carga linear ao longo do membro, é calculada as forças nodais equivalentes, que passam a compor o vetor de cargas externas.

Observa-se que o termo de arraste da fórmula de Morison, proporcional ao quadrado da velocidade relativa fluido-estrutura, incorpora efeitos de amortecimento hidrodinâmico. Trata-se de um efeito muito importante no comportamento dinâmico de sistemas offshore em análise de linhas submetidas a grandes velocidades, o valor deste amortecimento costuma ser muito superior ao considerado para o amortecimento estrutural. Em alguns casos o amortecimento do fluido pode chegar a 10% do crítico.

3.4.5 Carga Hidrodinâmica nos Membros Reticulados do Casco: Formulação de Híbrida

Na fórmula de Morison os parâmetros importantes como pressão, velocidade e aceleração são calculadas na linha de centro da seção transversal, e considera-se que a força que o fluido exerce em cada membro não é afetada pela presença de outros membros. Sendo assim, a resultante das forças poderia ser obtida somando-se as forças calculadas para cada membro individual.

Geralmente este não é o caso em plataformas compostas por membros reticulados, com diâmetros relativamente maiores que os de linhas de ancoragem e risers, e um tratamento mais rigoroso deveria considerar que

- A presença do corpo altera de forma significativa o campo de ondas na sua vizinhança, gerando efeitos de difração, interferência e radiação de ondas pelo corpo;

- Existe interação entre os membros, o que leva a efeitos de cancelamento ou sobreposição de ondas, o que depende da freqüência de cada componente de onda.

Para o cálculo das forças exercidas pelo fluido nos elementos da plataforma emprega-se, então, uma formulação híbrida que combina a fórmula de Morison com forças de segunda ordem em função da freqüência da onda, calculados por um programa baseado no modelo de difração/radiação da Teoria Potencial, tal como o WAMIT, mencionado anteriormente, que pode ser utilizado para gerar um arquivo contendo os dados necessários para que o programa Prosim leve em consideração os efeitos do amortecimento por radiação de ondas, e as forças de deriva média e lenta provenientes da reflexão de ondas. Maiores detalhes sobre esta formulação híbrida podem ser encontradas em [24].

4 AMBIENTE DE PROGRAMAÇÃO PARALELO

4.1 INTRODUÇÃO

Este capítulo tem o propósito de apresentar os conceitos básicos relacionados à programação em um ambiente paralelo. De uma forma sucinta, é apresentado o conceito de paralelismo, os tipos de paralelismo, as bibliotecas de comunicação MPI e medidas de desempenho.

4.2 PARALELISMO

Entende-se por paralelismo como sendo uma técnica de dividir tarefas grandes em tarefas menores, as quais serão distribuídas e executadas simultaneamente em vários processadores. Esses processadores se comunicam para que haja coordenação (sincronização) na execução das diversas tarefas executadas em paralelo. A paralelização é feita para aumentar o desempenho no processamento, resolver grandes desafios computacionais e diminuir o tempo gasto no processamento.

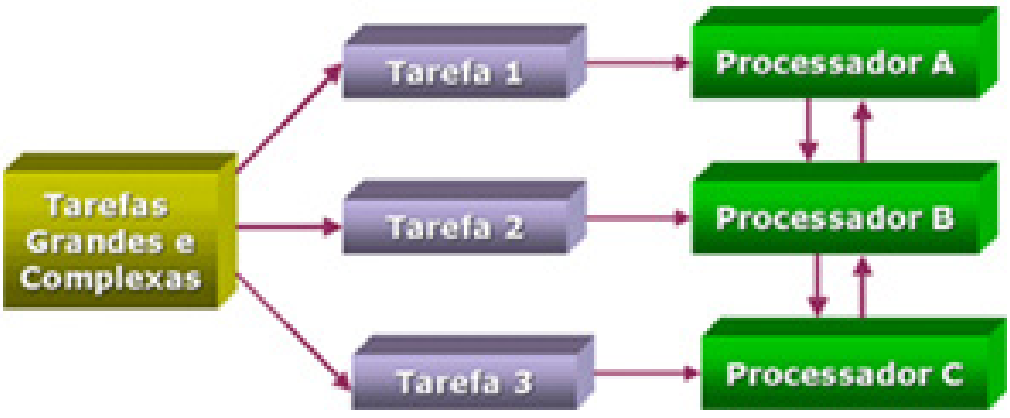


Figura 4.1– Divisão de tarefas.

Dentre as várias formas de classificar o paralelismo, pode-se levar em consideração o objeto paralelizado classificando-o como *paralelismo funcional* e *paralelismo de dados*.

4.2.1 Paralelismo de Dados

No paralelismo de dados o processador executa as mesmas instruções sobre dados diferentes e é aplicado, por exemplo, em programas que utilizam matrizes imensas e para cálculos de elementos finitos. Neste tipo de paralelismo, enquadram-se os sistemas SMP (*Symmetric Multi Processing*) (Figura 4.2), que possuem mais de um processador em um mesmo computador. Todos eles operam independentemente, mas compartilham os recursos de memória e disco, segundo uma política de controle de concorrência adotada pelo sistema operacional. Esta arquitetura é bem transparente ao usuário, ficando a cargo do sistema operacional a maior parte da complexidade. O acesso pelos processadores à memória é feito diretamente sem a necessidade de passagem de mensagem (*Message Passing*). E somente um processador acessa um endereço da memória por vez. Como exemplo de aplicação deste tipo de paralelismo pode-se citar o Cray Y-MP.

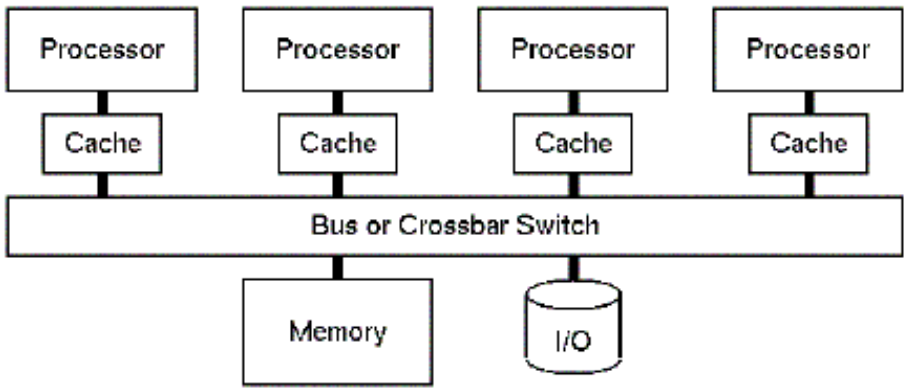


Figura 4.2 – Arquitetura SMP.

4.2.2 Paralelismo Funcional

Já no paralelismo funcional o processador executa diferentes instruções que podem ou não operar sobre o mesmo conjunto de dados. É aplicado em programas dinâmicos e modulares onde cada tarefa será um processo diferente como por exemplo programas em MPI. Neste tipo de paralelismo, enquadram-se os sistemas MPP (*Massive Parallel Processing*) (Figura 4.3), onde há pouco ou nenhum compartilhamento de recursos entre processadores. Normalmente, cada nó de um sistema MPP é um computador independente, com memória e discos próprios. Nestes sistemas, o controle do paralelismo é realizado pelo programador, que deve coordenar as tarefas e a coerência entre os diversos processos. Os processadores estão conectados em rede e o acesso as maquinas é feito por passagem de mensagem (*Message Passing*).

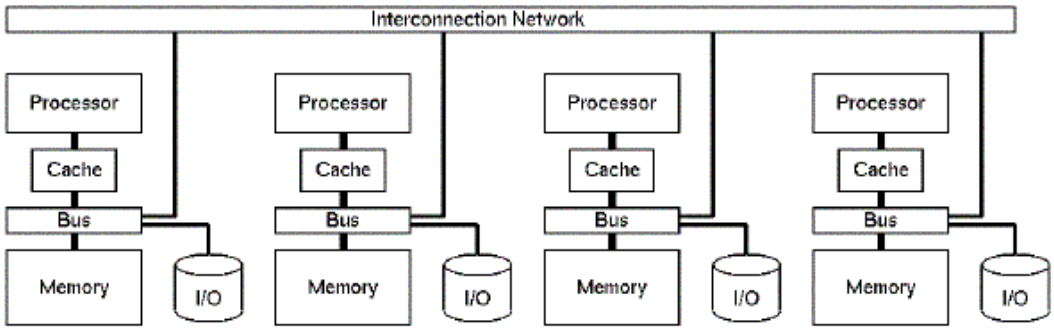


Figura 4.3 – Arquitetura MPP.

O *Message Passing* é um método de comunicação baseada no envio e recebimento de mensagens através de uma rede de computadores seguindo regras de protocolo de comunicação entre vários processadores que possuam memória própria. As informações são enviadas da memória local do processo para memória local do processo remoto. Como exemplos de *Message Passing* pode-se citar:

- PVM - Parallel Virtual Machine;
- MPI - Message Passing Interface;

- MPL - Message Passing Library.

Um exemplo deste tipo de paralelismo pode ser encontrado em uma máquina IBM Risc/6000 SP2 ou em um cluster de PC's. A grande vantagem de uma máquina do tipo IBM Risc/6000 SP2 e cluster de PC's é o acesso à memória local onde não há contenção e, teoricamente, não existe um limite para o número de processadores. No entanto, existe uma dificuldade maior para mapear as informações, e o usuário é responsável pelo sincronismo e recebimento de dados.

4.3 BIBLIOTECA DE COMUNICAÇÃO MPI

O MPI é um dos modelos de *Message Passing* mais empregado atualmente nas diversas áreas da computação em paralelo para ambiente de memória distribuída. Foi introduzido pelo MPI Fórum em maio de 1994 e atualizado em junho de 1995. É um produto resultante de um Fórum aberto constituído por 40 organizações de pesquisadores, empresas, usuários e vendedores que definiram a sintaxe, semântica e o conjunto de rotinas padronizadas para Message Passing. A documentação oficial se chama "*MPI: A Message Passing Standard*", publicada pela University of Tennesee. MPI 2 fornece extensões para MPI e foi finalizado em julho de 1997.

Como característica pode-se citar a eficiência, pois foi projetado para executar eficientemente em máquinas diferentes. É especificado somente o funcionamento lógico das operações. A implementação fica a cargo do próprio desenvolvedor que usa as características de cada máquina para gerar um código mais otimizado.

O MPI ou *Message Passing Interface* consiste em um conjunto de bibliotecas ou funções que auxiliam na comunicação entre processos. Apesar do grande número de funções utiliza-se, na prática, poucas funções. Um programa simples e muitas vezes eficaz pode conter apenas 6 funções, como mostra a Figura 4.4, as quais são responsáveis pela inicialização e terminação do programa, identificação do rank do processo, e envio e recebimento da mensagem. As rotinas MPI empregadas na implementação efetuada neste trabalho são relacionadas no APÊNDICE A.

Figura 4.4 – Programa simples em MPI.

O programa de MPI é um programa único e as tarefas são divididas em forma de desvios, onde cada tarefa executa uma cópia do programa. Este tipo de programa é classificado como *SPMD* (*Single Program Multiple Data*). O *SPMD* é uma taxonomia orientada a programação e é essencialmente igual a taxonomia de Flynn *MIMD* (*Multiple Instruction Multiple Data*) utilizada para classificação de máquinas.

O primeiro comando da Figura 4.4 (MPI_INIT) inicializa o MPI. Em seguida MPI_COMM_SIZE contabiliza o número de tarefas ou processos (*np*), definido na linha de comando de execução (APÊNDICE B). Esses processos são associados a um comunicador e são capazes de comunicar apenas entre os processos pertencentes ao seu comunicador. Inicialmente, todos os processos são membros de um grupo com um comunicador já pré-estabelecido denominado MPI_COMM_WORLD. Os processos têm uma única identificação denominada de rank (0, 1, .., *np*-1, sendo *np* o número de processos), atribuída pelo sistema quando o processo é inicializado e identificado pela chamada do comando MPI_COMM_RANK. Após estas etapas pode-se realizar a troca

de mensagens entre os processos associados ao comunicador. Toda troca de mensagem em MPI possui o formato:

função (endereço, contador, tipo de dado, destino ou origem, etiqueta, comunicador, erro)

onde:

- *função*: pode corresponder a um comando de envio ou recebimento de mensagem;
- *endereço*: localização da memória (*buffer*) onde está armazenada a mensagem a ser enviada ou recebida;
- *contador*: especifica o tamanho da mensagem a ser enviada ou recebida;
- *tipo de dado*: : especifica o tipo de dado a ser enviado ou recebido, normalmente, devem ser iguais nas chamadas de envio e recebimento, exceto, como por exemplo, quando o dado é definido do tipo MPI_PACKED.
- *destino ou origem*: identificação do rank do processo receptor ou emissor;
- *etiqueta*: identificação da mensagem;
- *comunicador*: define um contexto e grupo de comunicação;
- *erro*: código de erro, retorna 0 em caso de sucesso ou código de erro em caso de falha na comunicação.

4.4 MEDIDAS DE DESEMPENHO

4.4.1 Desempenho em Aplicações Paralelas

Os parâmetros mais empregados para avaliar o desempenho em implementações paralelas são:

- Mflop/s: quantidades de operações flutuantes por segundo;

- Speed-up: medida que avalia o ganho de desempenho do algoritmo;

- Eficiência: a medida de desempenho que exprime o comportamento do desempenho da aplicação com a adição de processadores, variando na faixa de $0 < E_p \leq 1$,.

A medida de Mflop/s é calculada a partir da quantidade de instruções de ponto flutuante executadas pelo programa durante toda execução. O speed-up é baseado no tempo total de execução do melhor algoritmo serial e no tempo de execução paralela. A formulação para o cálculo do speed-up é dada por:

$$S_p = \frac{T_s}{T_p} \tag{4.1}$$

onde *Ts* é o tempo de execução do programa serial, *Tp* é o tempo de execução do programa paralelo e *p* é o número de processadores utilizados na execução.

A medida de speed-up pode variar no intervalo $0 < S_p \leq p$. Neste contexto, o speed-up ideal ocorreria quando fosse igual à *p*, o que é pouco provável de acontecer, pois ao ocorrer a comunicação entre os processadores sempre é adicionada uma carga extra ao tempo de processamento.

Por fim, a eficiência a qual pode variar no intervalo $0 < E_p \leq 1$ é dada pela fórmula:

$$E_p = \frac{S_p}{p} \tag{4.2}$$

A escalabilidade de um programa é avaliada através do cálculo de sua eficiência. O programa apresentará boa escalabilidade se a sua eficiência for mantida a medida que se aumenta o número de processadores e o tamanho do problema cresce.

4.4.2 Lei de Amdahl

A lei de Amdahl é uma outra formulação do speed-up levando em consideração o fato de que apenas parte do programa é paralelizado.

Sendo T_s o tempo de execução do programa serial, pode-se supor que o tempo de execução da fração passível de paralelização do programa seja $r.T_s$ e o tempo da fração serial seja (1- r)$.T_s$ (PACHECO, 1997). Desta forma, o tempo de execução deste programa com p processadores será dada por:

$$T_p = \frac{r\,T_s}{p} + (1 - r)\,T_s \tag{4.3}$$

Resultando em:

$$S_p = \frac{1}{(1 - r) + \frac{r}{p}} \tag{4.4}$$

É importante ressaltar que esta formulação está baseada na instância do problema. Portanto, com o aumento da dimensão do problema a parte serial do programa diminui e conseqüentemente o speed-up aumenta.

A eficiência também pode ser obtida pela lei de Amdahl pela fórmula:

$$E_p = \frac{1}{p\,(1 - r) + r} \tag{4.5}$$

Com este conceito de medida conclui-se que a eficiência do programa paralelo aplicado a uma instância de um problema se aproxima de zero à medida que se aumenta o número de processadores.

4.4.3 Overhead

Já que a eficiência é a medida da utilização dos processadores durante a execução paralela de um programa, deve-se considerar também toda a carga extra adicionada ao programa devido à sua paralelização (PACHECO, 1997). Seja a carga de trabalho de um programa serial igual ao seu tempo de execução, isto é, $W_s = T_s$, e a carga de trabalho de um programa paralelo igual à soma dos tempos de execução do programa em cada um dos processadores (T_{proc}), $W_p = T_p = p.T_{proc}$. Assim, pode-se reescrever a formulação de eficiência como:

$$E_p = \frac{T_s}{p\,T_{proc}} = \frac{W_s}{W_p} \tag{4.6}$$

Na prática o overhead é a carga extra de trabalho introduzida devido à paralelização do programa. Ou seja, é diferença entre a quantidade de trabalho realizada por um programa serial e a quantidade de trabalho realizada por um programa paralelo, a qual é dada por:

$$T_0 = p\,T_{proc} - T_s \tag{4.7}$$

O overhead se origina de três fontes:

- Comunicação entre os processadores;
- Tempo ocioso em alguns processadores devido a uma possível diferença de desempenho em alguns dos processadores;
- Cálculos extras necessários na implementação paralela ou cálculos repetidos em vários processadores.

5 IMPLEMENTAÇÃO COMPUTACIONAL PARALELA

5.1 INTRODUÇÃO

O mais importante antes de adotar uma estratégia de implementação paralela é conhecer bem o código seqüencial o qual será paralelizado. Por isto, será apresentada inicialmente neste capítulo a estrutura geral do código seqüencial do programa Prosim, bem como sua predisposição para o paralelismo natural. Em seguida no mesmo capítulo serão descritas as estratégias empregadas para a implementação em paralelo.

5.2 ANÁLISE DO CÓDIGO SEQUENCIAL

Para a implementação do código seqüencial adotado neste trabalho realizou-se primeiramente uma decomposição do algoritmo em módulos. Esta decomposição levou em consideração o aspecto funcional da estrutura do código. De uma forma geral, o código seqüencial do Prosim se divide basicamente em seis módulos funcionais (Figura 5.1):

- leitura de dados de entrada;
- equilíbrio estático individual das linhas (de ancoragem e risers);
- análise estática acoplada (casco, linhas de ancoragem e risers);
- análise dinâmica acoplada (casco, linhas de ancoragem e risers);
- geração de arquivos de saída;
- geração de arquivo para posterior reinício da análise.

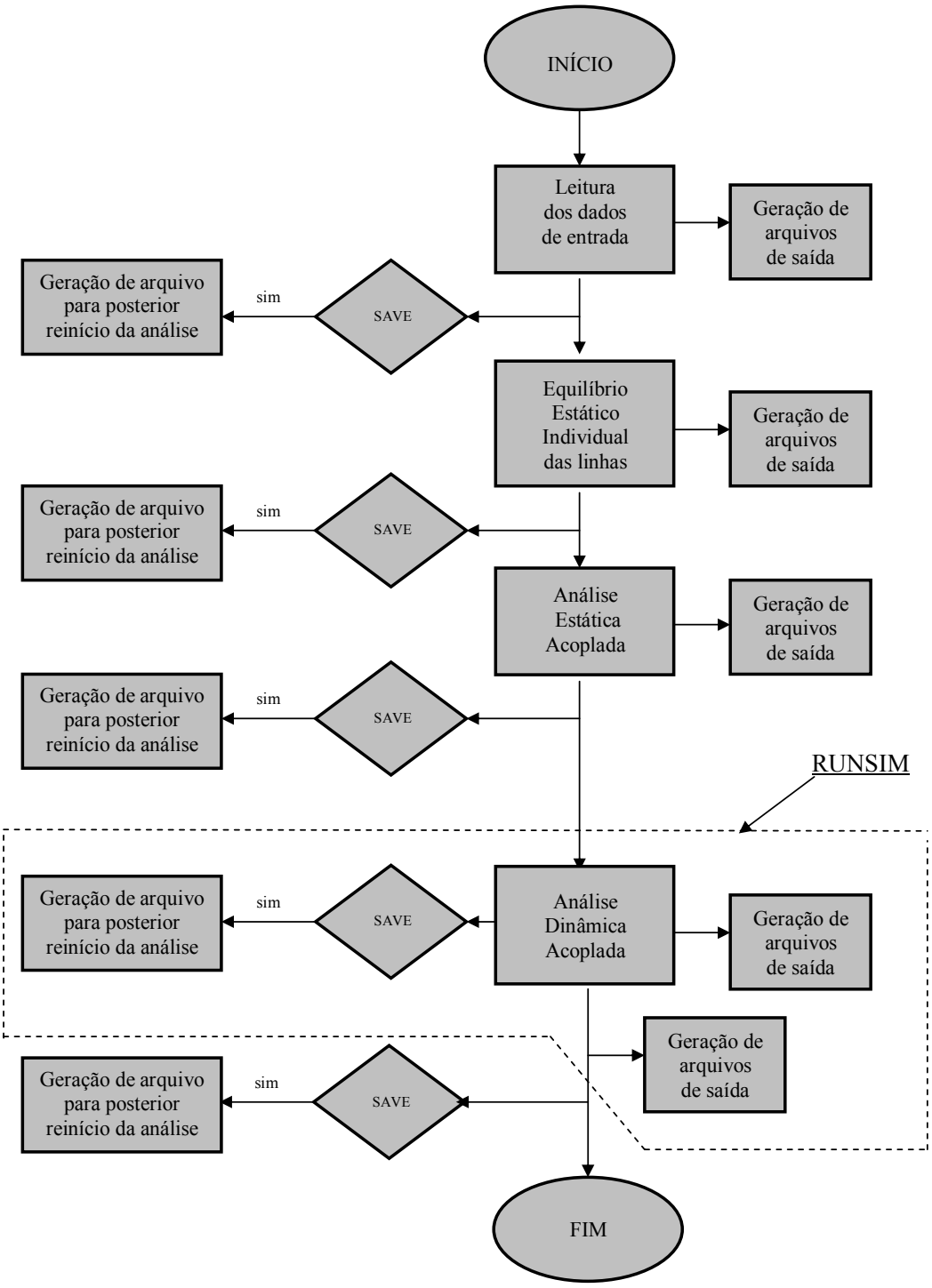


Figura 5.1 – Fluxograma Geral do Programa PROSIM.

ENTRADA DE DADOS

No módulo de entrada de dados efetua-se inicialmente a procura e leitura do arquivo `INPROSM.DAT` o qual fornecerá os prefixos dos arquivos de entrada e saída, bem como o caminho (*path*) dos arquivos de entrada. Em seguida efetua-se a leitura do arquivo com extensão `.dat` (APÊNDICE B), no qual contem todos os dados necessários para análise. Durante a entrada de dados, o programa também gera arquivos de saída contendo relatórios dos dados lidos.

GERAÇÃO DE ARQUIVOS DE SAÍDA

As atividades de "geração de arquivos de saída" ocorrem também em todos os demais módulos. Estes arquivos podem ser divididos em arquivos textos com relatórios, arquivos para pós-processamento gráfico e estatístico (APÊNDICE B). Alguns destes arquivos são gerenciados por diversas sub-rotinas e outros são gerenciados por uma única sob-rotina.

Para os arquivos gerenciados por apenas uma sub-rotina é fácil restringir a geração do mesmo apenas à um determinado processo. Quanto aos arquivos gerenciados por diversas sub-rotinas isto se torna mais trabalhoso pois seria necessário recodificar quase todo o código seqüencial para restringir a geração destes arquivos a um processo apenas predefinido. Este trabalho todo pode ser contornado fazendo com que cada processo gere um arquivo com mesma extensão mas com prefixo diferenciado de acordo com o processo. Por exemplo, no código seqüencial existe o arquivo de saída `prefout.LIS`, com a implementação paralela cada processo de rank n gera um arquivo `prefout_p(`n`).LIS` (APÊNDICE B). Este artifício é criado para que não ocorra erro durante a abertura dos arquivos.

Essa solução, fácil e prática, pode no entanto sobrecarregar o disco do sistema quando os arquivos são grandes, tais como, arquivos gerados em análises muito longas. Mais adiante será apresentada a metodologia adotada para solucionar tal problema.

EQUILÍBRIO INICIAL DAS LINHAS

No módulo “equilíbrio inicial das linhas” o programa efetua uma análise estática não-linear de cada linha individualmente, com o objetivo de obter a configuração inicial das linhas sob a ação apenas do peso próprio.

ANÁLISE ESTÁTICA ACOPLADA

O módulo posterior “análise estática acoplada” consiste basicamente no equilíbrio de forças e momentos resultantes dos carregamentos estáticos nas direções globais x, y e z, considerando o casco e o conjunto de todas as linhas.

O objetivo deste módulo é a obtenção de uma configuração equilibrada sob a ação das parcelas estáticas do carregamento. Com isso, pode-se iniciar a simulação dinâmica diretamente a partir da configuração de equilíbrio estático, desta forma reduzindo o transiente e tornando a análise dinâmica mais eficiente.

Para obtenção do equilíbrio do sistema utiliza-se um método puramente iterativo de Newton-Raphson. As parcelas de carregamento consideradas neste módulo são o peso próprio das linhas, peso da plataforma, empuxo da plataforma, cargas externas concentradas, correnteza e vento nas direções globais x, y e z. A parcela de deriva média de onda não é considerada neste módulo, pois o carregamento devido à onda depende da fase da mesma [23].

GERAÇÃO DE ARQUIVOS PARA POSTERIOR REINÍCIO DA ANÁLISE

Durante a execução do programa todos os dados e resultados da análise podem ser gravados em um arquivo permitindo um posterior reinicio da análise. A geração deste arquivo é acionada através da opção SAVE. O arquivo é gravado no formato binário ao final de cada módulo, e opcionalmente pode ser gravado também ao longo da análise dinâmica, em intervalos definidos pela variável TIMSAVE (APÊNDICE B).

5.2.1 Análise Dinâmica Acoplada

O módulo, “análise dinâmica acoplada”, se destaca pelo consumo de CPU. Com base nos tempos de processamento obtidos do código seqüencial, observou-se que a maior parte do tempo de execução do programa seqüencial se concentra neste módulo. Para os tempos de simulação usualmente empregados em uma análise dinâmica, este módulo consome praticamente a totalidade do tempo de processamento. Desta forma, pode-se dizer que para obter uma paralelização eficiente deve-se concentrar os esforços no módulo de análise dinâmica.

O módulo de “análise dinâmica acoplada” se subdivide basicamente em dois procedimentos que ocorrem simultaneamente:

- Solução das equações de movimento do casco, resultando nos movimentos da unidade flutuante, sob ação das forças ambientais e da tração de topo das linhas de ancoragem e risers;
- Solução das equações de movimento de cada linha de ancoragem e risers, resultando nas da forças de topo das mesmas, sob ação dos movimentos do casco e das cargas ambientais.

A Figura 5.2 resume o procedimento de análise dinâmica acoplada, a qual é gerenciada pela rotina RUNSIM. As rotinas RUNGK4 e NEWMRK incorporam o procedimento de integração no tempo para solução das equações de movimento do casco. A rotina RUNGK4 emprega o método de Runge-Kutta, enquanto NEWMRK emprega o método de Newmark, ambos descritos respectivamente no item 3.2.2 e 3.3.4.

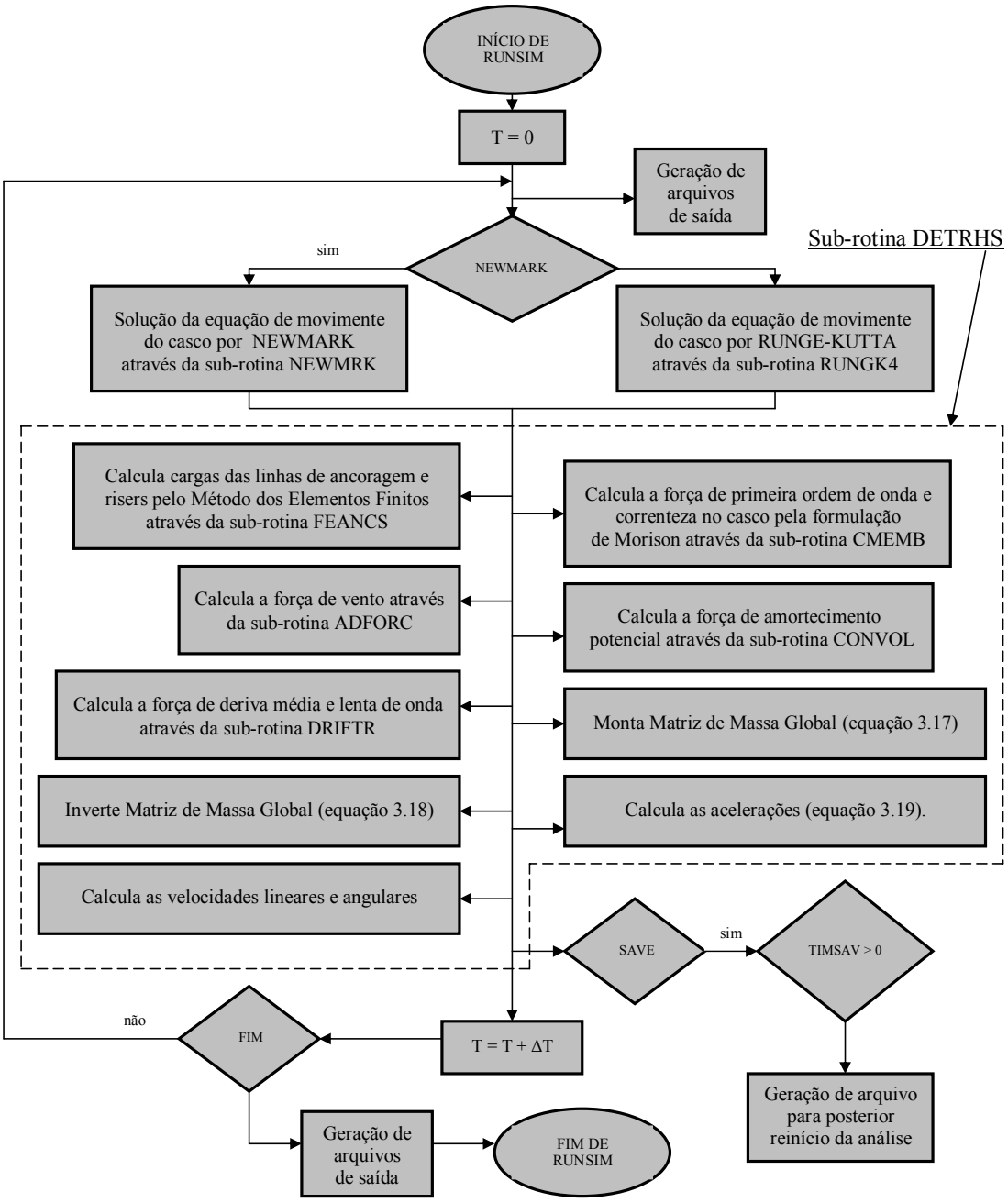


Figura 5.2 – Fluxograma Geral da Sub-rotina RUNSIM.

A sub-rotina DETRHS chama FEANCS, CMEMB, ADFORC, CONVOL e DRIFTR as quais gerenciam o cálculo das parcelas de força atuando sobre o casco da plataforma. Em seguida, na rotina DETRHS são obtidas as acelerações e velocidades translacionais e rotacionais da unidade flutuante. De volta à RUNGK4 ou NEWMRK

são obtidos os deslocamentos e rotações da unidade flutuante para o instante de tempo considerado, de acordo com o método empregado (Newmark ou Runge-Kutta).

Dentre as rotinas chamadas por DETRHS, inclui-se a rotina FEANCS que incorpora o módulo de análise das linhas de ancoragem e risers pelo Método de Elementos Finitos. Esta rotina recebe de DETRHS os deslocamentos da unidade flutuante no passo de tempo t , aplica-os no topo das linhas, resolve as equações de movimento das linhas para este passo de tempo, e determina os esforços no topo de cada linha no passo de tempo $t+\Delta t$, transferindo-os para o centro de gravidade da unidade flutuante. Finalmente, retorna o controle à rotina DETRHS para calcular o deslocamento da unidade flutuante no passo de tempo $t+\Delta t$.

5.2.2 Loop Paralelizado

Na sub-rotina FEANCS a análise de cada linha é efetuada de forma independente, sendo realizada seqüencialmente em um *loop* como descrito no trecho de código a seguir, onde NLINES é o número de linhas da unidade flutuante (linhas de ancoragem e risers) e FACTL(NLINES) é o vetor que indica se a linha está ativada (1 para linha ativada e 0 para linha desativada).

Assim, é natural observar que o paralelismo deve ser realizado na análise dinâmica das linhas, dividindo as linhas ativas entre os processos definidos no começo da execução do programa.

```
      SUBROUTINE FEANCS(…)
C
C     FORC(1-3,4-6)
C        1-3 força na plataforma no sistema fixo
C        4-6 momento na plataforma no sistema fixo

      REAL*8 FORC(6)
          ……
      DO 1000 ILINE = 1,NLINES
           ……
C se a linha estiver desativada go to 1000
          IF ( FACTL(ILINE) .EQ. 0.D0 ) GO TO 1000
          ……
          ISUBDT = 1
          WHILE ( ISUBDT .LE. NSUBDT ) ! subciclagem casco-linhas
             Integração das equações de movimento
             ISUBDT = ISUBDT + 1
          ENDDO
            ……
          Composição da força no topo da linha – FT(3,NNOPX)
          Composição da força no CG da plataforma - FT_SM ← FT
          Composição do momento no CG da plataforma - FM_SM ← FT
          FORC(1-3) ← FORC(1-3) + FT_SM(1-3)
          FORC(4-6) ← FORC(4-6) + FM_SM(1-3)
            ……
1000 CONTINUE
            ……
```

5.2.3 Subciclagem Casco-Linhas

No procedimento de análise acoplada implementada na primeira versão do Prosim, a cada instante do processo de integração no tempo das equações de movimento do casco efetuava-se uma análise não-linear dinâmica de cada linha sob a ação da onda, correnteza, peso próprio e das componentes de movimentos do casco.

No entanto, como em geral os intervalos de tempo requeridos para a análise dinâmica das linhas de ancoragem e risers são menores do que os intervalos de tempo

para a análise do casco, foi introduzido o conceito de subciclagem, no qual possibilita utilizar um passo de integração para as linhas NSUBDT vezes menor que o passo de integração para o casco (Figura 5.3). Desta forma, a cada análise do casco faz-se uma ou mais análises das linhas.

Certamente esta modificação proporcionou ganho de desempenho do programa seqüencial, possibilitando utilizar um passo de integração maior para o casco sem comprometer os resultados da análise dinâmica das linhas. No entanto, a grande contribuição foi dada à implementação paralela, pois com a subciclagem o tempo total de processamento das linhas aumenta em relação ao tempo de processamento total e o número de troca de dados diminui, como será exposto mais adiante.

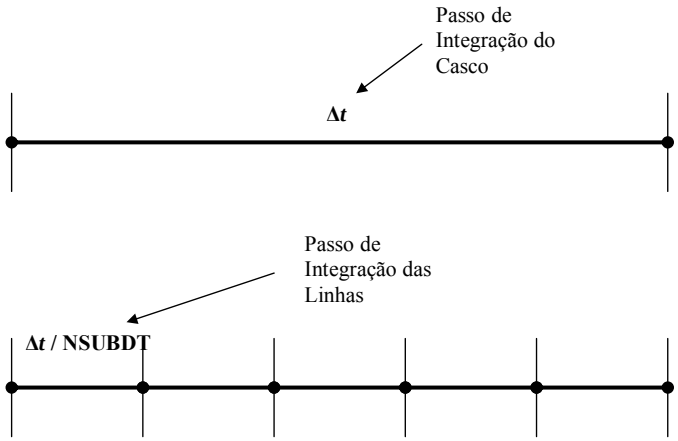


Figura 5.3 – Subciclagem Casco-Linhas

5.3 ESTRATÉGIA DE IMPLEMENTAÇÃO PARALELA

5.3.1 Introdução

O primeiro passo para a implementação paralela, depois do estudo detalhado do código seqüencial, foi definir os aspectos importantes para a implementação, destacando-se:

- distribuição de tarefas;
- troca de mensagens;
- balanceamento da carga computacional;
- gerenciamento de arquivos (entrada e saída);
- save e restart.

Cada um destes aspectos será comentado nos próximos itens, após a apresentação do fluxograma geral da implementação paralela.

O segundo passo foi definir o processador MASTER, o qual recebe a identificação com rank 0. O processo MASTER é responsável por grande parte das tarefas e pelo gerenciamento dos demais processos denominados de "escravos".

5.3.2 Fluxograma Geral

Na Figura 5.4 é exposto o fluxograma geral da implementação paralela. Nesta figura pode-se comparar algumas das modificações realizadas. Para que se compreenda melhor tais modificações é necessário retornar a Figura 5.1, onde é ilustrado o fluxograma geral do programa Prosim.

A troca de mensagens entre os processos foi dividida em quatro fases. As fases I,III e IV são ilustradas na Figura 5.4 e a fase II é ilustrada mais adiante no trecho de código do item 5.3.3.

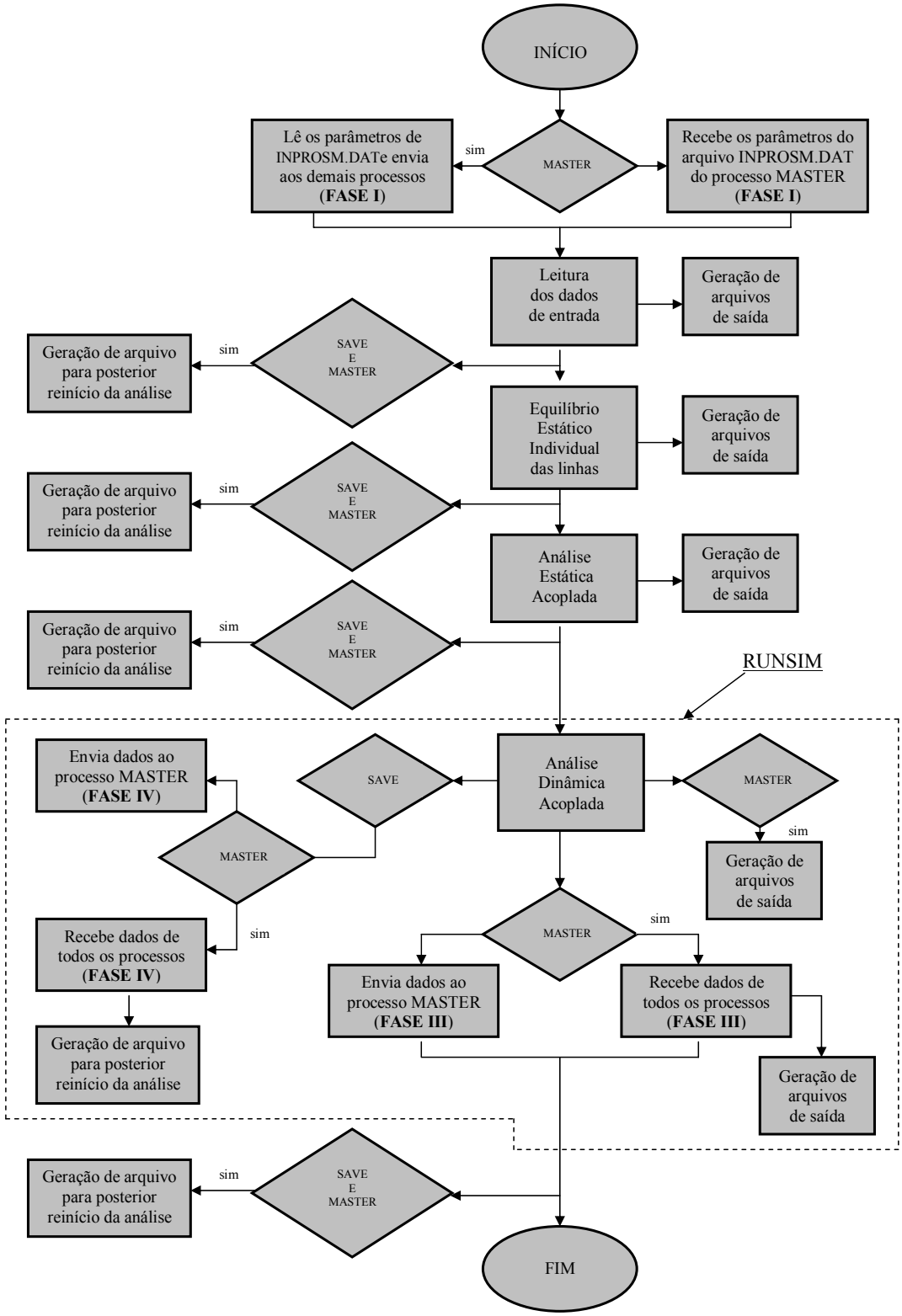


Figura 5.4 – Fluxograma Geral da Implementação Paralela.

5.3.3 Distribuição de Tarefas

Para a distribuição de tarefas foi adotada primeiramente uma distribuição seqüencial das linhas, a qual não leva em consideração o equilíbrio de carga computacional entre os processos.

Nesta distribuição cada processo recebe um conjunto de linhas, e cada conjunto é formado por linhas com numeração adjacente. O número de linhas de cada conjunto é igual ao quociente inteiro da divisão entre o total de linhas modeladas ativadas (no Prosim as linhas modeladas podem ser desativadas dependendo da análise) e a quantidade de processos. Caso a divisão não seja exata, incorpora-se uma linha a cada processo até a distribuição da totalidade das linhas ativadas restantes.

Em uma implementação paralela o número de processos pode ser diferente do número de processadores. No entanto, quando um processador realiza mais de um processo há troca de informação entre ele mesmo, afetando assim o desempenho, pois haverá comunicação desnecessária gerando desta forma um tempo de espera ocioso (overhead). Portanto, para não comprometer o desempenho do programa utiliza-se número de processos igual ao número de processadores disponíveis.

Vale ressaltar também que mesmo assim os processadores ainda podem ficar com carga computacional desequilibrada. Isso deve-se ao fato de que diferentes linhas podem ser modeladas por diferentes tipos de elemento, ou ainda empregando malhas com diferente número de elementos. Ou seja, um dado conjunto de linhas pode requerer uma carga computacional maior do que os demais conjuntos. Este tópico será tratado novamente mais adiante no item 5.3.5.

Outra dificuldade encontrada na distribuição das linhas são as linhas desativadas. Estas linhas podem estar dentro de um conjunto, mas, no entanto, não podem ser consideradas na análise dinâmica das linhas. Para solucionar este problema criou-se o array IOFFSET(TASKID+1), onde TASKID é o rank do processo e a variável IOFFSET(TASKID+1) define o número da primeira linha do conjunto de linhas destinado ao processo de rank TASKID. Desta forma, cada processo recebe um

conjunto de linhas pertencente ao intervalo (IOFFSET(TASKID+1), IOFFSET(TASKID+2)-1).

A Figura 5.5 exemplifica melhor a distribuição de linhas mencionado. Nela tem-se 3 processos e 11 linhas, sendo 3 linhas desativadas e dispostas aleatoriamente. Com a distribuição das linhas ter-se-á 3 linhas ativadas para o processo 0, 3 linhas para o processo 1 e 2 linhas para o processo 2.

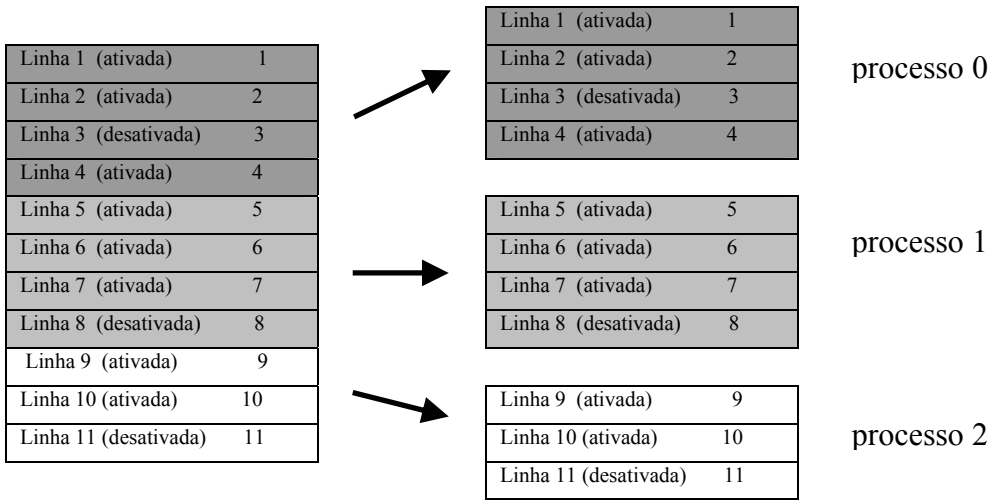


Figura 5.5 – Distribuição Seqüencial das Linhas.

Como foi mencionado anteriormente cada processo recebe um conjunto de linhas pertencentes ao intervalo (IOFFSET(TASKID+1), IOFFSET(TASKID+2)-1). Ou seja, por exemplo no processo 0 (MASTER), haverá processamento das linhas pertencentes ao intervalo (IOFFSET(1) = 1, IOFFSET(2)-1 = 4). Quanto a linha desativada que esteja dentro deste intervalo não será processada, pois já no código seqüencial há um desvio.

Este procedimento é também exemplificado no trecho de código a seguir, onde é demonstrada a modificação feita no *loop paralelizado*, na sub-rotina FEANCS. Observa-se no final do loop a segunda fase de comunicação entre os processos. Esta comunicação é a essência da implementação paralela e será mencionada novamente neste trabalho.

```
      SUBROUTINE FEANCS(…)
C
C     FORC(1-3,4-6)
C        1-3 força na plataforma no sistema fixo
C        4-6 momento na plataforma no sistema fixo
C      FORCSUM(1-3,4-6) - > vetor resultante da somatória do vetor FORC
C                           de cada processo
      REAL*8 FORCSUM(6)
      REAL*8 FORC(6)
         ……
C --- zera o vetor FORC caso o processo não for MASTER
       IF ( TASKID .NE. 0 ) THEN
          DO I = 1,6
             FORC(I) = 0.D0
          ENDDO
       ENDIF
       DO 1000 ILINE = IOFFSET(TASKID+1) , IOFFSET(TASKID+2)-1
          ……
C se a linha estiver desativada go to 1000
          IF ( FACTL(ILINE) .EQ. 0.D0 ) GO TO 1000
          ……
          ISUBDT = 1
          WHILE ( ISUBDT .LE. NSUBDT ) ! subciclagem casco-linhas
             Integração das equações de movimento
             ISUBDT = ISUBDT + 1
          ENDDO
            ……
          Composição da força no topo da linha – FT(3,NNOPX)
          Composição da força no CG da plataforma - FT_SM ← FT
          Composição do momento no CG da plataforma - FM_SM ← FT
          FORC(1-3) ← FORC(1-3) + FT_SM(1-3)
          FORC(4-6) ← FORC(4-6) + FM_SM(1-3)
            ……
1000  CONTINUE
       DO I = 1,6
          FORCSUM(I) = 0.D0
       ENDDO
       CALL MPI_ALLREDUCE(FORC,FORCSUM,6,MPI_DOUBLE_PRECISION,
                          MPI_SUM,MPI_COMM_WORLD,IERR)
          ……
       DO I = 1,6
          FORC(I) = FORCSUM(I)
       ENDDO
          ……
```

FASE II

5.3.4 Troca de Mensagens

Como mencionado anteriormente, a troca de mensagens entre os processos foi dividida em quatro fases:

- As fase I,III e IV, que correspondem às etapas iniciais (de leitura de dados) e finais (de tratamento dos resultados) são ilustradas na Figura 5.4;

- A fase II, que corresponde à paralelização efetuada durante a análise dinâmica propriamente dita, foi ilustrada no trecho de código da rotina FEANCS apresentado no item anterior.

FASE I

A primeira fase trata-se de uma comunicação muito rápida onde o processador MASTER envia, aos demais processadores, apenas informações necessárias para abertura dos arquivos de entrada. Efetivamente, a leitura dos arquivos de entrada de dados é feita por todos os processos aproveitando a fração de memória de disco compartilhada. Em cluster de PC's todos os processadores compartilham uma fração de memória do disco do PC administrador e desta forma todos processadores conseguem ler os arquivos contidos nesta fração de memória. Para distribuir os parâmetros foi empregada a operação de difusão realizada pelo comando MPI_BCAST.

FASE II

Na segunda fase de comunicação entre os processadores há uma troca de dados envolvendo todos os processos a cada passo de integração do casco, logo após o termino do *loop paralelizado*.

Os dados envolvidos nessa comunicação são as forças e os momentos resultantes das linhas atuantes na unidade flutuante, os quais compõem o vetor de forças FORC(6). Com o auxílio do comando MPI_ALLREDUCE este vetor é somado, em cada passo de integração do casco, ao vetor FORCSUM(6) e em seguida os valores de FORMSUM(6) são transferidos para FORC(6).

Este procedimento é a essência do código paralelizado. Através desta troca de dados é possível atualizar o vetor de forças de restauração atuante na unidade flutuante, em todos os processos, resultante da ação das linhas de ancoragem e risers. Vale salientar que apenas as tarefas no *loop paralelizado* são divididas entre os processos, enquanto que os demais procedimentos de cálculo (que requerem tempo de processamento menor), tais como os realizados na sub-rotina DETRHS, são realizados por todos os processos, evitando assim comunicação excessiva.

Normalmente, o passo de integração do casco é da ordem de 0,20 seg para a obtenção de uma boa precisão. Já o passo de integração das linhas é menor, da ordem de 0,02 seg. Como pode-se observar caso não houvesse subciclagem (descrita no item 5.2.3) a análise acoplada deveria ser feita com 0,02 seg. Com a subciclagem é possível empregar o valor de 0.20 para a análise acoplada, permitindo que se realize menos eventos de troca de informação da FASE II diminuindo o overhead. Para ilustrar, considere-se uma análise com tempo total de simulação de 2.400 seg, e com os valores de intervalo de tempo mencionados acima (0,20 seg para o casco e 0,02 para as linhas); com a subciclagem haverá 12.000 eventos ao invés de 120.000.

Apesar de pequena, a troca de dados nesta fase exerce forte influencia sobre o desempenho do programa. Esta influência não se deve ao fato desta comunicação ser realizada em todos os passos de integração do casco, mas sim, por ser uma comunicação sincronizada onde todos os processos devem realizá-la ao mesmo tempo. Por este motivo quando a carga computacional entre os processadores não for balanceada conseqüentemente haverá processador que ficará ocioso. Este tempo ocioso aumentará o overhead e conseqüentemente ter-se-á uma diminuição da performance do programa.

FASE III

Na terceira fase de comunicação entre os processos, que ocorre no final da análise dinâmica acoplada, o processo “MASTER” recebe dos demais processos todos os resultados necessários para geração dos arquivos de saída (Figura 5.6). A troca de mensagem nesta fase é facilitada pelo modo como os dados são armazenados no código seqüencial.

No Prosim a maior parte dos dados, principalmente referentes as linhas, são armazenados em um só array inteiro IA(70000000) [18]. O mesmo ocorre com caracteres, sendo armazenados no array CA(70000000) e declarados como CHARACTER*4 (onde cada posição do array possui 4 caracteres). A alocação das posições são feitas de forma eficiente através de apontadores que são calculados pelas funções IDEFAR/IDEFI/IDEFAC as quais definem respectivamente um array real, inteiro e caractere [18].

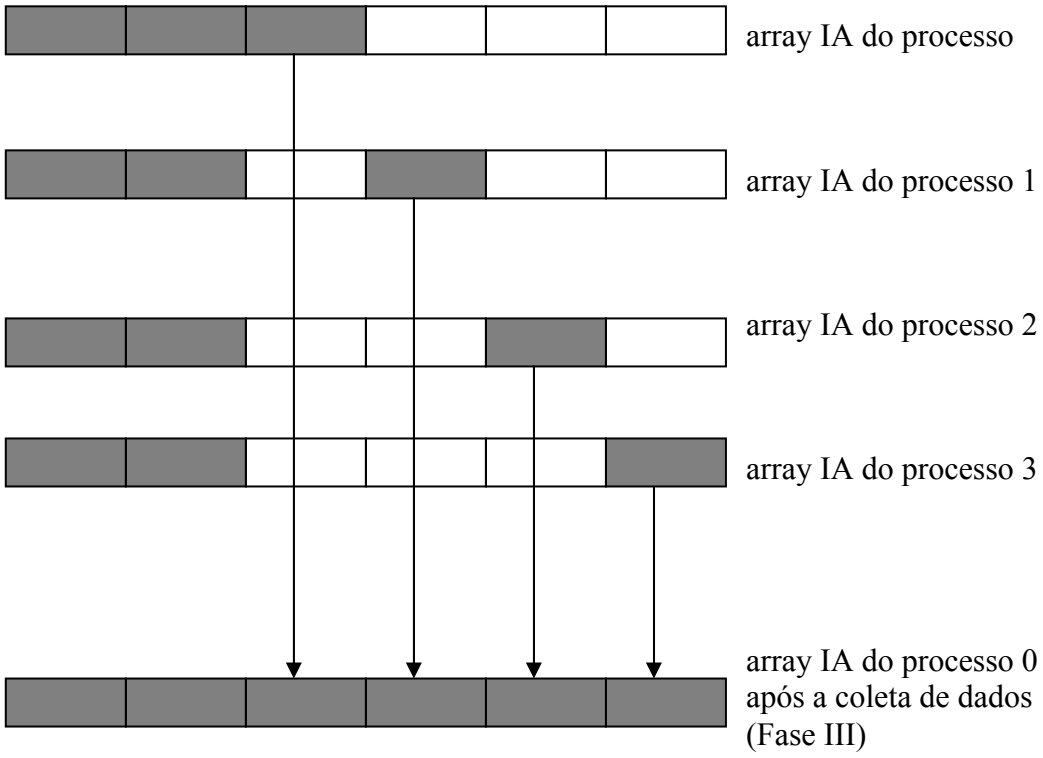


Figura 5.6 – Ilustração do Armazenamento e Coleta de Dados.

Na implementação paralela esta forma de armazenamento dos dados não altera. Cada processo terá seu próprio array e a forma de alocação dos ponteiros será idêntica em todos os processos. Contudo, vale ressaltar que o conteudo do array de um processo não é idêntico ao conteúde do array de um outro processo, pois cada processo opera sobre diferentes conjunto de dados. Portanto ao final da análise é necessário coletar os

dados que não são manipulados pelo processo MASTER para gerar corretamente os arquivos de saída.

Para realizar a comunicação desta última fase empregou-se então a comunicação ponto a ponto adotando as funções MPI_SEND e MPI_RECV. As funções MPI_SEND e MPI_RECV foram empregadas por oferecem segurança e eficiência na comunicação. Vale salientar que apenas parte dos dados armazenados nos trechos do array IA pertencentes aos demais processos serão coletados pelo processo MASTER na fase de comunicação, pois são necessários somente os dados utilizados para gerar os arquivos de saída. A coleta de todos os dados ocorre na quarta fase de comunicação e ocorre quando a opção SAVE é ativada e TIMSAVE é maior que zero.

FASE IV

A quarta fase de comunicação entre os processos é semelhante a terceira fase de comunicação a diferença está na quantidade de dados que são transferidos. A troca de dados entre os processos nesta fase é realizada quando o módulo SAVE está ativado. Conseqüentemente, é necessário que os dados estejam completos, ou seja, que o array IA(70000000) esteja completo. Desta forma, todos os dados que não foram manipulados pelo processo MASTER terão que ser coletado pelo mesmo. Este procedimento de coleta dos dados pelo processo MASTER é idêntico a terceira fase de comunicação, mencionada anteriormente.

5.3.5 Balanceamento de Carga Computacional

O balanceamento de carga foi adotado a partir do momento em que se verificou a existência de um overhead muito grande, proveniente do tempo espera de alguns processadores. Este tempo de espera é conseqüência do desequilíbrio da carga computacional. Alguns processadores são sobrecarregados recebendo tarefas maiores do que os demais, devido por exemplo ao diferente número de elementos das malhas de cada linha e, conseqüentemente, tendem a consumir um tempo de processamento maior. Com o objetivo de diminuir a carga extra gerada pelo desequilíbrio da carga

computacional buscou-se portanto criar uma forma de distribuição simples e ao mesmo tempo eficaz capaz de realizar o balanceamento desta carga computacional.

Para alcançar este objetivo foi criada uma forma de distribuição, também seqüencial, mas buscando o equilíbrio entre o número total de equações da malha de cada elementos finitos atribuído a cada processador. Neste procedimento tenta-se aproximar o número de equações de um processador da média (número total de equações considerando todas as linhas ativadas / número de processadores).

No processo de montagem dos conjuntos de linhas soma-se o número de equações da linha cada vez que se adiciona esta linha ao conjunto. Caso esta soma ultrapasse a média, verifica-se se a diferença entre esta soma e a média é menor que a diferença entre a média e esta soma sem as equações da última linha adicionada, caso não seja verdade, a última linha adicionada passa a pertencer ao próximo conjunto de linhas. Apesar da modificação do procedimento de distribuição das linhas manteve-se o array IOFFSET(TASKID+1) para identificar o intervalo do conjunto.

5.3.6 Gerenciamento de Arquivos

Na implementação paralela, o gerenciamento de arquivo foi destinado principalmente ao processo MASTER (RANK 0). A ele fica incumbida a carga extra de ler os parâmetros contidos no arquivo INPROSM.DAT (APÊNDICE B) e gerar os arquivos de pós-processamento gráfico e estatístico com extensão `.WAV`, `.WIND`, `.PLD`, `.PLT`, `.PLP`, `.GR2`, `.GR3`, `.GR4`, `.GR5`, `.GLT`, `.GRD`, `.GRV`, `.GRC` e `.GTD` (APÊNDICE B). A gravação desses arquivos apenas pelo processo MASTER é facilmente manipulada graças a metodologia de alocação de ponteiros empregada no programa prosim descrito anteriormente e por ser gerenciada por uma única sub-rotina. Como todos os dados são armazenados em um único array os dados podem ser gravados apenas no final da análise, restringindo assim esta etapa a um só processo, não tendo que criar vários arquivos de saída.

Além dos arquivos citados anteriormente são gerados também pelo Prosim os arquivos-textos com extensão `.LIS` (relatório) e `.DBG` (depuração) e os arquivos para

pós-processamento .DEF, .SNP, .MAP e .RES (APÊNDICE B). O controle de impressão por apenas um processo implicaria a recodificação quase que total do código seqüencial, e troca de mensagens desnecessárias que afetariam a performance do código paralelizado.

Para solucionar este problema adota-se a metodologia onde cada processo gera um arquivo com a mesma extensão. Desta forma, tem-se na análise estática *N* (número de processos) arquivos idênticos. Pois vale lembrar que a análise estática é realizada por todos os processos. Já na análise dinâmica, estes arquivos poderão ter trechos diferentes. Por exemplo, o arquivo SNP contém deslocamentos de todos os nós do reticulado do casco, e das malhas de elementos finitos das linhas. Os resultados do casco são gravados fora do loop paralelizado, e portanto serão idênticos em todas as instâncias deste arquivo nos diferentes processadores. Já os resultados das linhas são gravados dentro do loop paralelizado, e portanto serão diferenciados de um processador para outro.

Como já foi mencionada essa adoção pode sobrecarregar o disco do sistema quando os arquivos forem muito grandes. Para minimizar este problema pode-se ao longo da análise eliminar alguns destes arquivos, mantendo apenas os estritamente necessários. Por exemplo, os arquivos gerados na análise estática (<prefout>_s.DEF, <prefout>_s.SNP, <prefout>_s.MAP e <prefout>_s.RES) são sempre idênticos e na verdade é necessário apenas um. Portanto, deixa-se apenas o arquivo gerado pelo processo MASTER, os demais são eliminados ao final da análise estática. Mesmo assim é preciso gerá-los com prefixos diferentes para não ocorrer erro na abertura destes arquivos. Neste caso, acrescentou-se ao prefixo o caractere _p(rank do processo) (APÊNDICE B).

Quanto aos demais arquivos da análise dinâmica (<prefout>_d_0.DEF, <prefout>_d_0.SNP, <prefout>_d_0.MAP e <prefout>_d_0.RES) realiza-se o mesmo procedimento de diferenciação dos arquivos. Mas ao contrário da análise estática eles não são apagados, portanto, tem-se no final da análise *N* arquivos com a mesma extensão, diferenciado pelo atributo _p(rank do processo) (APÊNDICE B). Vale ressaltar que este procedimento é também adotado para os arquivos <prefout>.LIS e <prefout>.DBG, mencionados no terceiro parágrafo deste item.

5.3.7 Save e Restart

Inicialmente, pensou-se em adotar, para gravação do arquivo <prefres>.SAV (gerado quando é ativada na análise dinâmica a opção SAVE e quando TIMSAV é maior que zero) a mesma estratégia adotada para gerar os arquivos <prefout>.LIS, <prefout>.DBG, <prefout>_d_0.DEF, <prefout>_d_0.SNP, <prefout>_d_0.MAP e <prefout>_d_0.RES, mencionados no item anterior. Nesta estratégia teria-se *N* arquivos <prefres>_p(rank do processo).SAV e para reiniciar a análise deveria-se adotar o mesmo número de processos, visto que os dados estão espalhados pelos arquivos. Outro inconveniente é o excessivo consumo de espaço visto que os arquivos <prefres>.SAV são muito grandes.

A segunda alternativa estudada, e que foi adotada na implementação final consiste da coleta de todos os dados que são gravados no arquivo <prefres>.SAV. Esta coleta é facilitada pela forma como os dados são armazenados. Este procedimento de armazenamento de dados já foi mencionando no item 5.3.4, na FASE III, e a forma de coleta de dados é feita pela quarta fase de comunicação entre os processos.

Com esta alternativa tem-se apenas um arquivo único <prefres>.SAV, diminuindo assim, o consumo de espaço em disco. Além disso, verificou-se que o tempo necessário para realizar a coleta de dados é pequeno. Mesmo assim a performance do programa paralelo tende a cair pois o processo de geração do arquivo <prefres>.SAV consome um tempo de processamento considerável, aumentando portanto a parcela seqüencial e conseqüentemente diminuindo a parcela paralelizável.

6 APLICAÇÕES NUMÉRICAS

6.1 INTRODUÇÃO

Neste capítulo apresentam-se estudos de performance obtida com as estratégias de implementação paralela empregadas sobre o código seqüencial do programa Prosim, na análise acoplada de unidades flutuantes com linhas de ancoragem e risers. Foram realizadas duas aplicações:

- A primeira aplicação teve como objetivo estudar a influência da subciclagem casco-linhas e do módulo SAVE, responsável pela geração do arquivo para posterior reinício da análise. Nesta aplicação escolheu-se a plataforma de perfuração P10 por se tratar de um modelo onde todas as linhas possuem a mesma malha de elementos finitos. Desta forma sabe-se a princípio que há um balanço natural da carga computacional caso a divisão das linhas seja exata;

- Já na segunda aplicação o objetivo foi estudar uma aplicação mais complexa envolvendo linhas de ancoragem e diversos risers, diferentes entre si, e com malhas diferentes. Para isto foi escolhida a plataforma de produção P18 por se tratar de um modelo mais complexo com 8 linhas de ancoragem e 73 risers.

Para aplicação numérica destes dois exemplos utilizou-se o cluster instalado no LAMCSO (Laboratório de Métodos Computacionais e Sistemas Offshore), o qual apresenta as seguintes características:

Tabela 6.1 – Especificação do cluster.

Processadores	9 processadores Pentium III 850 MHz
Memória RAM	256 Mbytes por unidade
Dispositivo de Comunicação	Rede Fast Ethernet 100 Mbps
Sistema Operacional	Linux Red Hat
Software de Comunicação	LAM/MPI versão 6.5.5

6.2 PLATAFORMA DE PERFURAÇÃO P10

6.2.1 Características do Casco

A plataforma P10 é uma semi-submersível de perfuração com as seguintes características principais:

- LDA : 1200 m;
- azimute: 90^{o};
- posição do centro de gravidade em relação ao eixo local: 16 m;
- calado da plataforma: 20,0 m;
- peso da plataforma: 2,3 E+05 kN.

Na Figura 6.1 é ilustrado o sistema de referência local (XYZ) e global (xyz) adotado. Verifica-se que o eixo x do sistema global possui o sentido positivo da proa a popa, os eixos x e y do sistema global formam um plano horizontal com nível do mar, os eixos X e Y do sistema local ou eixo estrutural formam um plano horizontal com a quilha.

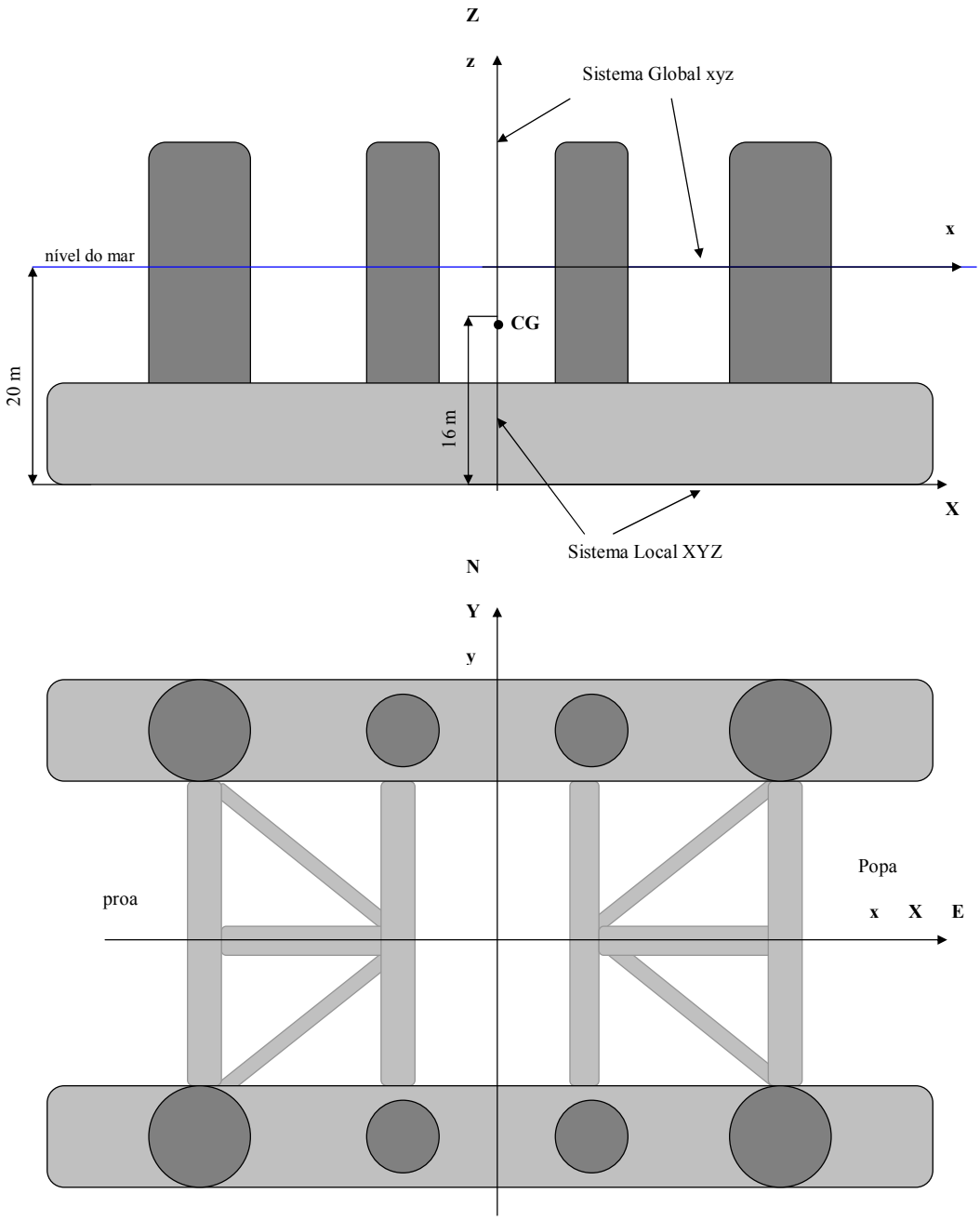


Figura 6.1 - Sistema de Referência da Plataforma P10.

6.2.2 Características das Linhas de Ancoragem

A P10 é ancorada por um total de 8 linhas de ancoragem em configuração de catenária simples. As linhas são compostas por um trecho de amarra no fundo e no topo e cabo de aço intermediário. As características mais detalhadas das linhas de ancoragem são apresentadas na Tabela 6.2 e na Tabela 6.3.

Tabela 6.2 - Propriedades das Linhas de Ancoragem da Plataforma P10.

Linha	Comprimentos (m)			Pré-tensão (kN)
	Amarra de Fundo	Cabo de Aço	Amarra de Topo	
Linha 1	926	1380	300	1292.2
Linha 2	926	1380	300	1298.9
Linha 3	926	1380	300	1355.6
Linha 4	926	1380	300	1340.4
Linha 5	926	1380	300	1340.4
Linha 6	926	1380	300	1355.6
Linha 7	926	1380	300	1298.9
Linha 8	926	1380	300	1292.2

Tabela 6.3 - Coordenadas de Conexão das Linhas de Ancoragem da Plataforma P10.

Linha	Coordenadas Globais			Azimute (^{o})
	X	Y	Z	
Linha 1	-35,00	32,50	-5,91	295.0
Linha 2	-31,00	32,50	-5,91	335.0
Linha 3	35,00	32,50	-5,91	25.0
Linha 4	35,00	32,50	-5,91	65.0
Linha 5	35,00	-32,50	-5,91	115.0
Linha 6	31,00	-32,50	-5,91	155.0
Linha 7	-31,00	-32,50	-5,91	205.0
Linha 8	-35,00	-32,50	-5,91	295.0

6.2.3 Dados Ambientais

Para análise em questão foram considerados os seguintes carregamentos ambientais:

- Carregamentos atuantes no casco: correnteza;
- Carregamentos atuantes nas linhas: gravitacional e amortecimento hidrodinâmico.

O perfil de corrente utilizado na simulação com a P10 é o apresentado a seguir.

Tabela 6.4 - Perfil de Correnteza Atuante na Plataforma P10.

Profundidade (m)	Velocidade (m/s)	Direção
0	1.07	NW
50	1.02	NW
100	0.97	NW
140	0.76	NW
230	0.89	NW
340	0.84	NW
415	0.68	NW
545	0.56	NW
640	0.54	NW
750	0.5	NW
915	0.48	NW
1200	0,00	NW

6.2.4 Modelo Numérico

CASCO

A embarcação é representada por um modelo reticulado de 36 elementos tubulares. Os dados necessários para compor o modelo são as coordenadas dos nós iniciais e finais de cada elemento, seus diâmetros equivalentes, e os respectivos coeficientes hidrodinâmicos.

As Figuras 6.2 apresentam a visualização sólida da plataforma P10 levando em conta os diâmetros dos membros tubulares equivalentes.

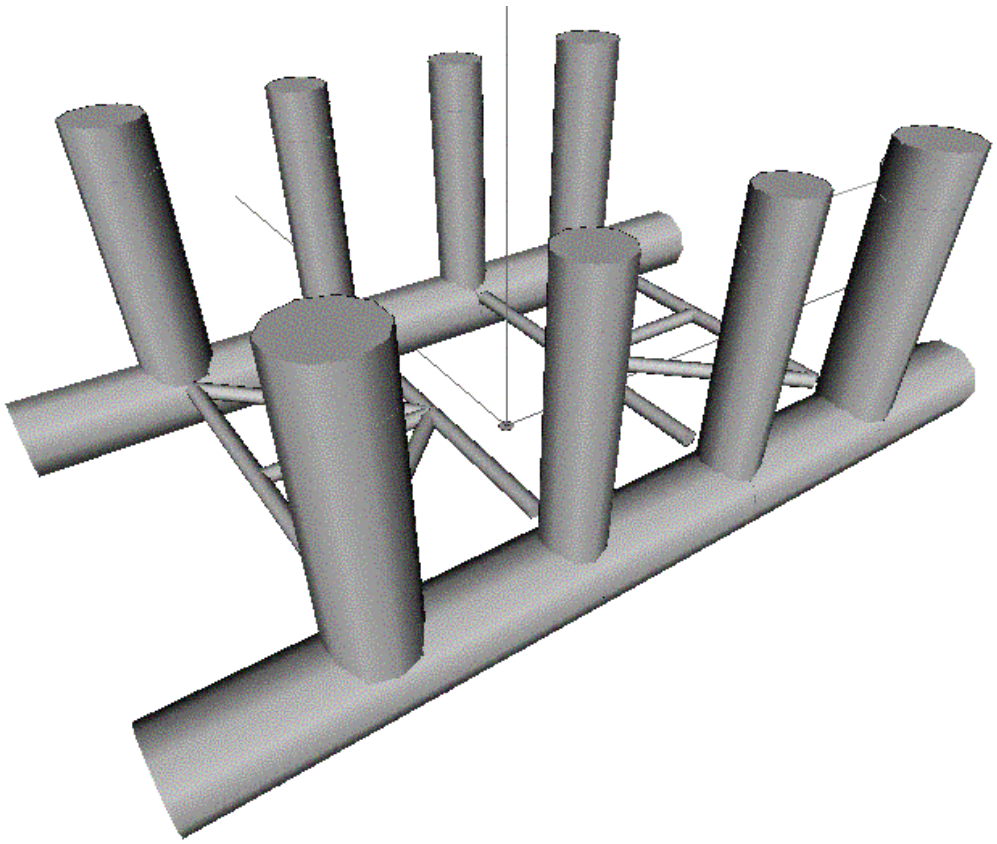


Figura 6.2 - Visualização Sólida do Modelo do Casco da P10.

LINHAS DE ANCORAGEM

A discretização das linhas foi realizada com uma malha uniforme de elementos de treliça, resultando num total de 88 elementos de treliça para cada linha. Cada elemento tem um comprimento de 30 metros.

Como todas as linhas são modeladas pelo mesmo tipo de elemento e têm o mesmo número de elementos, pode-se dizer que se trata de um problema onde o balanço da carga de processamento entre os processadores é perfeito, com relação ao número de equações processado por cada processador, a não ser é claro, em casos onde a divisão do número de linhas pelo número de processadores não resultasse em um número inteiro.

Na Figura 6.3 é ilustrado uma das linhas da plataforma P10 discretizada com elemento de treliça.

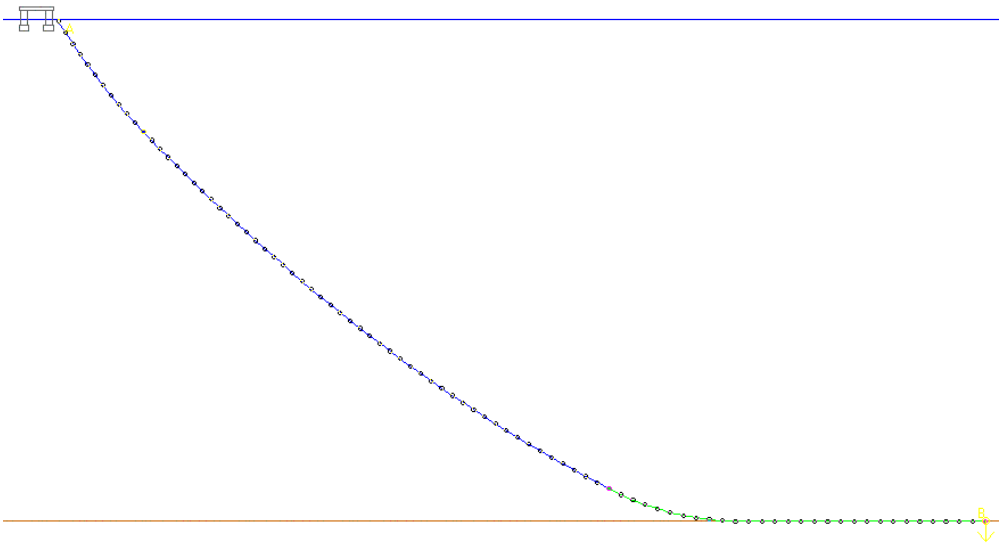


Figura 6.3 - Discretização de uma Linha de Ancoragem da Plataforma P10.

MODELO ACOPLADO

A seguir são apresentadas figuras ilustrando o modelo acoplado da P10, incorporado as linhas de ancoragem.

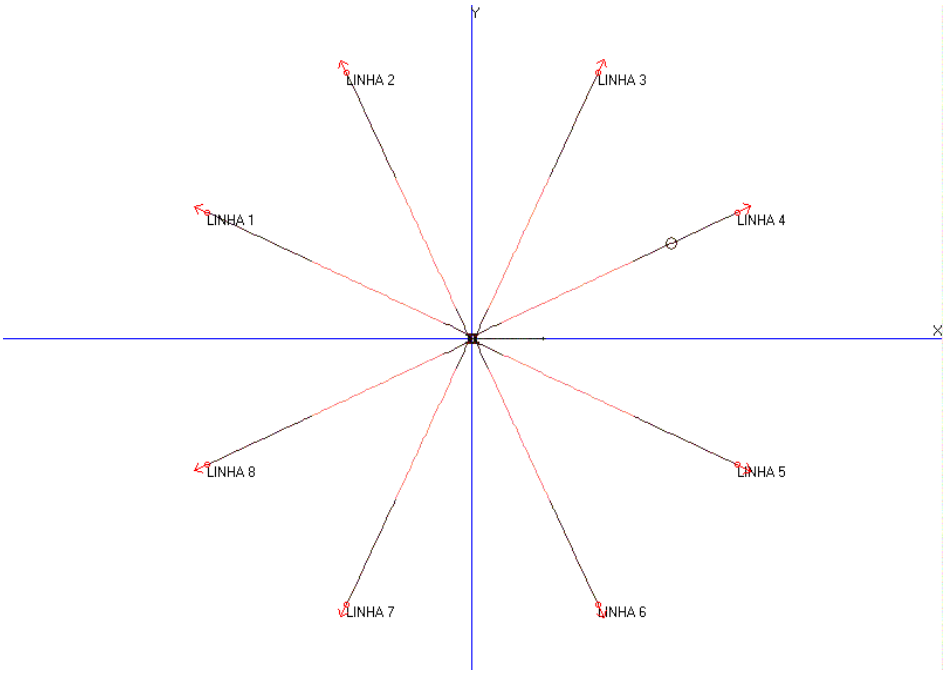


Figura 6.4 - Modelo Acoplado da Plataforma P10 – Vista XY.

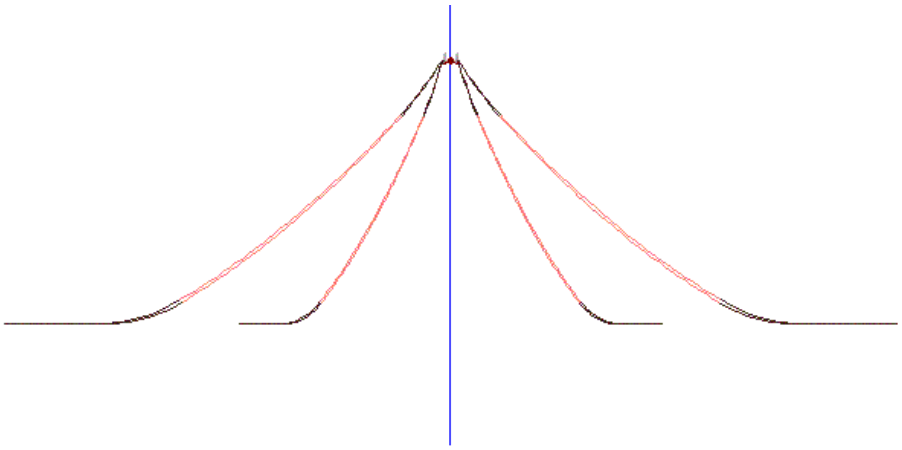


Figura 6.5 - Modelo Acoplado da Plataforma P10 – Vista XZ.

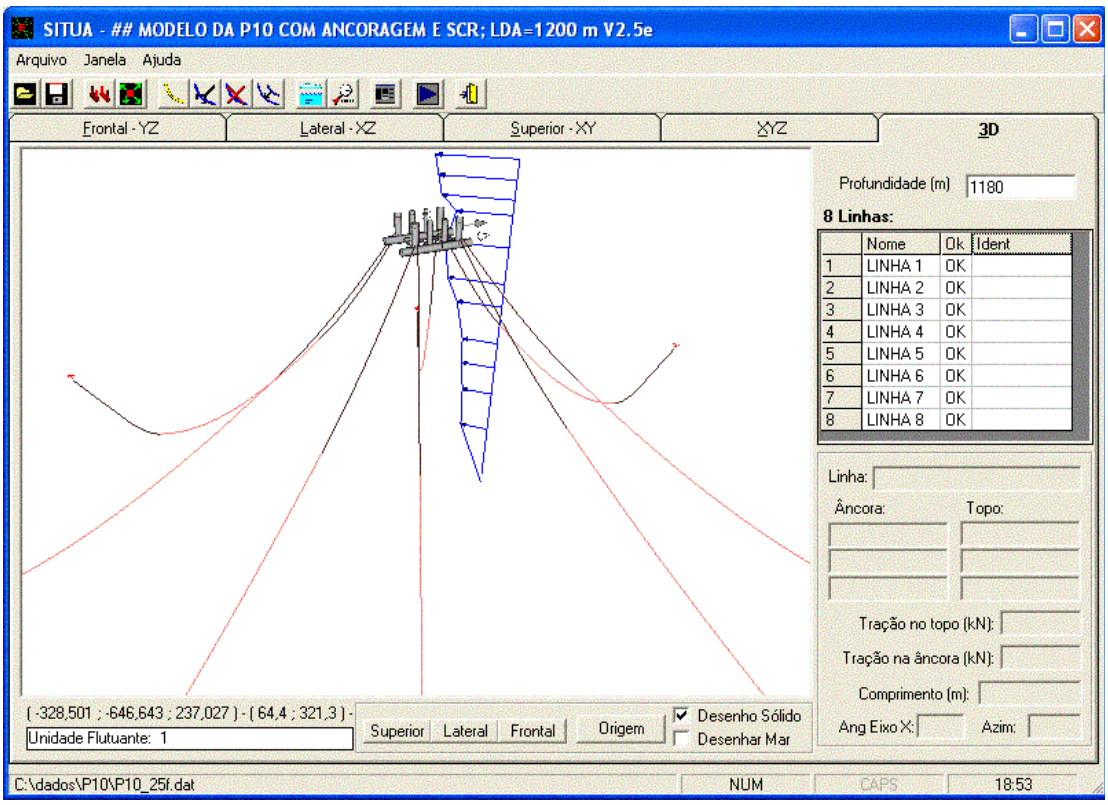


Figura 6.6 - Modelo Acoplado da Plataforma P10 - Vista 3D.

6.2.5 Análise de Performance

Para a análise de performance neste exemplo foram realizados dois conjuntos de simulações: com e sem módulo SAVE. Para cada conjunto foram realizados três grupos de análises com níveis de subciclagem casco-linhas diferenciados, e em cada grupo foi variado o número de processadores para avaliar a performance em termos de speed-up e eficiência.

SIMULAÇÃO SEM MÓDULO SAVE

Foram efetuadas três simulações sem ativar o módulo SAVE, com diferentes valores pra o intervalo de integração das linhas no procedimento de subciclagem casco-linhas. A Tabela 6.5 descreve os parâmetros adotados nestas análises sem módulo SAVE.

Tabela 6.5 - Parâmetros de Análise: P10 / sem SAVE.

Simulação	S_P10_1	S_P10_2	S_P10_3
Tempo total de análise	3.600,00 seg	3.600,00 seg	3.600,00 seg
Intervalo de integração do casco	0,20 seg	0,20 seg	0,20 seg
Intervalo de integração das linhas	0,02 seg	0,04 seg	0,20 seg
NSUBDT	10	5	1

Na Tabela 6.6 são apresentados os tempos de processamento em cada etapa (ver item 5.2) das análises S_P10_1, S_P10_2 e S_P10_3 da P10 sem o módulo SAVE.

Tabela 6.6 - Tempo de Processamento Seqüencial: P10 / sem SAVE.

Simulação	S_P10_1	S_P10_2	S_P10_3
Entrada de dados	0,42 seg	0,36 seg	0,36 seg
Equilíbrio inicial	0,11 seg	0,11 seg	0,11 seg
Análise estática acoplada	0,66 seg	0,64 seg	0,64 seg
Análise dinâmica acoplada	18.133,20 seg	6.916,68 seg	2.151,47 seg
Tempo total de processamento	18.134,39 seg	6.917,17 seg	2.152,58 seg

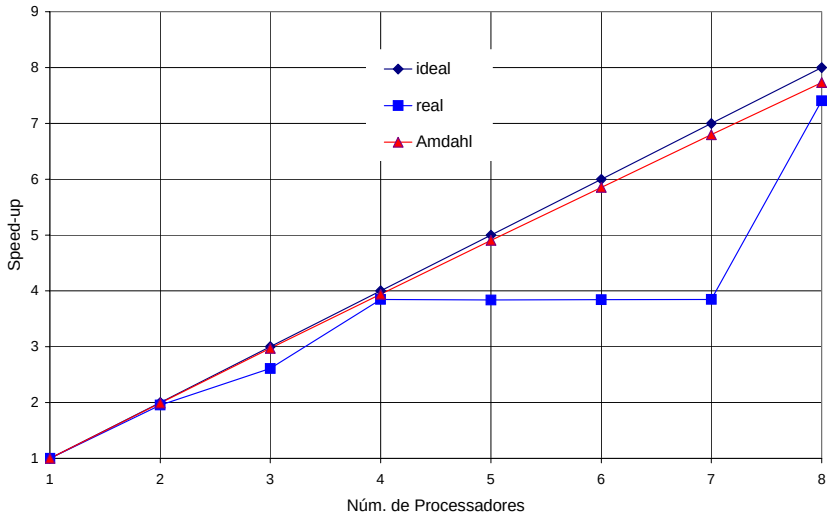
Tabela 6.7 - Fração Paralelizável Apontada pela Lei de Amdahl: P10 / sem SAVE.

Análise	S_P10_1	S_P10_2	S_P10_3
Loop paralelizado	18.053,43 seg	6.836,92 seg	2.071,132 seg
Fração paralelizável	99,55 %	98,83 %	96,23 %

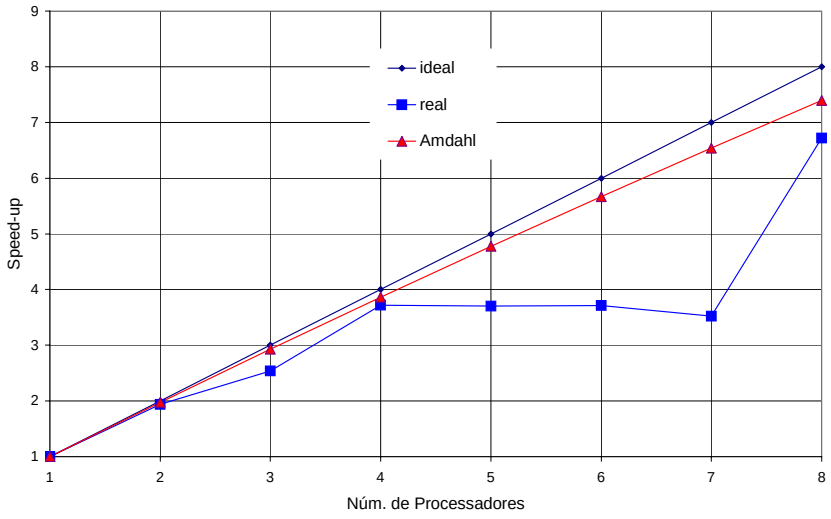
Na Tabela 6.7 verifica-se que a fração definida pela lei de Amdahl diminui a medida que o nível de subciclagem casco-linhas diminui. Este fato já era esperado, pois à medida que a subciclagem casco-linhas diminui automaticamente o processamento das linhas diminui.

O efeito disto é observado na Figura 6.7 onde verifica-se a perda de performance à medida em que o intervalo de integração das linhas se aproxima do intervalo de integração do casco. A mesma perda de performance com intervalo de integração das linhas próximo ao intervalo de integração do casco é comprovada na Figura 6.8, onde é apresentada a eficiência do código paralelizado.

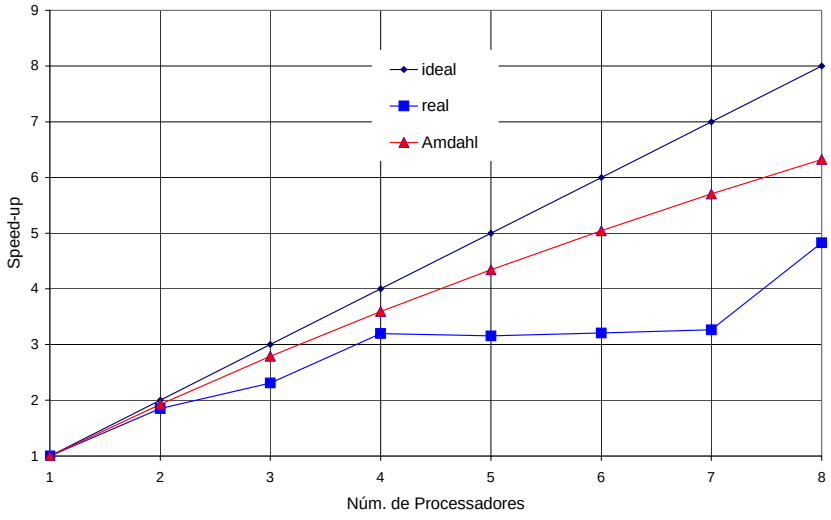
Outro fato observado na Figura 6.7 são os resultados para 2, 4 e 8 processadores, eles se aproximam do Speed-up ideal. Por outro lado os resultados para 3, 5, 6, e 7 processadores sofrem quedas bruscas. Este fato se deve ao desequilíbrio da carga computacional, já que nestes casos a divisão das linhas pelo número de processadores não é exata.



(a)

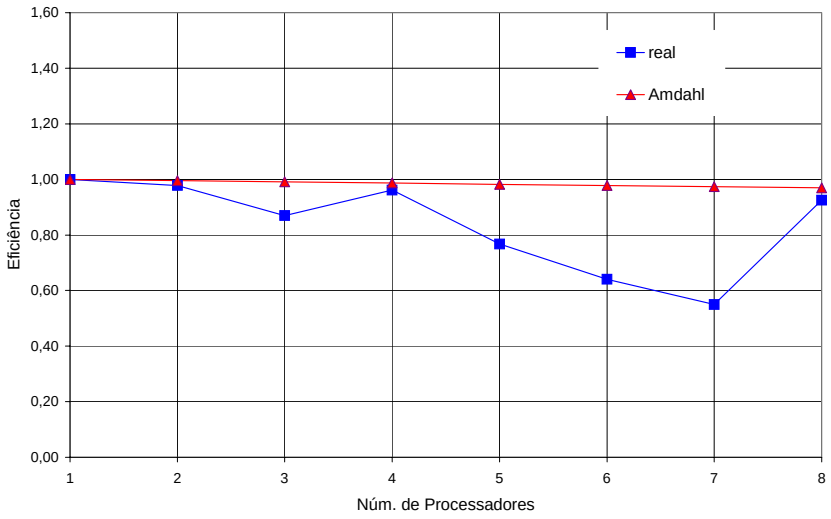


(b)

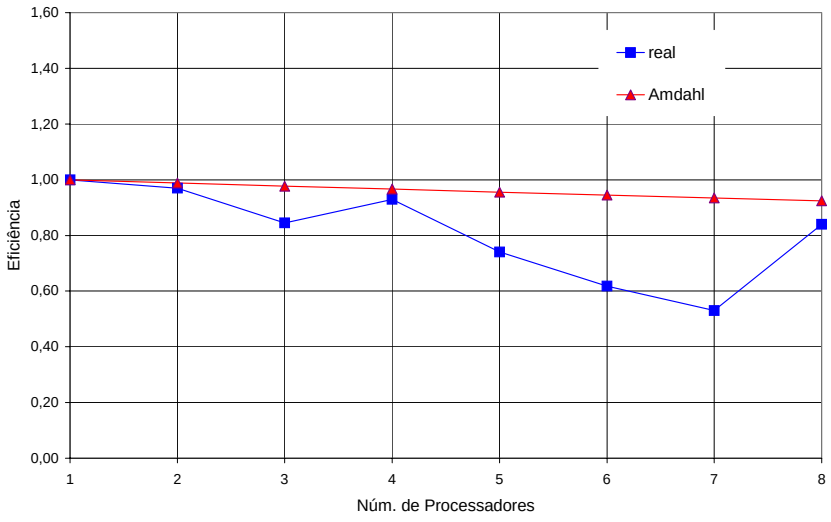


(c)

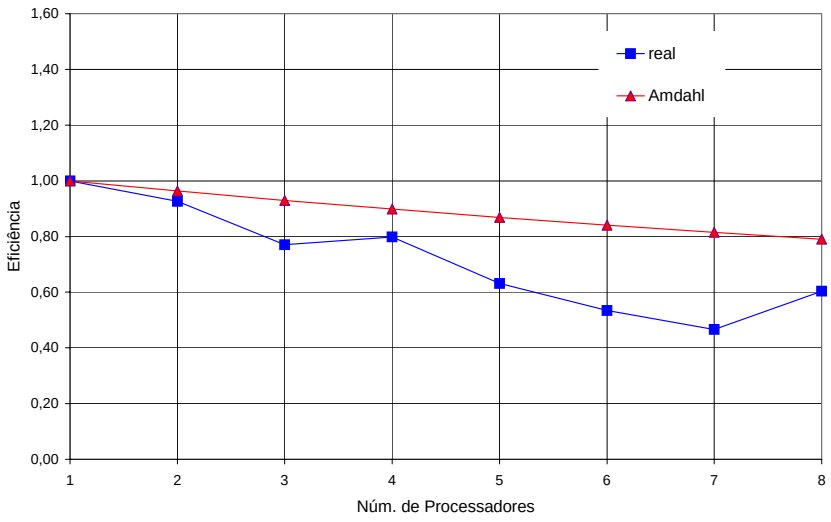
Figura 6.7 - Speed-up: (a) S_P10_1; (b) S_P10_2; (c) S_P10_3.



(a)



(b)



(c)

Figura 6.8 - Eficiência: (a) S_P10_1; (b) S_P10_2; (c) S_P10_3.

SIMULAÇÃO COM MÓDULO SAVE

Foram efetuados três simulações com o módulo SAVE ativado que se diferenciam pelo intervalo de integração das linhas. A Tabela 6.8 descreve os parâmetros adotados nas análises com módulo SAVE.

Tabela 6.8 - Parâmetros de Análise: P10 / com SAVE.

Simulação	S_P10_4	S_P10_5	S_P10_6
Tempo total de análise	3.600,00 seg	3.600,00 seg	3.600,00 seg
Intervalo de integração do casco	0,20 seg	0,20 seg	0,20 seg
Intervalo de integração das linhas	0,02 seg	0,04 seg	0,20 seg
NSUBDT	10	5	1
TIMSAVE	400,00 seg	400,00 seg	400,00 seg

Na Tabela 6.9 são apresentado os tempos de processamento em cada etapa das análises S_P10_4 S_P10_5 e S_P10_6 da P10 com módulo SAVE.

Tabela 6.9 - Tempo de Processamento Seqüencial: P10 / com SAVE.

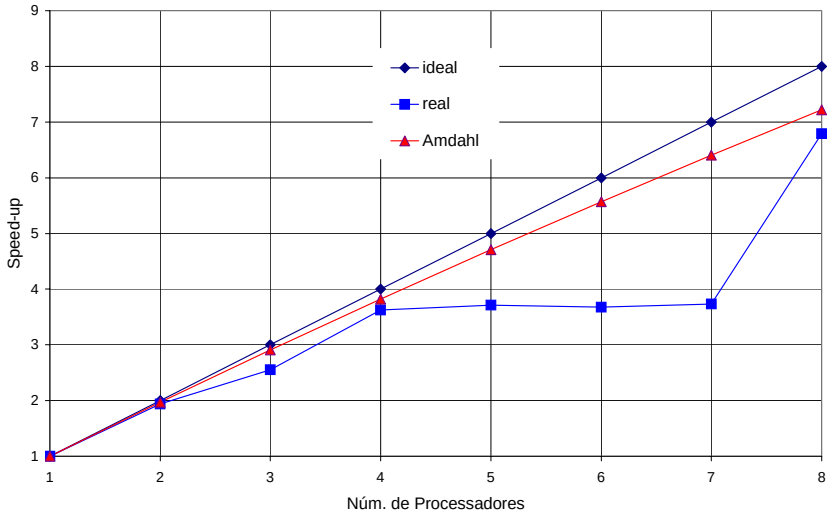
Análise	S_P10_4	S_P10_5	S_P10_6
Entrada de dados	0,36 seg	0,36 seg	0,55 seg
Equilíbrio inicial	19,53 seg	19,85 seg	23,38 seg
Análise estática acoplada	22,30 seg	21,78 seg	22,85 seg
Análise dinâmica acoplada	18.314,97 seg	7.598,22 seg	2.310,85 seg
Tempo total de processamento	18.357,17 seg	7.640,21 seg	2.357,66 seg

Tabela 6.10 - Fração Paralelizável Apontada pela Lei de Amdahl: P10 / sem SAVE.

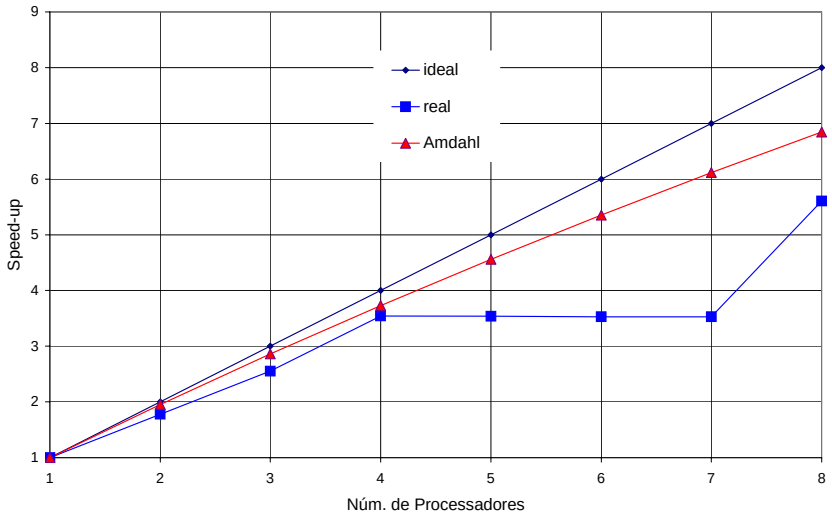
Análise	S_P10_4	S_P10_5	S_P10_6
Loop paralelizado	18.073,38 seg	7.456,27 seg	2.061,77 seg
Fração paralelizável	98,45 %	97,59 %	87,45 %

Analisando a Figura 6.9 e a Figura 6.10 pode-se observar que quando o módulo SAVE é ativado não é possível obter a mesma performance com todos os níveis de subciclagem casco-linhas. Isto ocorre pelo fato de haver um aumento no processamento pertencente a fração seqüencial em função do tempo de processamento consumido para gerar o arquivo para posterior reinício da análise. Conseqüentemente, a fração paralelizável diminui provocando a queda de performance comparado com o exemplo anterior.

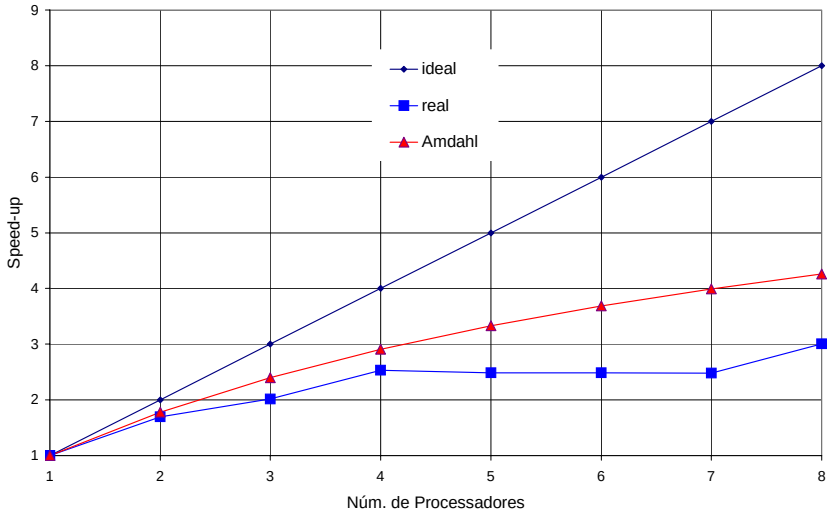
No entanto, verifica-se que a performance na simulação S_P10_4 apresenta resultado próximo da simulação S_P10_1. Vale lembrar que a simulação S_P10_4 foi realizado com o mesmo nível de subciclagem do que a simulação S_P10_1. Desta forma, verifica-se que o módulo SAVE exerce maior influência na performance a medida que o nível de subciclagem diminui, como se verifica na Figura 6.9 e na Figura 6.10 onde pode-se notar o decréscimo da performance. Este fato é compreendido, com maior facilidade, analisando-se a Tabela 6.10 onde se verifica uma queda na fração paralelizável a medida que NSUBDT diminui.



(a)

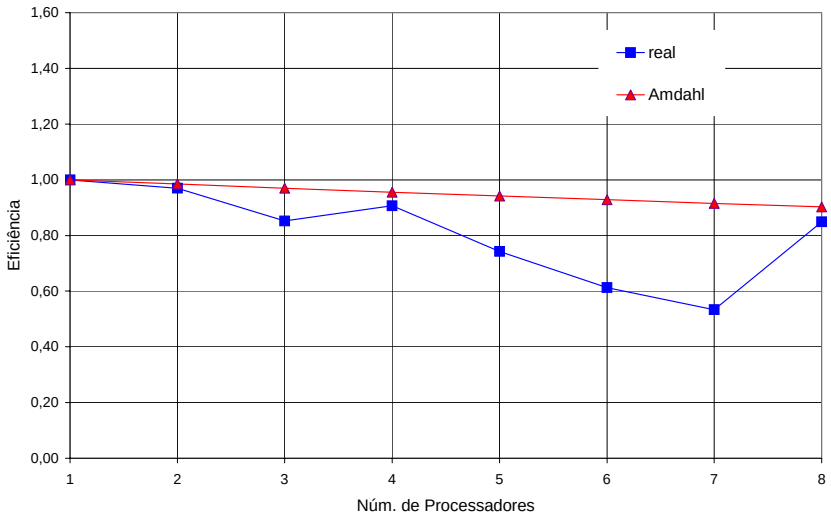


(b)

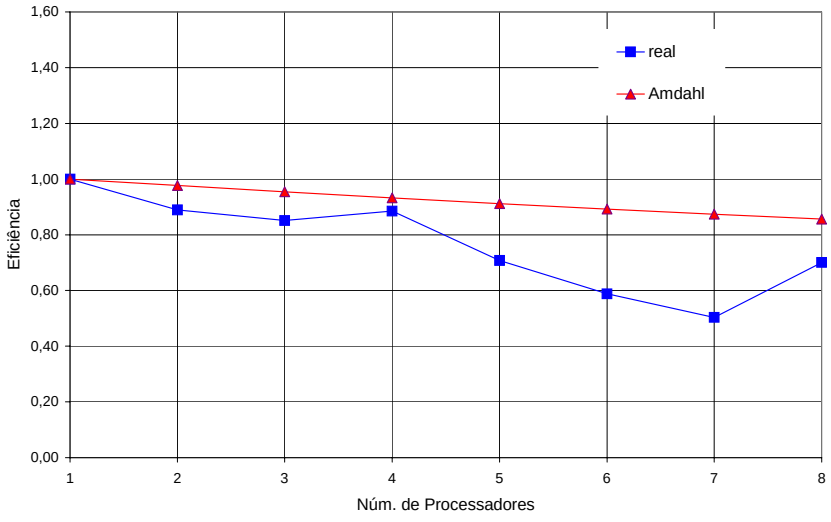


(c)

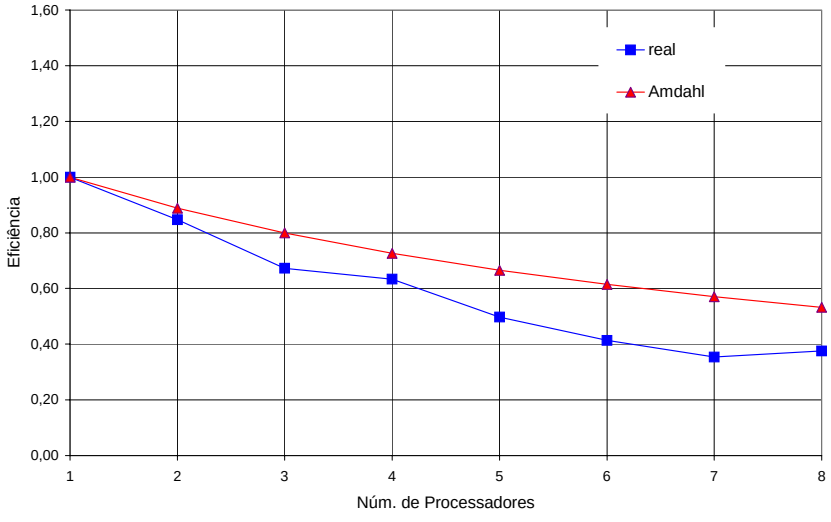
Figura 6.9 - Speed-up: (a) S_P10_4; (b) S_P10_5; (c) S_P10_6.



(a)



(b)



(c)

Figura 6.10 - Eficiência: (a) S_P10_4; (b) S_P10_5; (c) S_P10_6.

6.3 PLATAFORMA DE PRODUÇÃO P18

A plataforma P18 é ancorada por 8 linhas. Possui um total de 73 risers, de produção, exportação, injeção e umbilicais, sendo 72 risers flexíveis e um SCR (Steel Catenary Riser), todos com configurações em catenária livre.

6.3.1 Características do Casco

A seguir são apresentadas algumas características da plataforma P18:

- lâmina d'água: 910 m;
- azimute da plataforma: 203^{o};
- posição do centro de gravidade em relação ao eixo local: 19,30 m;
- Calado da plataforma: 23,10 m;
- peso da plataforma: 3.27E+05 kN.

Tabela 6.11 – Matriz de Raios de Giração.

	Valor
RXX	30.49
RYY	29.35
RZZ	32.64

Na Figura 6.11 é ilustrado o sistema de referência local (XYZ) e global (xyz) adotado. Empregaram-se coeficientes hidrodinâmicos gerados pelo programa WAMIT. O sistema de referência do WAMIT é o mesmo do Prosim. A Figura 6.12 mostra o modelo do casco da plataforma P18 utilizado na análise prévia do programa WAMIT para a obtenção das matrizes de amortecimento potencial e dos coeficientes de deriva.

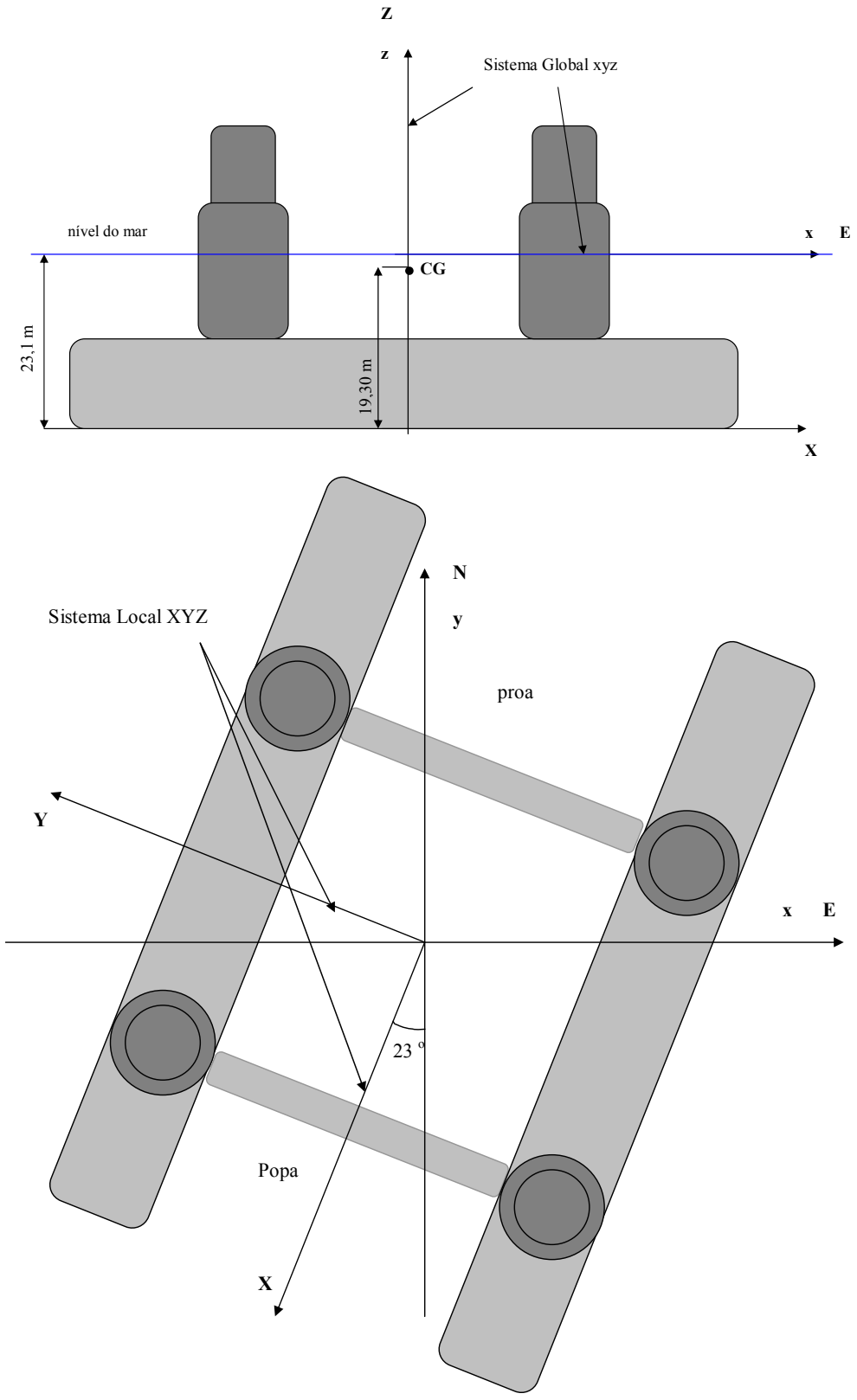


Figura 6.11 - Sistema de Referência da Plataforma P18.

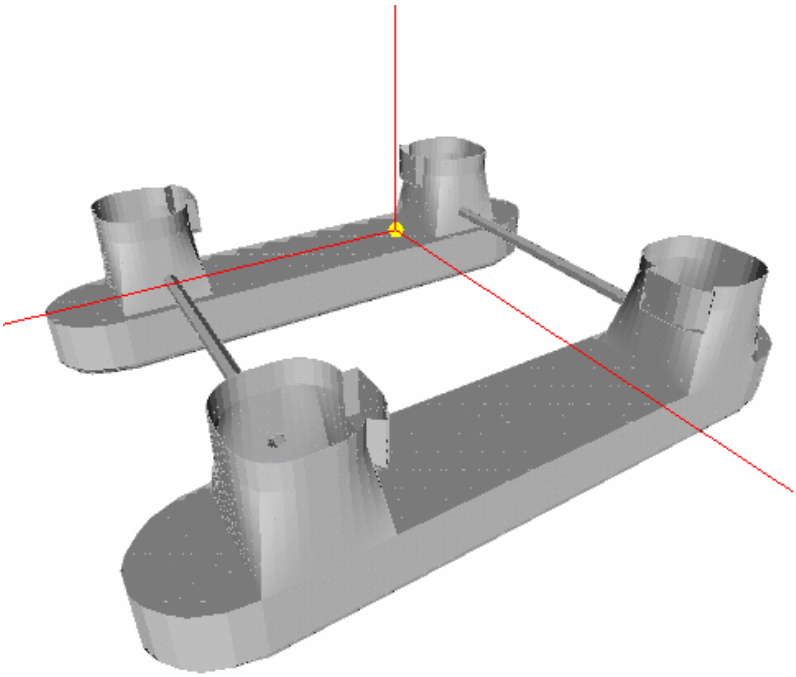


Figura 6.12 – Modelo do Casco da P18 Usado pelo Programa WAMIT.

6.3.2 Características das Linhas de Ancoragem

A P18 é ancorada por um total de 8 linhas de ancoragem compostas por um trecho de amarra no fundo e no topo e cabo de aço intermediário. As características das linhas de ancoragem são apresentadas na a seguir nas tabelas 6.12 e 6.13.

Tabela 6.12 - Propriedades das Linhas de Ancoragem da P18.

Linha	Comprimentos (m)			Projeção Horizontal (m)
	Amarra de Fundo	Cabo de Aço	Amarra de Topo	
Linha 1	1060	2377,735	368	2377,74
Linha 2	1290	2430,074	230	2430,07
Linha 3	1400	2597,106	354	2597,11
Linha 4	1230	2482,344	283.5	2482,34
Linha 5	1110	2513,828	439	2513,83
Linha 6	1480	2549,931	157	2549,93
Linha 7	1050	2418,237	610	2418,24
Linha 8	1735	2824,911	207	2824,91

Tabela 6.13 - Coordenadas de Conexão e Azimutes das Linhas de Ancoragem da P18.

Linha	Coordenadas Globais			Azimute (^{o})
	X	Y	Z	
Linha 1	-30,8	-37,4	-7,7	354.8
Linha 2	-25,5	-37,4	-7,7	325.5
Linha 3	25,5	-37,4	-7,7	268.1
Linha 4	30,8	-37,4	-7,7	227.3
Linha 5	30,8	37,4	-7,7	173.2
Linha 6	25,5	37,4	-7,7	131.8
Linha 7	-25,5	37,4	-7,7	74.8
Linha 8	-30,8	37,4	-7,7	39.8

6.3.3 Características dos Risers

A Tabela 6.14 apresenta as características dos risers, em termos de azimutes e os ângulos de topo. A P18 tem um total de 73 risers conectados, incluindo produtores, injetores e de exportação (dentre eles o SCR).

Tabela 6.14 – Risers Conectados à P18.

Número do Poço	Tipo de Riser		Azimute	Ângulo de Topo
1	Injetor	*4”*	7	3.81
	Umbilical	*UH*	7	3.81
2	Anular	*2.5”*	13	8.42
	Umbilical	*UH*	14	4.40
	Produtor	*4”*	12	8.75
3	Umbilical	*UH*	18	5.00
	Injetor	*4”*	17	4.40
4	Injetor	*4” ISU*	23	7.60
	Gasoduto	*8”*	28	8.00
5	Anular	*2.5”*	34	6.11
	Umbilical	*UH*	35	6.00
	Produtor	*4”*	36	4.44
6	Umbilical	*UH*	39	4.15
		4”	41	4.15
7	Anular	*2.5”*	85	4.85
	Umbilical	*UH*	86	4.82
	Produtor	*4”*	84	4.76
8	Umbilical	*UH*	82	4.80
	Injetor	*4”*	88	4.80
9	Anular	*2.5”*	104	7.90
	Umbilical	*UH*	109	7.30
	Produtor	*4”*	103	7.90
10	Anular	*2.5”*	110	4.06
	Umbilical	*UH*	111	4.14
	Produtor	*4”*	106	3.88
11	Umbilical	*UH*	114	4.20
	Injetor	*4”*	116	4.20
12	Anular	*2.5”*	119	5.00
	Umbilical	*UH*	121	5.00
	Produtor	*4”*	124	4.52
13	Anular	*2.5”*	126	5.98
	Umbilical	*UH*	128	5.40
	Produtor	*4”*	127	5.10
14	Anular	*2.5”*	179	5.91
	Umbilical	*UH*	180	7.50
	Produtor	*4”*	182	6.38
15	Anular	2.5”	198	7.00
	Umbilical	UH	199	7.00
	Produtor	4”	197	7.00
16	Anular	2.5”	201	7.24
	Umbilical	UH	200	6.00
	Produtor	4”	202	6.81
	SCR – Gasoduto	10”	192.11	21.67

Número do Poço	Tipo de Riser		Azimute	Ângulo de Topo
17	Anular	2.5”	211	4.33
	Umbilical	UH	212	4.09
	Produtor	4”	213	3.98
18	Anular	2.5”	214	2.38
	Umbilical	UH	215	5.50
	Produtor	4”	218	5.23
19	Umbilical	UH	220	4.40
	Injetor	4”	221	3.89
20	Anular	2.5”	280	7.00
	Umbilical	UH	284	7.00
	Produtor	6”	285	7.00
21	Anular	2.5”	289	4.74
	Umbilical	UH	290	4.75
	Produtor	4”	288	4.76
	Gasoduto	10”	299	6.26
	Oleoduto	12”S	296	6.96
	Oleoduto	12”N	290	8.28
22	Anular	2.5”	319	3.87
	Umbilical	UH	320	4.00
	Produtor	4”	311	9.14
23	Anular	2.5”	318	5.00
	Umbilical	UH	317	5.00
	Produtor	4”	319	5.00
24	Umbilical	UH	317	4.50
	Injetor	4”	316	4.05
25	Anular	2.5”	305	8.82
	Umbilical	UH	309	7.93
	Produtor	4”	312	9.02
26	Umbilical	UH	323	5.70
	Injetor	4”	321	6.00

6.3.4 Dados Ambientais

Para a análise em questão foram considerados os seguintes carregamentos ambientais:

- Carregamentos atuantes no casco: onda, correnteza e vento;

- Carregamentos atuantes nas linhas: onda, correnteza, gravitacional e amortecimento hidrodinâmico.

Foi considerado para este exemplo um espectro de onda irregular unidirecional com Hs = 1.75, Tp = 5.67 e direção vindo de NE.

Para o vento, considerou-se o espectro proposto pela API, com velocidade média de 13.709, ângulo de ataque em relação ao eixo X Global de 29.6^{o} e direção vindo de N.

O perfil de corrente utilizado na simulação com a P18 é o apresentado a seguir.

Tabela 6.15 – Perfil de Correnteza Atuante na Plataforma P18.

Profundidade (m)	Velocidade (m/s)	Direção
0	0.45	SW
50	0.46	SW
100	0.42	SWS
150	0.42	SWS
250	0.25	SWS
350	0.13	S
450	0.09	SE
550	0.12	E
650	0.18	NE
750	0.18	NE
900	0.23	NE
910	0	NE

6.3.5 Modelo Numérico

CASCO

O casco da plataforma é representado por um modelo reticulado de 36 elementos tubulares. As Figuras 6.13 e 6.14 apresentam, respectivamente, o esquema reticulado da plataforma P18 e a visualização sólida da plataforma P18 levando em conta os diâmetros dos membros tubulares equivalentes. Além disso, foram calculados os valores de coeficientes hidrodinâmicos CD e CM para cada trecho destes membros, baseando-se na Norma da DNV/POSMOOR , como descrito em [24].

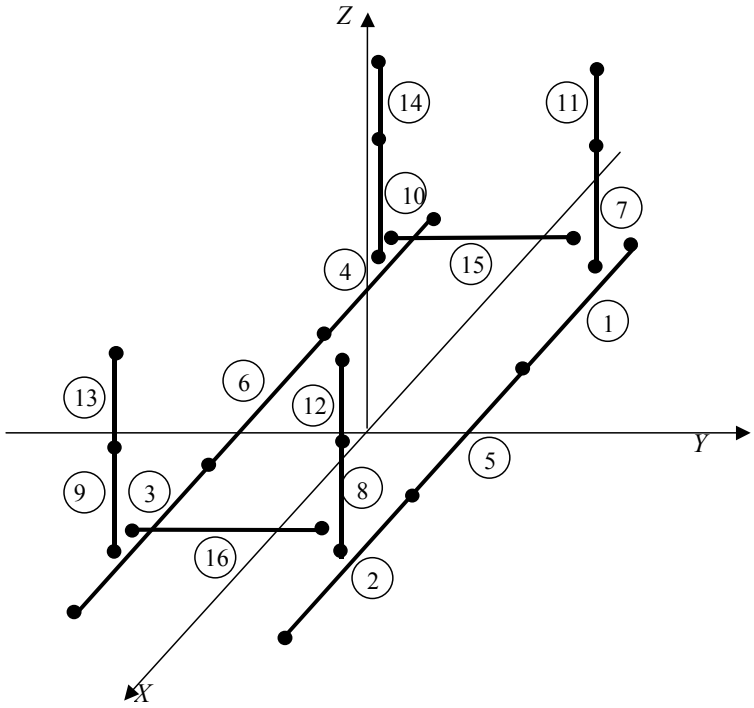


Figura 6.13 - Esquema Reticulado da Plataforma P18.

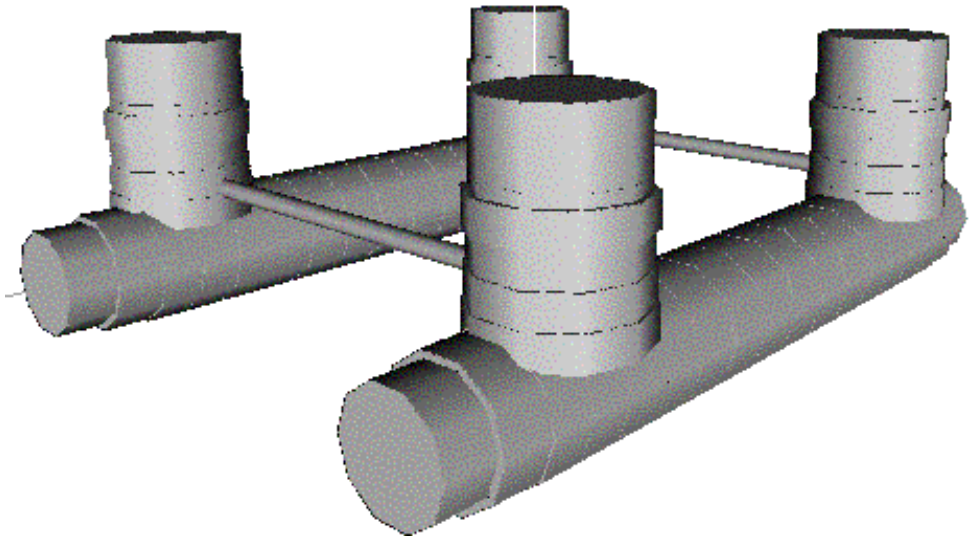


Figura 6.14 - Visualização Sólida do Modelo do Casco da P18.

CONJUNTO DE LINHAS DE ANCORAGEM E RISERS

As linhas e risers flexíveis foram discretizadas com elementos de treliça espacial e o SCR foi discretizado com elementos de pórticos espacial. As Tabelas 6.16, 6.17 e 6.18 apresentam, respectivamente, a malha típica de uma linha de ancoragem da P18, a malha típica de um riser flexível da P18 e a malha do SCR. A Tabela 6.19 apresenta o número total de elementos e número de equções para todas as linhas do modelo.

Tabela 6.16 - Malha Típica de Elementos Finitos de uma Linha de Ancoragem da P18.

Segmento	Comprimento do Segmento (m)	Comprimento Inicial (m)	Comprimento Final (m)
1 – Topo	368	20	20
2	1250	20	20
3 – Âncora	1060	20	20

Tabela 6.17 - Malha Típica de Elementos Finitos de um Riser Flexível Produtor da P18.

Segmento	Comprimento do Segmento (m)	Comprimento Inicial (m)	Comprimento Final (m)
1 – Topo	400	25	5
2	400	25	25
3	100	5	25
4	200	5	5
5 – Fundo	100	25	5

Tabela 6.18 - Malha Típica de Elementos Finitos do SCR da P18.

Segmento	Comprimento do Segmento (m)	Comprimento Inicial (m)	Comprimento Final (m)
1 – Topo	1,103	0,1575	0,1575
2	108,757	1	0,1575
3	100	20	1
4	926,304	20	20
5	100	5	20
6	130,786	5	5
7 – Fundo	700	20	5

Tabela 6.19 – Número de Elementos e Graus de Liberdade para Cada Linha.

Linha	Tipo	Tipo de Elemento	Número de Elementos	Número de Equações
1 - 8	Riser Flexível	Treliça	60	180
9	Riser Flexível	Treliça	80	240
10-42	Riser Flexível	Treliça	60	180
43	Riser Rígido	Pórtico	264	1584
44-51	Riser Flexível	Treliça	60	180
52-54	Riser Flexível	Treliça	72	219
55-57	Riser Flexível	Treliça	60	180
58	Riser Flexível	Treliça	69	207
59-60	Riser Flexível	Treliça	66	198
61	Riser Flexível	Treliça	62	189
62-73	Riser Flexível	Treliça	60	180
74	Ancoragem	Treliça	133	402
75	Ancoragem	Treliça	138	420
76	Ancoragem	Treliça	150	453
77	Ancoragem	Treliça	138	417
78	Ancoragem	Treliça	140	423
79	Ancoragem	Treliça	144	435
80	Ancoragem	Treliça	144	441
81	Ancoragem	Treliça	159	480

Nas Figuras 6.15, 6.16 e 6.17 são ilustrados respectivamente a configuração típica de uma linha de ancoragem, do SCR e de um riser flexível da plataforma P18. Em todas as figuras citadas tem-se a configuração discretizada.

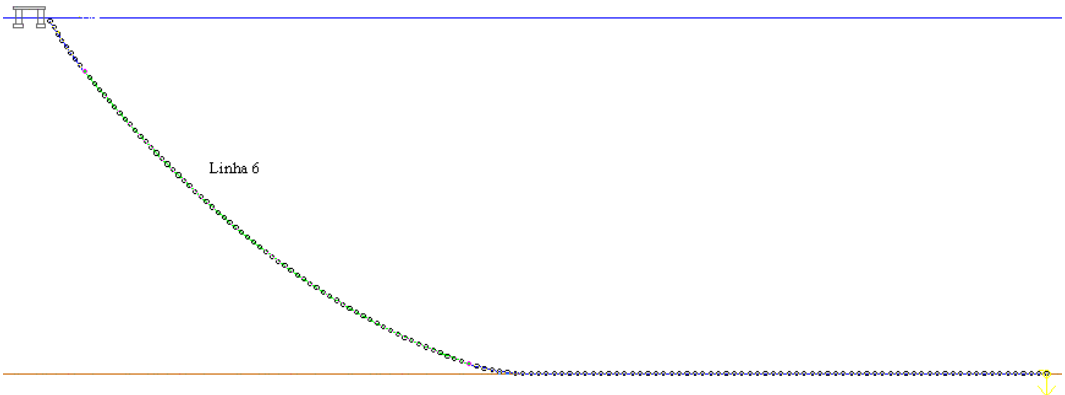


Figura 6.15 – Configuração Típica de uma Linha de Ancoragem da P18.

Figura 6.16 – Configuração do SCR da P18.

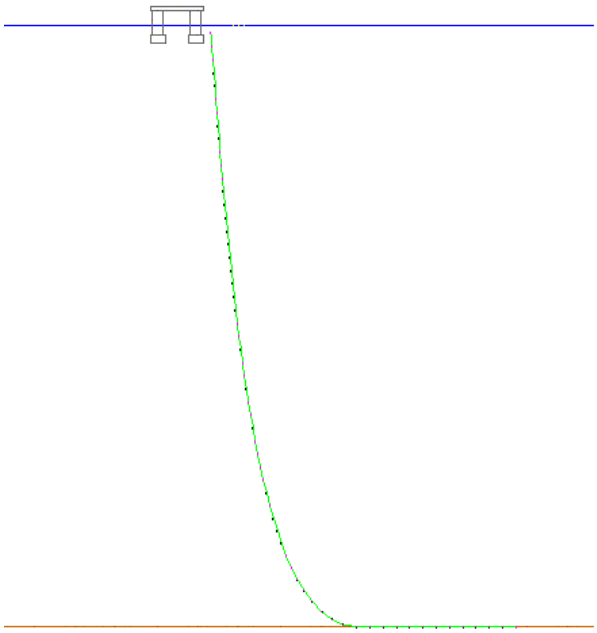


Figura 6.17 - Configuração Típica de um Riser de Produção da P18.

MODELO ACOPLADO

A seguir são apresentadas figuras ilustrando o modelo acoplado da P18, incorporado as linhas de ancoragem e aos risers.

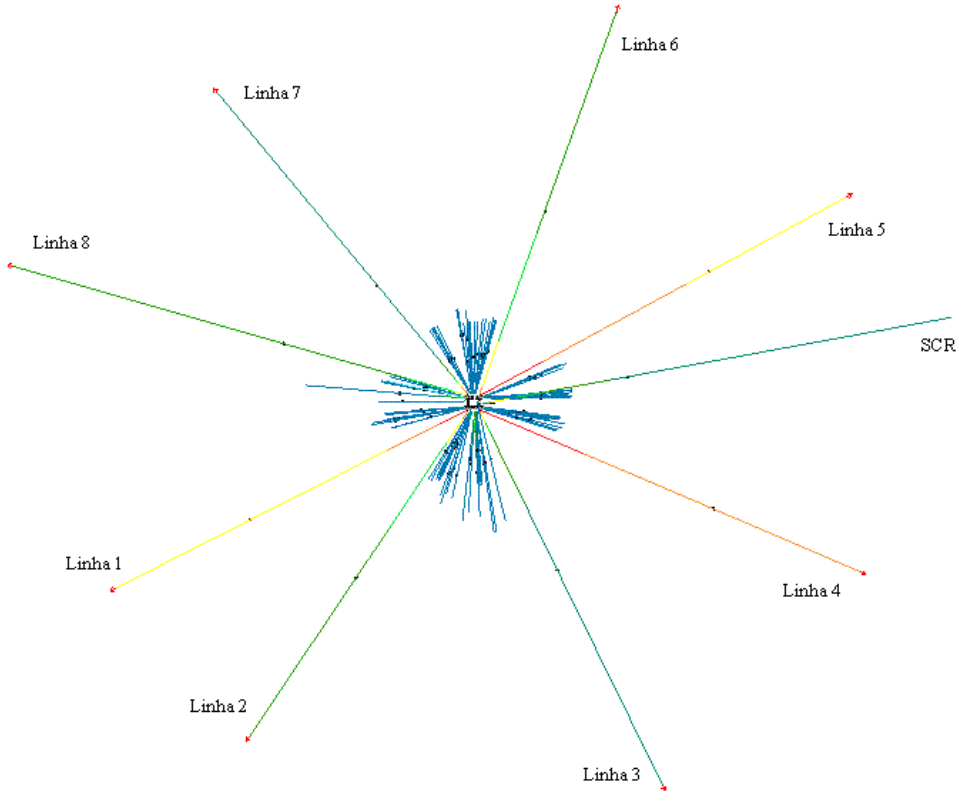


Figura 6.18 - Modelo Acoplado da Plataforma P18 – Vista XY.

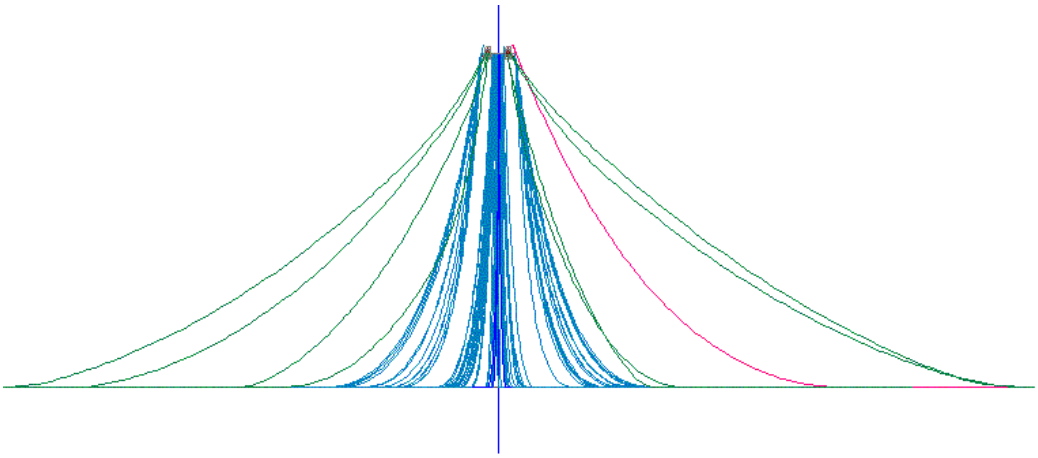


Figura 6.19 – Modelo Acoplado da Plataforma P18 - Vista Lateral.

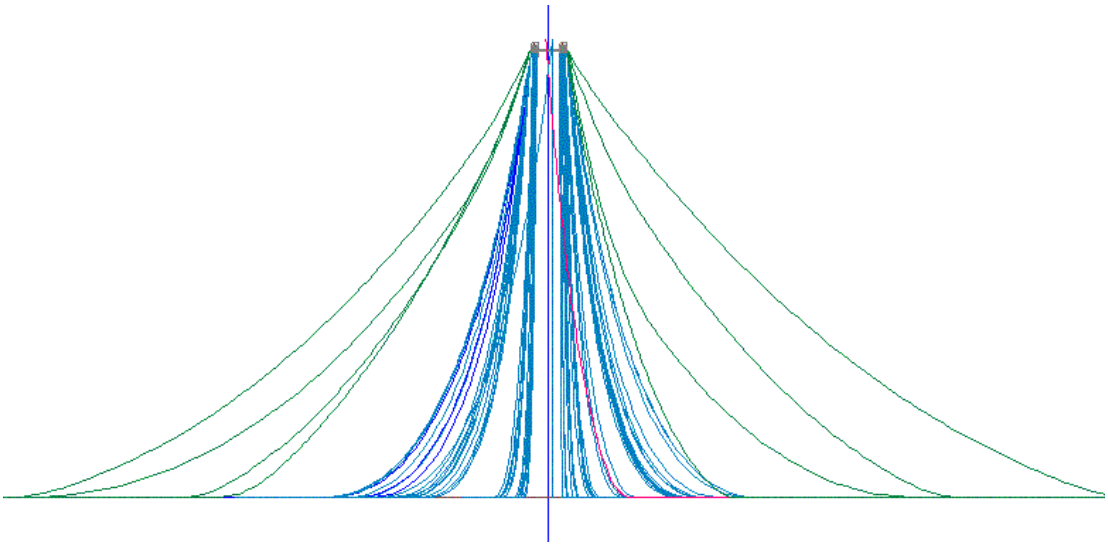


Figura 6.20 – Modelo Acoplado da Plataforma P18 - Vista Frontal.

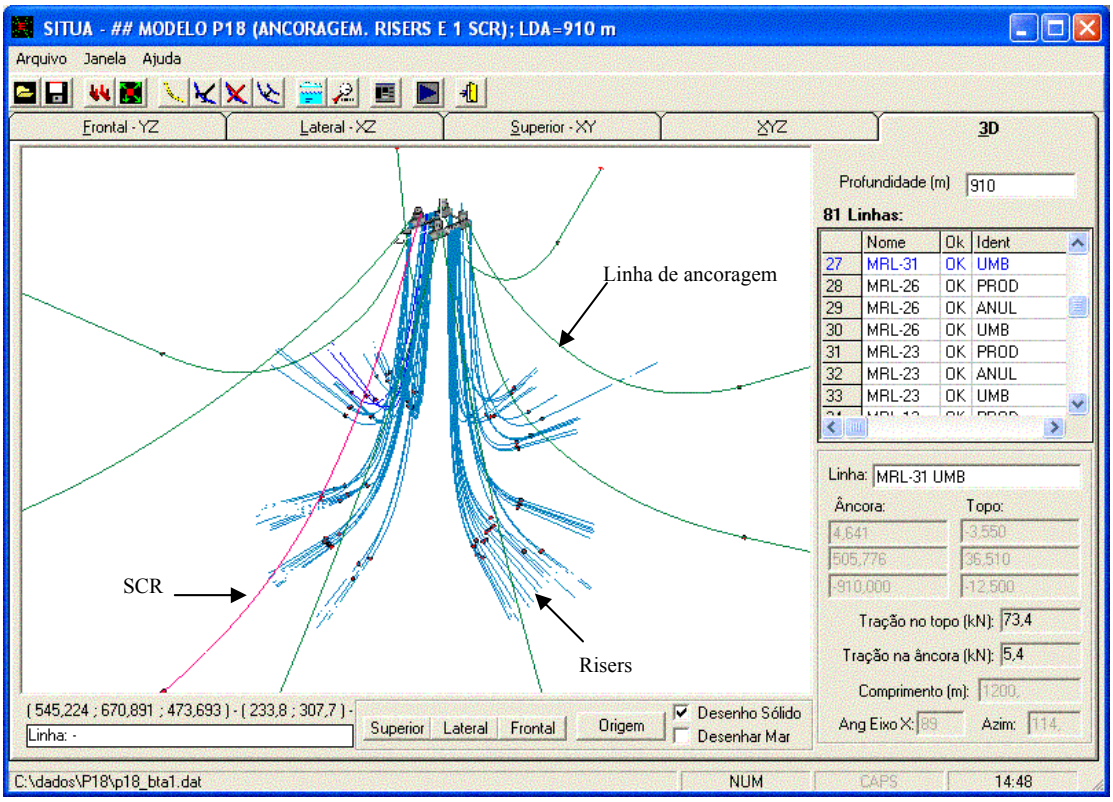


Figura 6.21 - Modelo Acoplado da Plataforma P18 - Vista 3D.

6.3.6 Resultados Típicos

Para ilustrar alguns dos resultados gerados através do programa Prosim será apresentada neste item os movimento de Surge, Sway, Heave, Yaw, Pitch e Roll resultante da análise realizada com a plataforma P18. Os parâmetros empregados nesta análise são os mesmo adotados na análise performance do código paralelizado.

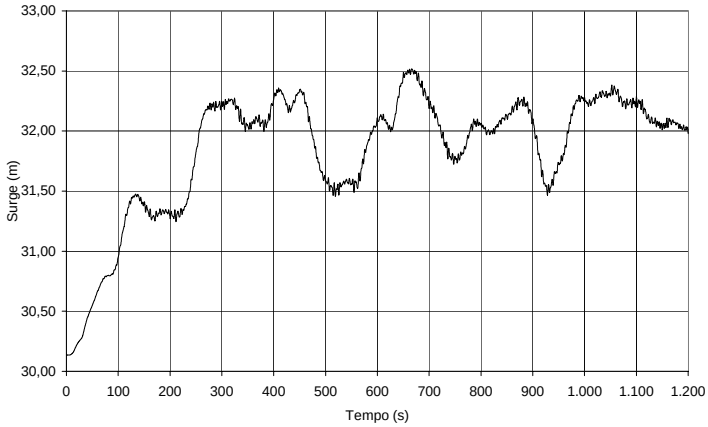


Figura 6.22 – Movimento de Surge da Plataforma P18.

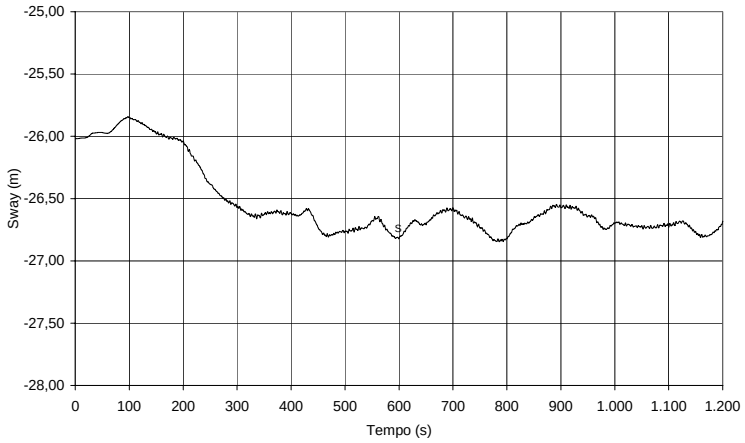


Figura 6.23 – Movimento de Sway da Plataforma P18.

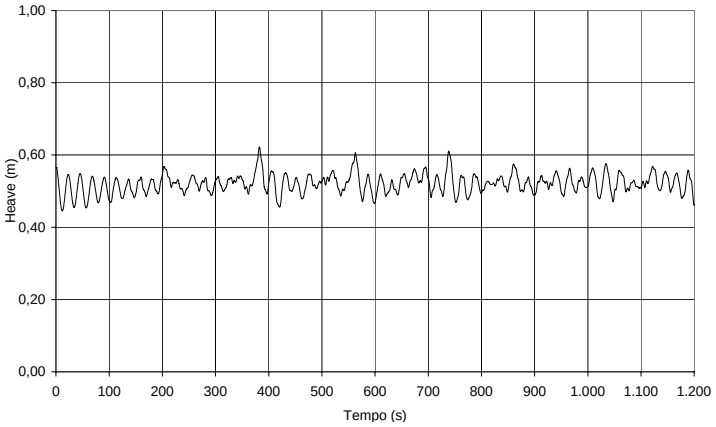


Figura 6.24 – Movimento de Heave da Plataforma P18.

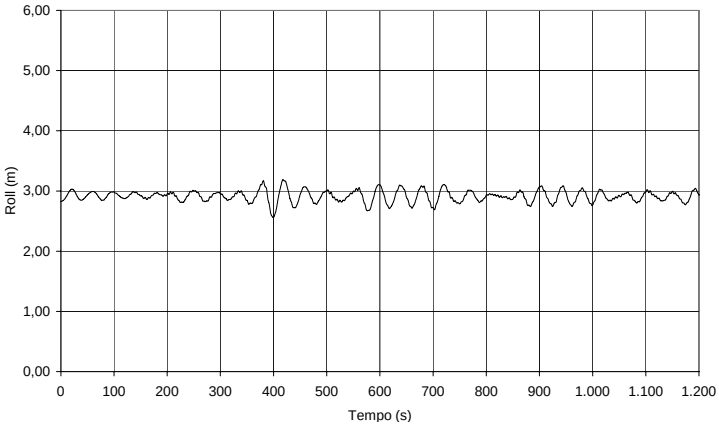


Figura 6.25 – Movimento de Roll da Plataforma P18.

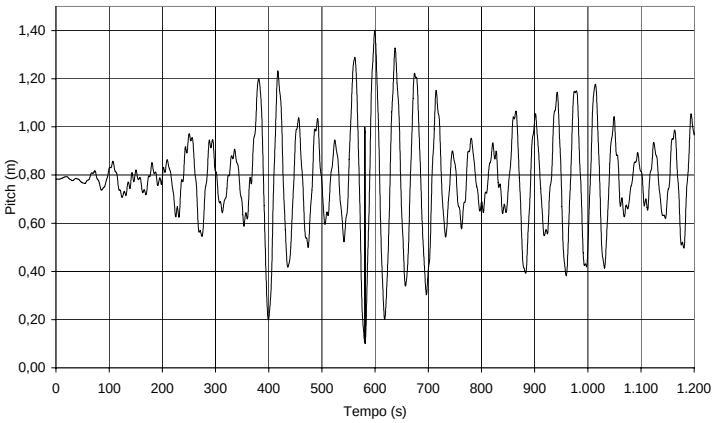


Figura 6.26 – Movimento de Pitch da Plataforma P18.

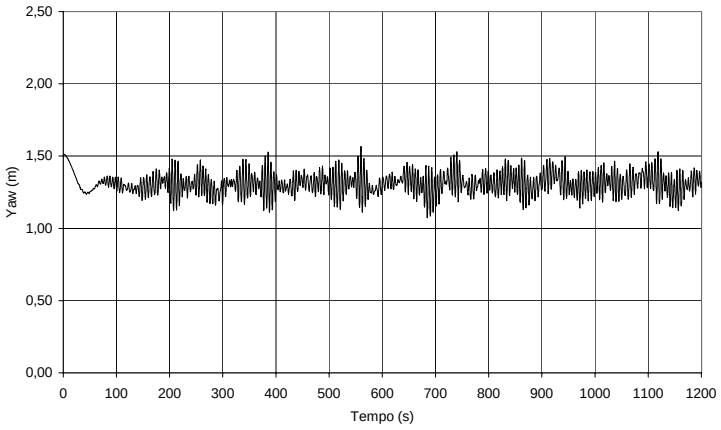


Figura 6.27 – Movimento de Yaw da Plataforma P18.

6.3.7 Análise de Performance

Nesta aplicação foram realizados dois conjuntos de simulações, sem ativar o módulo SAVE, com o intuito de comparar o ganho de performance em função do procedimento adotado para o balanço de carga computacional entre os processadores. Para cada conjunto de simulação foram adotados 3, 6 e 9 processadores para o processamento paralelo. Os mesmos parâmetros de análise dinâmica das simulações realizadas nesta aplicação são apresentados a seguir:

Tabela 6.20 - Parâmetros de Análise: P18 / sem SAVE.

Simulação	S_P18_1
Tempo total de análise	1.200,00 seg
Intervalo de integração do casco	0,20 seg
Intervalo de integração das linhas	0,02 seg
NSUBDT	10

Tabela 6.21 - Tempo de Processamento Seqüencial: P18 / sem SAVE.

Simulação	S_P18_1
Entrada de dados	1,60 seg
Equilíbrio inicial	1,90 seg
Análise estática acoplada	14,28 seg
Análise dinâmica acoplada	252.383,50 seg
Tempo total de processamento	252.400,28 seg

Tabela 6.22 - Fração Paralelizável Apontada pela Lei de Amdahl: P18 / sem SAVE.

Simulação	S_P18_1
Loop paralelizado	250.500,64 seg
Fração paralelizável	99,24 %

Na Figura 6.28 é apresentado o Speed-up da simulação com distribuição seqüencial das linhas com e sem o balanço de carga. Observa-se que o Speed-up da simulação sem o balanço de carga tem uma queda brusca de performance à medida que o número de processadores aumenta. Isto se deve ao fato de que alguns processadores ficam sobrecarregados em função do desequilíbrio de carga computacional. Por outro lado, a simulação com balanço de carga apresenta uma melhor performance comparada com a simulação sem balanço de carga.

Para entender melhor a perda de performance quando não se emprega o procedimento de balanço de carga, basta analisar a Figura 6.30, onde é apresentado o tempo de comunicação e de espera gasto por cada processador na segunda etapa de comunicação. Vale lembrar que nesta fase de comunicação é responsável por quase todo o overhead adicionado ao programa pelo fato do tempo ocioso de alguns processadores. Caso haja desequilíbrio de carga computacional alguns processadores ficam esperando pelo final do processamento dos processadores sobrecarregados.

As tomadas de medidas de tempo que compõem o resultado ilustrado na Figura 6.30 e também na Figura 6.31 foram feitas através do processamento paralelo com 9 processadores. Para realizar tais medidas levou-se em consideração o sincronismo entre os processadores no início e final do loop paralelizado, ou seja, todos os processadores iniciam e terminam ao mesmo tempo o loop. O resultado final da Figura 6.30 e 6.31 trata-se da somatória do tempo de espera e comunicação consumido ao longo de toda análise dinâmica.

Analisando o resultados da Figura 6.30 e da Figura 6.31 verifica-se que em ambas o tempo consumido na comunicação é extremamente pequeno comparado ao tempo de espera. Isto acontece em todos os processadores e em particular para o processador 9, de ambas as figuras, não há tempo de espera, pois verifica-se na Tabela 6.23 que a maior carga computacional está destinada ao processador 9, em função do número de equações.

Como foi mencionado anteriormente, o balanceamento de carga computacional considerando o número de equações de cada linha é uma aproximação, pois o custo computacional da solução do sistema de equações não é uma grandeza linear. No

entanto, quando a diferença entre o número de equações destinado a cada processador se aproxima de zero pode-se considerar esta aproximação aceitável.

Isto pode ser verificado na Figura 6.28 no processamento paralelo com 6 processadores, onde tem-se uma média de 3044 equações / processador. Utilizando o balanceamento de carga computacional tem os respectivos números total de equações 3120, 3060, 3024, 3042, 2942, 2949 e 3069 para os processadores 1, 2, 3, 4, 5 e 6. Como se observa o número total de equações de cada processador se aproxima da média de 3044 equações / processador. Em função deste balanceamento de carga tem-se um Speed-up de 5,61, considerado satisfatório comparado com o Speed-up de 5,78 resultante da lei de Amdahl ou mesmo com o Speed-up ideal de 6.

Tabela 6.23 - Número de Total de Equações por Processo.

	Distribuição de Tarefas Sem Balanceamento de Carga Computacional		Distribuição de Tarefas Com Balanceamento de Carga Computacional	
Processador	L. Ini. - L. Fin.	Núm. Total de Equações	L. Ini. - L. Fin.	Núm. Total de Equações
1	1-9	1680	1-11	2040
2	10-18	1620	12-22	1980
3	19-26	1620	23-33	1980
4	27-36	1620	34-42	1620
5	37-45	3034	43-45	1940
6	46-55	1737	46-56	2097
7	56-64	1692	57-67	2052
8	65-73	1620	68-75	1902
9	74-81	3651	76-81	2649

Na simulação com balanço de carga, a mesma performance com 6 processadores não foi alcançada com 9 processadores pelo fato de que em alguns processadores o número total de equações não se aproxima da média de 2029 equações / processador.

Como se verifica na Tabela 6.23 o processador 9 recebe 2649 equações resultante da somatória das equações da linha 76 a linha 81. Este valor equivale algo próximo a 30 % a mais da média de 2029 equações / processador e portanto a estimativa do aumento do overhead seria algo em torno de 30 %. O resultado do aumento de overhead é verificado no Speed-up com o processamento com 9 processadores ilustrado na Figura 6.28.

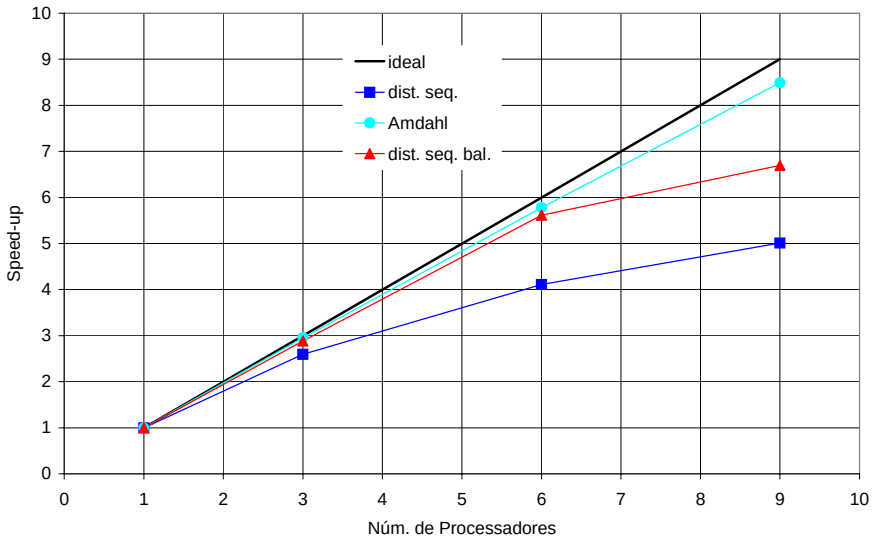


Figura 6.28 - Speed-up: S_P18_1.

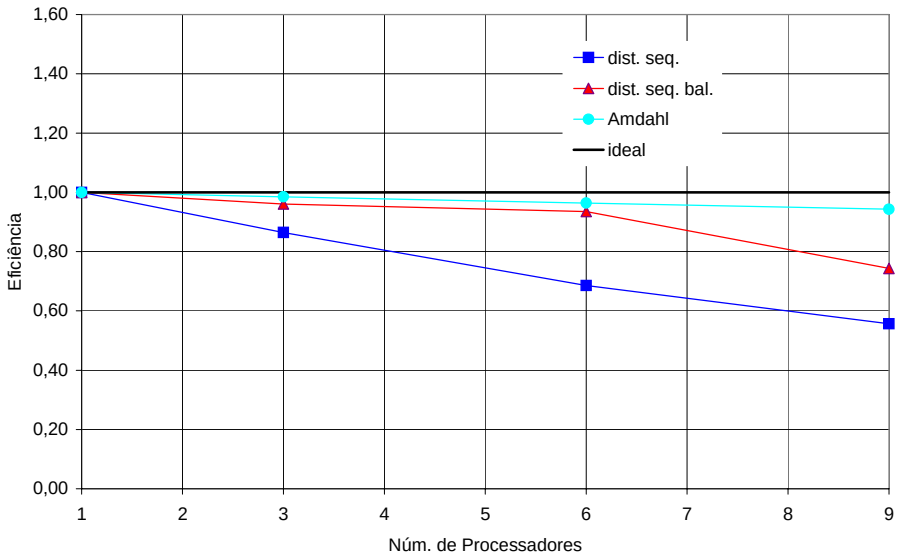


Figura 6.29 - Eficiência: S_P18_1.

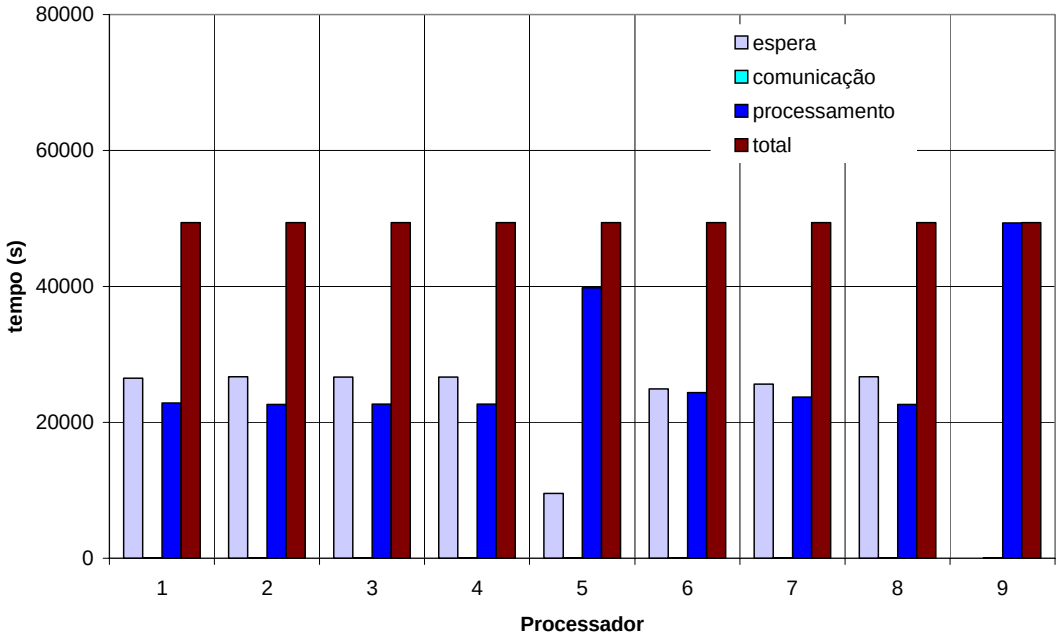


Figura 6.30 - Tempo de Comunicação e Espera na Fase II Sem Balanceamento de Carga Computacional.

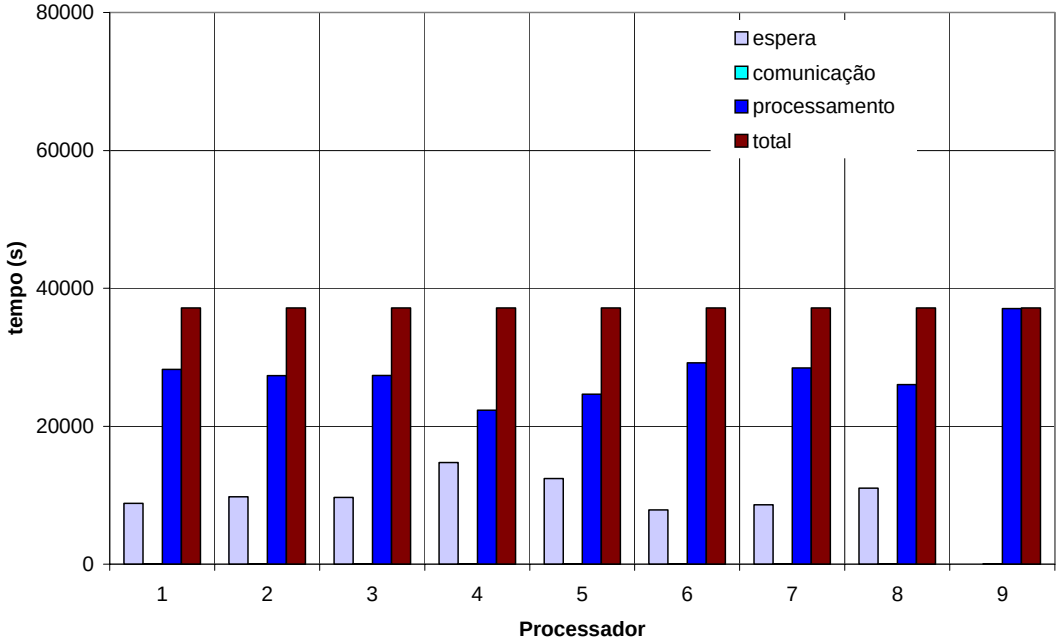


Figura 6.31 - Tempo de Comunicação e Espera na Fase II Com Balanceamento de Carga Computacional.

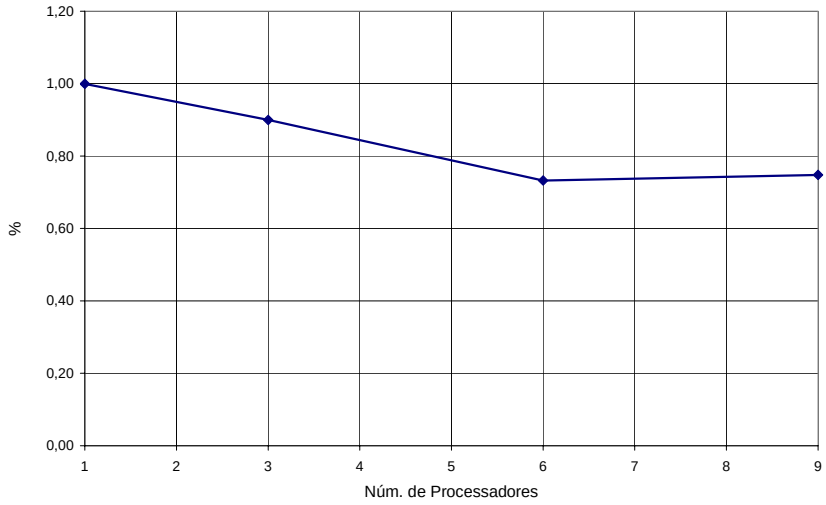


Figura 6.32 – Relação Entre o Speed-up da Distribuição Sem Balanceamento e Com Balanceamento de Carga Computacional.

A Figura 6.32 ilustra a relação entre o speed-up da distribuição sem balanceamento e o speed-up com balanceamento de carga computacional. Através desta figura pode-se notar a vantagem em realizar o balanceamneto mesmo com o procedimento simples. Mas pode-se notar que o sucesso do procedimento de balanceamento de carga adotado depende da configuração do conjunto de linhas que está sendo analisado e do número de processadores. Em alguns casos, como o processamento com 9 processadores, não é possível obter um balanceamento de carga mais preciso através do procedimento adotado. Pois a forma seqüencial como são distribuídas as linhas não permite contabilizar alternadamente as linhas o que daria mais alternativas para compor os conjuntos de linhas e certamente poderia realizar-se um balanceamento de carga mais próximo do ideal.

7 CONCLUSÕES

7.1 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo a implementação em ambiente paralelo de memória distribuída de um programa orientado à análise de unidades flutuantes ancoradas. Para alcançar tal objetivo, tomou-se como ponto de partida o código seqüencial do programa Prosim, desenvolvido de forma cooperativa por pesquisadores do LAMCSO (Laboratório de Método Computacionais e Sistemas Offshore) do Programa de Engenharia Civil e do CENPES/Petrobras. Para a implementação em paralelo foram utilizadas bibliotecas de comunicação MPI.

Estudos do código seqüencial mostraram que a maior parte do tempo de processamento é consumida na análise dinâmica acoplada. Mais especificamente no cálculo da força que atua na unidade flutuante, resultante da análise dinâmica dos modelos de elementos finitos que representam o sistema de ancoragem e do conjunto de risers. Além disto, nesta etapa há uma predisposição para o paralelismo natural, visto que a análise de cada linha de ancoragem ou riser é feita de forma independente uma da outra. Desta forma, o paralelismo foi realizado na análise dinâmica das linhas, dividindo-se as linhas entre os processadores.

Na divisão das linhas foram propostas duas formas de distribuição, em ambas as distribuições foram feitas de forma seqüencial. Primeiramente, propôs-se uma distribuição sem balanceamento de carga computacional, onde cada processador recebe um conjunto de linhas igual ao quociente inteiro da divisão entre o total de linhas e a quantidade de processadores. Caso a divisão não seja exata, as linhas restantes são incorporadas a cada processo até a distribuição da totalidade das linhas.

O segundo procedimento de distribuição proposto leva em consideração o balanceamento da carga computacional em função do número de equações atribuídas a cada processador, e considera também que a rede de processadores seja homogênea. O

algoritmo criado para realizar esta distribuição tenta distribuir entre os processadores conjuntos de linhas que possuam um número de equações próximo da média (resultado inteiro da divisão do número total de equações pelo número de processadores). No processo de montagem dos conjuntos de linhas soma-se o número de equações de cada linha cada vez que se adiciona uma linha ao conjunto. Caso esta soma ultrapasse a média, a última linha adicionada pode ser a última linha do presente conjunto, ou pode passar a ser a primeira linha do próximo conjunto (a decisão é tomada de modo a levar a conjuntos com número de equações que menos se afastem da média).

Analisando os resultados de performance obtidos pela primeira aplicação, verifica-se que a implementação do procedimento de subciclagem casco-linhas, realizada em versões mais recentes do programa Prosim, proporcionou uma grande contribuição para o processamento paralelo. Em função desta subciclagem foi possível diminuir a comunicação coletiva de redução entre os processos, realizada logo após a composição do vetor de forças atuante no CG.

É conveniente lembrar que o tempo total consumido na comunicação de redução realizada na FASE II (descrita do Capitulo 5) não é o maior causador do *overhead* (ou tempo de processamento extra adicionado com a paralelização) gerado nesta fase de comunicação. O principal fator gerador do *overhead* é o tempo ocioso em alguns processadores devido ao desequilíbrio de carga computacional, como foi observado no caso das diferentes linhas da plataforma analisada na segunda aplicação.

Com isso, e observando os resultados de performance obtidos nesta aplicação, pode-se concluir que o balanceamento de carga computacional é extremamente importante e necessário para que se tenha um ganho de processamento satisfatório. O tempo ocioso em cada instante de tempo da análise dinâmica pode até ser pequeno, mas o overhead total será grande, visto que, o tempo de espera é acumulado em todos os instantes de integração das equações de movimento da unidade flutuante, e portanto, ao final da análise ter-se-á valores expressivos que podem comprometer a performance do programa.

Este fato, pode ser naturalmente observado no ganho de performance, ilustrado na aplicação com a plataforma P18, quando se emprega o balanceamento de carga

computacional para a distribuição das linhas. Apesar do procedimento de balanceamento de carga proposta neste trabalho ser relativamente simples, compondo uma aproximação linear, pode-se considerá-lo satisfatório para um número pequeno de processadores com relação ao número de linhas. Neste caso, mesmo se as linhas apresentarem grandes diferenças com relação a suas respectivas malhas de elementos finitos, pode-se obter um balanceamento de carga próximo do ideal. Isto pode ser observado na Figura 6.28 no processamento paralelo com 3 e 6 processadores.

7.2 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

As aplicações estudadas neste trabalho empregaram um cluster com um número relativamente pequeno de processadores. Seria conveniente investigar o comportamento das implementações efetuadas em máquinas com mais processadores. Neste caso, para garantir que a implementação paralela apresente um melhor ganho de performance em um processamento paralelo envolvendo um número considerável de computadores, deve-se prosseguir investindo no balanceamento de carga computacional. Para isto segue abaixo algumas sugestões:

- Balanceamento de carga computacional em função do tempo de processamento consumido por cada linha. Este procedimento pode ser realizado medindo-se o tempo de processamento de cada linha durante a análise estática que precede a análise dinâmica, onde não há ainda o paralelismo. Caso a distribuição seja seqüencial basta utilizar o mesmo algoritmo proposto para o balanceamento de carga em função do número de equações;

- A distribuição seqüencial pode ser substituída pela distribuição cíclica, onde os conjuntos de linhas são formados com numeração alternada, não adjacente. Este procedimento permite realizar uma distribuição mais balanceada por oferecer maior possibilidades de combinação entre as linhas, no entanto é necessário o uso de algoritmos mais inteligente, como por exemplo um algoritmo genético;

- Mesmo com as melhores técnicas consideradas para melhorar o balanceamento de carga computacional, sempre haverá em alguns casos em

que o balanceamento não será totalmente satisfatório. Isto ocorrerá quando por exemplo o número de processadores se aproximar do número de linhas, enquanto algumas linhas tiverem malhas muito distintas e portanto consumirem tempo de processamento muito superior a outras linhas. Neste caso tem-se um desequilíbrio de carga computacional localizado, ou seja, não será possível, com a técnica de distribuição apresentada, distribuir parte da linha que está sobrecarregando um determinado processador para outro processador com uma carga computacional menor. Na verdade, em aplicações práticas esta situação pode ser comum, já que as plataformas de produção de petróleo recentemente consideradas pela Petrobras têm um número de linhas que se aproxima de uma centena, o que se aproxima do número de nós que pode ser montado em um cluster de alto desempenho. É possível também imaginar situações onde apenas uma linha seja modelada por elementos finitos; por exemplo, no caso da própria P18 poderia-se estar interessado em estudar mais detalhadamente apenas o comportamento do SCR, e as demais linhas poderiam ser representadas por modelos escalares baseados em curvas de restauração. Estes problemas podem ser contornados com a implementação de técnicas de paralelismo baseadas em métodos de decomposição do domínio, agindo diretamente no algoritmo de solução do problema dinâmico de cada linha; com isso, diferentes trechos de uma mesma linha seriam distribuídos em diferentes processadores. Acredita-se que este é o avanço natural e o próximo passo nas atividades de paralelização de programas de análise de unidades flutuantes ancoradas, e já está sendo considerado em outros trabalhos presentemente em andamento, associados ao uso de algoritmos implícitos para a solução da dinâmica das linhas, proposta em [23].

REFERÊNCIAS BIBLIOGRÁFICAS

[1] AMERICAN PETROLEUM INSTITUTE (API), *Recommended Practice for Planning, Design and Construction Fixed Offshore Platforms, Working Stress Design*, 20th edition, 1991.

[2] BATHE, K. J.M., *Finite Elements Procedures*, Prentice-Hall, New Jersey, 1996.

[3] CENTRO NACIONAL DE PROCESSAMENTO DE ALTO DESEMPENHO-SP. *Introdução ao MPI*. Disponível em: http://www.cenapad.unicamp.br . Acesso em: 12 dez. 2003.

[4] CLOUGH, R. W., PENZIEN, J., *Dynamics of Structures*, McGraw-Hill, New York, 1975.

[5] CHAKRABARTI, S. K, *Hydrodynamics of Offshore Structures*, Computational Mechanics Publications Southampton Boston, 1987.

[6] COOK, R. D., MALKUS, D. S. PLESHA, M., E., *Concepts and Applications of Finite Element Analysis*, Third Edition, Jonh Wiley & Sons, 1989.

[7] CORREIA, F.N., *Aplicação de Metodologia Híbridas em Estudos Paramétricos sob o Comportamento de Sistemas Offshore*, Tese de M. Sc. , COPPE/UFRJ, Programa de Engenharia Civil, Março de 2003.

[8] COSTA, M., C., A., *Data Mining em Computadores de alto Desempenho Utilizando-se Redes Neurais*, Tese de D. Sc. , COPPE/UFRJ, Programa de Engenharia Civil, 1999.

[9] DIETZ, H., *Linux Parallel Processing HOWTO*. 1998. Disponível em: http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Parallel-Processing-HOWTO.pdf . Acesso em: 10 abr. 2003.

[10] FREITAS, E. L. GEYER, C., *Uma comparação entre os modelos de Message Passing MPI e PVM*. UFRGS. Março de 2003. Disponível em : www.inf.ufrgs.br/procpar/disc/cmp134/trabs/T2/981/mpi.html. Acesso em: 17 de set. 2003.

[11] FOSTER, I., *Designing and Bulding Parallel Program*. 1995. Disponível em : http://www-unix.mcs.anl.gov/dbpp/text/book.html . Acesso em: 8 abr. 2003.

[12] GROPP, W., LUSK, E., *The MPI Communication Library: Its Design And A Portable Implementation*. Mathematics and Computer Science Division, Argone National Laboratory. Disponível em: http://www-fp.mcs.anl.gov/~lusk/papers/mississippi/paper.html. Acesso em: 9 abr. 2003.

[13] IBM: INTENATIONAL TECHNICAL SUPPORT ORGANIZATION, *RS/6000: Pratical MPI Programming*, Agosto 1999. Disponível em: http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg245380.html?Open. Acesso em: 16 set. 2003.

[14] JACOB, B.P, *Notas de Aula de Dinâmica de Sistemas Discretos*, COPPE/UFRJ, Programa de Engenharia Civil, Rio de Janeiro, 2001.

[15] JACOB, B.P, *Notas de Aula de Análise Dinâmica Não-Linear de Estruturas*, COPPE/UFRJ, Programa de Engenharia Civil, Rio de Janeiro, 2001.

[16] JACOB, B.P., *Estratégias Computacionais Para Análise Não-Linear Dinâmica de Estruturas Complacentes para Águas Profundas*, Tese de D. Sc. , COPPE/UFRJ, Programa de Engenharia Civil, Dezembro de 1990.

[17] JACOB, B.P, *Programa PROSIM: Simulação Numérica do Comportamento de Unidades Flutuantes Ancoradas, Versão 2.2ª – Manual de Entrada de Dados*, COPPE/UFRJ, Programa de Engenharia Civil, Rio de Janeiro, 2001.

[18] JACOB, B.P, LIMA, B., S., L., P., EBECKEN, N., F., F., BENJAMIN, A. C., "Portable Fortran Programming Tools in The Development of a Structural Analysis Program", *Elsevier Science Ltd*, vol.57, No. 6, pp. 1109-1125, 1995.

[19] MACDONAL, N., MINTY, E., HARDING, T., BROWN, S., *Writing Message-Passing Parallel Programs with MPI: A two-day course*. Disponível em: http://www.hku.hk/cc/sp2/training.html. Acesso em: 13 jan. 2003.

[20] MESSAGE PASSING INTERFACE FORUM. *MPI: A Message-Passing Interface Standard*. 1995. Disponível em : http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html. Acesso em: 15 jun. 2003.

[21] PACHECO, P. S.,MING, W. C., *Introduction to Message Passing Programming: MPI User Guide in FORTRAN*. 1997. Disponível em: http://www.hku.hk/cc/sp2/training.html. Acesso em: 13 jan. 2003.

[22] PAULLING, J. R., TDSIM 6 - Theory and User Guide, 1992.

[23] RODRIGUES, M.V., *Desenvolvimento de Ferramentas Numéricas para a Análise Acoplada de Linhas e Unidades Flutuantes*, Exame de Qualificação ao D.Sc. , COPPE/UFRJ, Programa de Engenharia Civil, Março de 2003.

[24] SENRA, S. F., *Metodologias de Análise e Projeto Integrado de Sistemas Flutuantes para exploração de Petróleo Offshore*. Tese de D.Sc., COPPE/UFRJ, Programa de Engenharia Civil, Rio de Janeiro, 2004.

[25] WHELLER, J. D., *Method for Calculating Forces Produced for Wave-Body Interactions*, Massachusetts Institute of Tecnology Department of Ocean Engeneering-MIT, 1991.

APÊNDICE A

SUB-ROTINAS DE MPI UTILIZADAS

MPI_INIT(IERROR)

Parâmetros:

IERROR - código de erro, retorna 0 em caso de sucesso ou código de erro em caso de falha na comunicação.

Descrição: inicializa o MPI e todo programa de MPI deve conter este comando. Nenhum comando MPI pode ser chamado antes de **MPI_INIT**, com exceção de **MPI_INITIALIZED**.

MPI_COMM_RANK(MPI_COMM_WORLD, RANK, IERROR)

Parâmetros:

MPI_COMM_WORLD - comunicador.

RANK - intero que especifica o rank da tarefa em um grupo de comunicação.

Descrição: retorna o rank do local da tarefa em um grupo de comunicação.

MPI_COMM_SIZE(MPI_COMM_WORLD, SIZE, IERROR)

Parâmetros:

SIZE - inteiro que especifica o número de tarefas ou processos de um grupo de comunicação.

Descrição: retorna o número de tarefas ou processos associado à um comunicador, por exemplo **MPI_COMM_WORLD**.

MPI_FINALIZE(IERROR)

Descrição: finalize MPI, é a última chamada a ser feita, depois de MPI_FINALIZE nenhuma outra chamada de MPI pode ser feita.

MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, MPI_COMM_WORLD, IERROR)

Parâmetros :

BUF - endereço inicial do buffer emissor.

COUNT - número de elementos do buffer emissor.

DATATYPE – tipo de dado do buffer emissor.

DEST - rank de destino.

TAG - número de identificação da mensagem.

Descrição: esta rotina é empregada na comunicação ponto a ponto classificada como modo padrão de comunicação *Standart* e usa comunicação *blocking*.

MPI_RECV(BUF, COUNT, DATATYPE, SOURCE, TAG, MPI_COMM_WORLD, STATUS(MPI_STATUS_SIZE), IERROR)

Parâmetros:

BUF - endereço inicial do buffer de recepitor.

COUNT - número de elementos recebidos.

SOURCE - rank do emissor

STATUS - é um array com três parâmetros de informação sobre a mensagem enviada.

Descrição: assim com MPI_SEND MPI_RECV também é classificado como modo padrão *blocking* . Ao receber a mensgem MPI_RECV armazena no buffer de recebimento contido para o elemento do tipo especificado através de **DATATYPE**, começando a BUF de endereço. A mensagem recebida

deve ser menor ou igual ao comprimento do buffer especificado em **COUNT** do buffer recptor. Se todas as mensagens de recepção não se ajustarem sem mutilação, um erro de transbordamento acontece. Se uma mensagem recebida for menor que a especificação do buffer de recepção, então só os locais, correspondendo à mensagem atual são mudados.

MPI_BCAST(BUFFER, COUNT, DATATYPE, ROOT, MPI_COMM_WORLD, IERROR)

Parâmetros:

BUFFER - endereço do buffer emissor

ROOT - processo raiz.

Descrição: envia mensagem do processo MASTER à todos os processos do grupo de comunicação.

MPI_ALLREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, MPI_COMM_WORLD, IERROR)

Parâmetros:

SENDBUF - endereço inicial do buffer emissor.

RECVBUF - endereço inicial do buffer receptor.

OP - tipo de operação (MPI_PROD,MPI_BAND,MPI_SUM).

Descrição: trata-se de uma função de operação de redução total, é aplicada sobre os dados localizados no buffer de envio **SENDBUF** de cada processo do grupo, o resultado desta função é difundido de volta para todos os processos no buffer de recebimento **RECVBUF** através da operação definida em **OP**.

MPI_BARRIER(MPI_COMM_WORLD, IERROR)

Descrição: sincroniza todos os processos de um comunicador.

APÊNDICE B

EXECUÇÃO DO PROSIM EM AMBIENTE PARALELO

A sintaxe de execução do PROSIM com qualquer outro programa em paralelo é:

```
mpirun –np (número de processos) prosim
```

Ao executar este comando o programa PROSIM procura o arquivo INPROSM.DAT no mesmo local onde foi executado o comando mpirun. Neste arquivo existem os argumentos descritos abaixo e que são necessários para o iniciar o programa. Caso não haja este arquivo a entrada dos mesmos deverá ser feita via monitor.

<prefin> : Prefixo do arquivo principal de entrada de dados, que deve ter a extensão .DAT, e deve estar localizado no disco e diretório indicado em <path>.

<prefout> : Prefixo dos arquivos de saída, que serão gravados em um sub-diretório \SAIDA do diretório indicado em <path>.

<prefcof> : Prefixo de um arquivo auxiliar de entrada de dados, contendo coeficientes hidrodinâmicos. O arquivo deve ter a extensão .WNF, e deve estar localizado no disco e diretório indicado em <path>. Se este parâmetro for omitido, o programa toma para o prefixo do arquivo .WNF o mesmo nome fornecido para <prefin>.

<prefres> : Prefixo de um arquivo binário que contém todos os dados e resultados de uma análise anterior, efetuada com a opção SAVE ativada. O arquivo deve ter a extensão .SAV, e deve estar localizado no disco e diretório indicado em <path>.

<path> : Disco e diretório onde estão contidos todos os arquivos de entrada (por exemplo, "/home/usuario/P10/").

ARQUIVOS DE ENTRADA

`<prefin>.DAT` : Arquivo principal de entrada de dados, contendo os registros descritos no corpo do Manual do PROSIM. O arquivo deve estar localizado no disco e diretório indicado em `<path>`.

`<prefin>.COF` ou

`<prefcof>.COF` : Arquivo auxiliar de entrada de dados, contendo coeficientes hidrodinâmicos. O arquivo deve estar localizado no disco e diretório indicado em `<path>`.

A leitura deste arquivo é opcional, sendo ativada através das opções FRQF e DRIFT. Este arquivo é gerado por um programa de difração tal como o WAMIT; sua estruturação está descrita no Apêndice B do manual do PROSIM.

`<prefres>.SAV` : Arquivo binário que contém todos os dados e resultados de uma análise anterior, efetuada com a opção SAVE ativada. O arquivo deve estar localizado no subdiretório `\SAIDA` do diretório indicado em `<path>`.

ARQUIVOS DE SAÍDA

Como resultado de uma execução do programa PROSIM, serão gerados diversos arquivos com resultados, todos localizados no subdiretório onde a linha de comando mpirun é executada, já na versão Windows os arquivos de saídas ficam localizados no sub-diretório `\SAIDA` do diretório indicado em <path>. Os arquivos são os seguintes:

Arquivos - texto

`<prefout>.ECO` – Arquivo contendo um espelho ou “eco” do arquivo de entrada de dados.

`<prefout>.LIS` – Arquivo contendo o relatório dos dados de entrada, e um resumo dos resultados estatísticos dos sinais de resposta no tempo dos

movimentos do CG da unidade flutuante e dos esforços nas linhas.

<prefout>.DBG – Arquivo contendo o relatório da alocação de ponteiros, números de interações por análise e interações para a convergência de cada análise .

Arquivo binário para armazenamento da análise e posterior reinicio

<prefout>.SAV – Arquivo contendo todos os dados e resultados da análise, para permitir um posterior reinicio. A geração deste arquivo é determinada através da opção SAVE. O arquivo é gravado ao final da análise, ao ser alcançado o tempo total TOTALT. Opcionalmente pode ser gravado também ao longo da análise, em intervalos definidos pela variável TIMSAVE.

Arquivos para pós-processamento gráfico

<prefout>.DEF – Arquivo contendo os dados da geometria e configurações deformadas das linhas discretizadas por elementos finitos (coordenadas dos nós, incidência dos elementos, e deslocamentos). O arquivo tem um formato apropriado para leitura pelo programa pós-processador gráfico POSNL do sistema ANFLEX, permitindo a visualização e animação de configurações deformadas das linhas.

<prefout>.SNP – Praticamente igual ao arquivo <prefout>.DEF mas tem um formato apropriado para leitura pelo programa pós-processador gráfico do PROSIM (SITUA).

<prefout>.RES – Arquivo com conteúdo semelhante ao anterior, mas com um formato apropriado para leitura pelo programa pós-processador gráfico POSNL do sistema ANFLEX

<prefout>.MAP – Arquivo contendo esforços em linhas discretizadas por elementos finitos. O arquivo tem um formato apropriado para leitura pela interface gráfica do PROSIM, permitindo a visualização de mapas de cores de esforços durante a animação de configurações deformadas das linhas

No prefixo <prefout> dos arquivos .DEF,.SNP,.RES e .MAP há a adição dos atributos _s e _d_0 para os arquivos gerados respectivamente na análise estática e dinâmica. Desta forma, os prefixos de tais arquivos são dados por <prefout>_s na análise estática e <prefout>_d_0 na análise dinâmica, sendo que _0 indica o número de vezes que a análise foi reinicializada (0, 1, 2,...).

Com a implementação paralela os prefixos destes arquivos ganharam mais um atributo _p<np>, sendo np o rank do processo (0, 1, número de processos - 1). Resultando em <prefout>_p<np>_s e <prefout>_p<np>_d_0 para a análise estática e para análise dinâmica. Por exemplo, o processo 1 gera o arquivo prefout_p1_d_0.LIS. Vale ressaltar que para o processo MASTER (rank 0) este atributo não é adicionado mantendo o prefixo do original.

Os arquivos descritos a seguir são gerados pelo PROSIM em um formato apropriado para leitura pelo programa pós-processador GRAPHER. Este programa oferece uma grande flexibilidade para a visualização e montagem de gráficos, permitindo alterar e redefinir eixos, escala, legendas, títulos, combinar curvas, redefinir tipos de linha, cores, tamanhos e tipos de caracteres, dentre outros recursos.

<prefout>.WAV – Arquivo contendo os pontos que definem o espectro de onda gerado pelo PROSIM.

<prefout>.WND – Arquivo contendo os pontos que definem o espectro de vento gerado pelo PROSIM.

Arquivos para pós-processamento gráfico e estatístico

Os seguintes arquivos contêm pontos que definem sinais de resposta no tempo, ainda em um formato apropriado para leitura pelo programa pós-processador GRAPHER. Estes arquivos podem também ser lidos pelo programa pós-processador estatístico POSSINAL, que calcula parâmetros estatísticos, verifica a estacionaridade do sinal, gera espectros, dentre outros recursos.

`<prefout>.PLD` – Arquivo contendo os sinais da elevação da onda e do vento aleatório, bem como os sinais de resposta do movimento nos seis graus de liberdade do CG da unidade flutuante (surge, heave, sway, roll, yaw e pitch) e do movimento relativo ("air-gap") nos pontos selecionados através de registros (ver manual do PROSIM).

`<prefout>.PLP` – Arquivo contendo os sinais de resposta do movimento absoluto (em termos das componentes x, y e z) nos pontos selecionados.

`<prefout>.PLT` – Arquivo contendo os sinais de resposta dos esforços nas linhas de ancoragem representadas por molas lineares ou não-lineares do modelo desacoplado (tendões, linhas "taut-leg", linhas em catenária, linhas definidas por tabelas força *x* deslocamento).

Os arquivos .PLD, .PLP e .PLT também são gerados na análise estática e na análise dinâmica, empregando o mesmo método de diferenciação dos arquivos demonstrados anteriormente (_s para análise estática e _d_0 para análise dinâmica).

`<prefout>.GR2` – Arquivo contendo os sinais de resposta de esforços nas linhas discretizadas por elementos de pórtico do modelo acoplado. Os elementos e as componentes de esforços são os selecionados.

`<prefout>.GR3` – Arquivo contendo os sinais de resposta de esforços nas linhas discretizadas por elementos de cabo do modelo acoplado.

<prefout>.GR4 – Arquivo contendo os sinais de resposta de esforços nas linhas discretizadas por elementos de treliça do modelo acoplado.

<prefout>.GR5 – Arquivo contendo os sinais de resposta de esforços em elementos escalares generalizados em linhas discretizadas por elementos finitos.

<prefout>.G2T – Arquivo contendo os sinais de resposta de tensões nas linhas discretizadas por elementos de pórtico do modelo acoplado. São gravadas componentes de tensão nos nós dos elementos para os quais foram solicitadas curvas de esforços.

<prefout>.GRD – Arquivo contendo os sinais de resposta de deslocamentos em nós das linhas discretizadas por elementos finitos do modelo acoplado.

<prefout>.GRV – Arquivo contendo os sinais de resposta de velocidades em nós das linhas discretizadas por elementos finitos do modelo acoplado. Os nós e os graus de liberdade são os mesmos selecionados para os deslocamentos. Observa-se que, para análises com acoplamento estático, aproximações para as velocidades são determinadas apenas quando for solicitada a consideração de amortecimento hidrodinâmico nas linhas.

<prefout>.GRC – Arquivo contendo sinais de resultados intermediários, empregados pela equipe de desenvolvimento do programa PROSIM para depuração e/ou estudos paramétricos para ajudar na compreensão da resposta dinâmica de um dado problema. Por exemplo, algumas colunas deste arquivo contêm a história no tempo das resultantes das cargas ambientais sobre as linhas; resultantes das forças de vento e onda sobre o casco; ou o ângulo

que uma linha de ancoragem originalmente vertical faz com a direção vertical durante a resposta.

<prefout>.GTD – Arquivo contendo os sinais de resposta de variação do touch-down point das linhas discretizadas por elementos finitos.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo