Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Engenharia Mecânica

Pós-graduação em Engenharia Mecânica

Desenvolvimento de um Simulador para Problemas Multi-físicas

Eduardo Roberto Rodrigues de Brito Junior

Dissertação de Mestrado

Recife
29 de junho de 2007

Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Engenharia Mecânica

Eduardo Roberto Rodrigues de Brito Junior

Desenvolvimento de um Simulador para Problemas Multi-físicas

Trabalho apresentado ao Programa de Pós-graduação em Engenharia Mecânica do Departamento de Engenharia Mecânica da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Engenharia Mecânica.

Orientador: *José Maria Andrade Barbosa*
Co-orientador: *Félix Christian Guimarães Santos*

Recife
29 de junho de 2007

B862d **Brito Junior, Eduardo Roberto Rodrigues de**

Desenvolvimento de um simulador para problemas multi-físicas / Eduardo Roberto Rodrigues de Brito Junior. – Recife: O Autor, 2007.

xiii, 93 f.; il., gráfs., tabs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Depto. de Engenharia Mecânica, 2007.

Inclui referências bibliográficas e apêndices.

1. Engenharia Mecânica. 2. Método do elemento finito. 3. Fenômenos acoplados. 4. Multi-físicas. 5. Simulador. I. Título.

621 CDD (22.ed.) **UFPE/BCTG/2007-112**

"DESENVOLVIMENTO DE UM SIMULADOR PARA PROBLEMAS MULTIFÍSICA".

EDUARDO ROBERTO RODRIGUES DE BRITO JUNIOR

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA MECÂNICA

ÁREA DE CONCENTRAÇÃO: MECÂNICA COMPUTACIONAL
APROVADA EM SUA FORMA FINAL PELO
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA/CTG/EEP/UFPE

Prof. Dr. JOSÉ MARIA ANDRADE BARBOSA
ORIENTADOR/PRESIDENTE

Prof. Dr. FÉLIX CHRISTIAN GUIMARÃES SANTOS
CO-ORIENTADOR

Profa. Dra. ANA ROSA MENDES PRIMO
COORDENADORA DO CURSO

BANCA EXAMINADORA:

Prof. Dr. JOSÉ MARIA ANDRADE BARBOSA (UFPE)

Prof. Dr. FÉLIX CHRISTIAN GUIMARÃES SANTOS (UFPE)

Prof. Dr. RAMIRO BRITO WILLMERSDORF (UFPE)

Prof. Dr. FERNANDO PEREIRA DUDA (UFRJ)

Dedico esta dissertação de mestrado aos meus pais, Eduardo Roberto Rodrigues de Brito e Sandra Suely Rodrigues de Brito, que sempre apoiaram todas as minhas decisões e sei que sempre estarão ao meu lado. Dedico também a minha namorada Edyla, com quem sempre poderei contar.

Agradecimentos

Iniciamente agradeço a Deus, que me protege e me dá força todos os dias, sem Ele nada disso seria possível.

Agradeço aos meus pais, Eduardo e Sandra Brito, pela minha educação, pelo apoio, enfim por tudo que vocês representam para mim, vocês são um exemplo que pretendo seguir.

Ao meu orientador, Professor José Maria Andrade Barbosa, com quem tenho a oportunidade de trabalhar desde o quarto período da graduação, onde tive o primeiro contato com uma linguagem de programação e com simulação computacional.

Ao meu co-orientador, Professor Félix Christian Guimarães Santos, pela sua orientação, pela sua ajuda e principalmente pelas cobranças, as quais foram fundamentais para este trabalho.

Ao Professor José Maria Bezerra pela ajuda nas implementações e que vem realizando uma trabalho fantástico no desenvolvimento do MPhyScas.

A Professora Márcia Mahon Campello de Souza com quem trabalhei como monitor na disciplina Sinais e Sistemas, disciplina do curso de Engenharia Eletrônica na Universidade Federal de Pernambuco.

A minha namorada, Edyla Cavalcanti Dourado, uma pessoa maravilhosa que entrou na minha vida. Agradeço pelo seu apoio, compreenção e paciência neste período de muito trabalho.

A toda a minha família, em especial meu irmão Leandro, minhas cunhadas Tábata e Bruna e meu concunhado Chico pelos momentos de descontração.

A todos os meus amigos, em especial aos amigos do Centro Federal de Educação Tecnológica de Pernambuco; aos colegas da graduação da Universidade Federal de Pernambuco; aos amigos da pós-graduação da Universidade Federal de Pernambuco; aos amigos do Laboratório de Robótica: Fred, Guaraci e Henrique; aos amigos do Labcom: Felipe, Manassés, Rodrigo, Michel, João, Hélder.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio finaceiro desde o período em que fui aluno de Iniciação Científica.

Por fim, agradeço a todas as pessoas que direta e indiretamente estão ligadas a este trabalho e que não foram citadas aqui pois certamente não coseguiria lembrar de todas.

"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original." (Albert Einstein)

Resumo

Simulação de fenômenos naturais acoplados, tais como evolução do dano em materiais devido carregamentos mecânicos e químicos estão se tornando uma prática importante na engenharia. Isto foi influenciado pelo enorme crescimento da indústria de softwares para a Mecânica Computacional. Apesar de todo esse avanço, técnicas de engenharia de software que remetem a década de setenta tem sido empregadas. O projeto e a implementação de simuladores para fenômenos multi-físicas acoplados apresentam uma grande quantidade de problemas, tornando-se impraticável no sentido computacional se ferramentas poderosas não estão disponíveis para auxiliar o analista. O Método do Elemento Finito é uma forma de obter aproximações numéricas de uma teoria matemática a qual descreve comportamentos físicos. Este método é considerado uma ferramenta computacional poderosa para a solução de equações diferenciais e integrais que surgem em vários campos da engenharia. Este trabalho tem como ojetivo apresentar a implementação de um simulador para fenômenos acoplados utilizando a linguagem de padrões. Estes padrões compõem um sistema de abstrações, os quais são capazes de representar os fenômenos e as interações entre eles e as interações entre os fenômenos e os algoritmos de solução empregado pelo simulador. As definições desses padrões bem como suas utilizações são apresentadas dentro de um contexto computacional desenvolvido para a produção automática de simuladores baseados no método do elemento finito. Isto inclui um esclarecimento sobre o simulador e suas camadas: Kernel, Block, Group e Phenomenon. Para a validação deste simulador, foram implementados dois modelos e a utilização de soluções manufaturadas. Este trabalho faz parte do projeto Danoplexus que visa analisar a degradação em dutos devido a corrosão e a fragilização por hidrogênio. Este projeto faz parte da Rede de Pesquisa Cooperativa em Modelagem Computacional (RPCMod).

Palavras-chave: Método do elemento finito, Simulador, Multi-físicas, Fenômenos acoplados

Abstract

Simulation of coupled natural phenomena, such as the evolution of material damage due to mechanical and chemical loads, are becoming more and more important in the engineering pratice. This was influenced by the growth of Computational Mechanics software industry. Despite the computational advance, it still applies software engiheering development techniques related to the seventies. The project and the implementation of simulators for coupled multi-physics presents a myriad of problems in such a way that they become impratical in a computational sense if powerful tools are not available to support the analyst. The Finite Element Method is a way to approximate a mathematical theory which describes physical behaviours. This method has been considered a powerful computational tool to solve differential and integral equations that arise in many fields of engineering. This work is aimed to present the implementation of a simulator for coupled phenomena through a language of patterns. Those patterns compose a system of abstractions, which are capable of representing the phenomena and their interaction with each other an with the solution algorithms employed by the simulator. The definitions of those patterns and their usage are presented within the a computational framework built for the automatic production of simulators based on the finite element method. This includes explanation about the simulator ans its layers: Kernel, Block, Group e Phenomenon. Two models were implemented to validate the simulator. This work is related to Danolexus project which analyses pipeline degradation due to corrosion and hydrogen embrittlement. This project is related to Network for Cooperative Research on Computational Modeling (RPCMod).

Keywords: Finite element method, Simulator, Multi-physics, Coupled phenomena

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

A Mecânica Computacional, nas últimas três décadas, tem provocado um impacto profundo na ciência e na tecnologia. A indústria de software para a Mecânica Computacional gera vários bilhões de dolares por ano [1]. O sucesso da Mecânica Computacional se deve ao fato de sua efetividade na solução de problemas de interesse da sociedade, bem como ter proporcionado um profundo conhecimento de fenômenos naturais (fatos que acontecem na natureza, tais como movimento de pontos materiais e a transferência de calor em um contínuo).

Apesar de todo esse desenvolvimento computacional técnicas de desenvolvimento de engenharia de software que remetem a década de setenta continuam sendo aplicadas. A conseqüência disso é que existem poucas ferramentas confiáveis e efetivas que facilitam desenvolvimento de simuladores e novos desenvolvimentos em métodos numéricos tornam-se difíceis de serem introduzidos nos simuladores existentes.

O enorme sucesso da Mecânica Computacional reside no seu poder de predição, tornando possível a simulação de eventos naturais complexos bem como o uso dessas simulações no desenvolvimento de sistemas de engenharia. Isto é feito através da modelagem computacional: o desenvolvimento de versões discretizadas de teorias mecânicas, as quais são acessíveis a cálculos digitais, juntamente com os processos complexos de manipulação desta representação digital para fornecer uma idéia real de como esses sistemas se comportam.

O Método do Elemento Finito (MEF) tem sido freqüentemente utilizado na Mecânica Computacional. O MEF é uma forma de se obter uma aproximação numérica de uma teoria matemática que descreve um comportamento físico. Este método é considerado uma técnica computacional para solução de equações diferenciais e integrais que surgem em vários campos da engenharia.

1.1 Objetivos

O objetivo principal deste trabalho consiste em implementar o ambiente de simulação proposto em [2]. A arquitetura proposta tem como principais requisitos o aumento da reusabilidade, da adaptabilidade e da manutenabilidade de simuladores multi-escala e multi-física além de possibilitar o trabalho cooperativo e o gerenciamento de conhecimento. Este ambiente considera as técnicas existentes e as facilidades de melhoramento das características de software através do uso de métodos científicos. O MEF é profundamente polimórfico significando que simuladores altamente versáteis baseados neste método podem ser produzidos.

Este trabalho faz parte do projeto Danoplexus que visa analisar a degradação em dutos devido à corrosão e a fragilização pelo hidrogênio. Este projeto faz parte da Rede de Pesquisa

Cooperativa em Modelagem Computacional (RPCMod). A RPCMod reúne 56 pesquisadores de Alagoas (UFAL), Ceará (UFCE), Paraíba (UFCG), Pernambuco (UFPE e UPE), Rio Grande do Norte (UFRN) e Sergipe (UFS), que desenvolvem projetos cooperativos na área de simulação computacional voltados para a indústria do petróleo e gás natural. Além de oferecer à sociedade um potencial instalado de grupos de pesquisa e laboratórios para a execução de pesquisa e desenvolvimento nas áreas relacionadas com a simulação computacional, a rede também tem como objetivo formar recursos humanos nas áreas correlatas à modelagem computacional, particularmente relacionada a aplicações na indústria do petróleo e do gás [3].

1.2 Organização do Trabalho

Esta dissertação divide-se em seis capítulos. Neste primeiro capítulo foi apresentada uma visão geral do que se trata esta dissertação e quais os seus principais objetivos.

No **Capítulo** 2 - **Revisão Bibliográfica** - uma breve descrição do Método do Elemento Finito é apresentada. Neste capítulo também é apresentado alguns trabalhos propondo modelos para análise do acoplamento termomecânico e análise de efeitos ambientais em sólidos, bem como algumas filosofias para construção de simuladores.

No **Capítulo** 3 - **Desenvolvimento do Simulador** - são apresentadas as dificuldades encontradas quando se trabalha com fenômenos multi-física e multi-escala. A implementação de um ambiente de simulação com a finalidade de contornar estas dificuldades é realizada.

No **Capítulo** 4 - **Modelos e Implementações** - são apresentados os modelos escolhidos para serem implementados no simulador, bem como os procedimentos para discretização desses modelos. As implementações destes modelos no simulador também são apresentadas neste capítulo.

No **Capítulo** 5 - **Soluções Numéricas** - são apresentadas as soluções numéricas obtidas nas simulações e também é realizada uma análise dos resultados obtidos nestas simulações.

Por fim, no **Capítulo** 6 - **Conclusões e Trabalhos Futuros** - são feitas conclusões e considerações finais sobre a utilização do ambiente de simulação implementado. Neste capítulo também são apresentados trabalhos que estão sendo realizados bem como as propostas para trabalhos futuros.

CAPÍTULO 2

Revisão Bibliográfica

Neste capítulo será apresentado uma breve descrição do Método do Elemento Finito. Serão apresentados também alguns trabalhos propondo modelos para análise do acoplamento termomecânico e análise de efeitos ambientais em sólidos, bem como algumas filosofias para construção de simuladores.

2.1 Método do Elemento Finito

O método do elemento finito (MEF) é uma técnica para discretização de equações diferenciais parciais que descrevem fenômenos, cujo comportamento é determinado por variáveis no contínuo (espaço e tempo). Este método é baseado na subdivisão de um domínio geométrico complexo utilizando entes geométricos mais simples (elemento finito), os quais são muito mais "tratáveis" matematicamente. Um computador é então utilizado em conjunto com um programa adequado, para resolver a equação diferencial em cada elemento finito. Então, considerando o equilíbrio e a compatibilidade no contorno entre os elementos uma certa quantidade de equações algébricas são obtidas. A solução dessas equações resulta em valores nodais para as funções. Os nós são então utilizados para descrever cada elemento e então todo o domínio. O MEF surgiu em 1956. A partir de então, um esforço enorme tem sido empregado para o desenvolvimento do MEF. Nos dias atuais, o método tem sido utilizado em transferência de calor, escoamento de fluido, acústica, eletrostática, guias de ondas eletromagnéticas, medicina e etc [4].

Considerando, como exemplo, que um dado fenômeno é definido pela função $u : \Omega \rightarrow R$, com $u \in H$, onde $\Omega \subset R^n$, $n \geq 1$, é um domínio geométrico fechado e H é um espaço de funções apropriado. Supondo que u satisfaz a lei de comportamento (uma equação diferencial parcial e condições de contorno, por exemplo) $L(u) = f$, onde $L : H \rightarrow Y'$ é um operador diferencial e $f \in Y'$ é dado. Y é um espaço vetorial e Y' é o espaço dual de Y. Baseado na equação diferencial parcial, uma formulação integral equivalente (forma fraca) é obtida através da forma bivariante $B_\Omega : H \times Y \rightarrow R$, tal que $B_\Omega(u,v) = L_\Omega(v)$, para todo $v \in Y$, onde $L_\Omega : Y \rightarrow R$ é um funcional linear. Isto corresponde ao cenário inicial do processo de discretização pelo método do elemento finito. Os principais processos envolvidos em uma simulação utilizando o método do elemento finito são [5]:

1. **Processo de geração de malha:** consiste na construção de uma aproximação, Ω_h para o domínio geométrico exato Ω, através da união de um conjunto (chamado de malha geométrica) de entes geométricos mais simples (arestas, triângulos, quadrilateros, tetraedros, hexaedros), onde cada ente geométrico pertencente a malha é chamado de um elemento

finito.

2. **Processo de discretização da forma fraca:** os espaços de dimensão finita S_H e S_Y são definidos de forma que uma aproximação da forma fraca original pode ser definido como $B_{\Omega h}(u_h, v_h) = L_{\Omega}(v_h)$, para todo $v_h \in S_Y$, onde $u_h \in S_H$ é a aproximação da solução exata u; $B_{\Omega h} : S_H \times S_Y \rightarrow R$ e $L_{\Omega h} : S_Y \rightarrow R$ são aproximações para B_{Ω} e L_{Ω}, respectivamente. u_h é definido pelo conjunto finito de parâmetros U, os quais são basicamente variáveis do problema. A forma fraca discreta representa um sistema (chamado global) de equações algébricas em U. Se o sistema é linear em u, então o sistema resultante global será um sistema algébrico linear, definido por uma matriz K e um vetor F. Neste caso, pode ser mostrado que tanto K quanto F podem ser montados através de matrizes e vetores locais, os quais são calculados baseados nas restrições de $B_{\Omega h}$ e $L_{\Omega h}$ para cada elemento finito na malha geométrica. Mesmo que o problema seja não linear, o processo de solução usualmente considera a solução de um sistema linearizado. Portanto, pode-se considerar que cada fenômeno contribui com um certo número de matrizes e vetores locais, os quais são calculados a nível do elemento.

3. **Processo do algoritmo de solução do sistema algébrico linear:** significa uma entre várias formas de realizar a simulação desejada, isto é, resolver o sistema algébrico global de equações, obtendo o vetor real U. É importante coletar todo o conjunto de possibilidades que um algoritmo de solução utiliza as matrizes e vetores claculados pelo fenômeno. Este conjunto, pode ser utilizado na definição de uma interface padrão entre os algoritmos de solução e fenômenos durante a simulação.

4. **Processo de pós-processamento:** este processo compreende o cálculo da quantia a ser visualizada e os procedimentos de visualização. Os dados de interesse para o usuário as vezes pode não ser o resultado do procedimento de solução (vetor U do exemplo acima), mas uma função deste resultado. O formato e o tamanho dos dados muitas vezes podem não ser compatível com o sofware utilizado na visualização. Nestes casos, as quantias a serem visualizadas são obtidas através de um processo posterior, o qual pode ser uma parte importante do simulador.

2.2 Acoplamento Termomecânico

Em função do alto desenvolvimento computacional alcançado nos últimos anos, a utilização de modelagens cada vez mais completas e sofisticadas que permitem simular e prever o comportamento de materiais usados em Engenharia tem sido empregadas.

Em [6] uma formulação completa de um modelo motivado fisicamente na termoplasticidade clássica em deformações finitas é descrito em detalhe. O modelo incorpora uma decomposição multiplicativa do gradiente de deformação em uma parte elástica e outra parte plástica. Duas características diferenciam este modelo termomecânico de modelos de plasticidade: a primeira é que a parte da entropia resultante do deslocamento e do movimento dos defeitos da rede é caracterizado no modelo por meio de uma variável interna independente chamada de entropia

plástica, e a segunda é a utilização de uma versão termomecânica estendida do princípio clássico da dissipação máxima. É proposta a decomposição do problema em dois subproblemas: um problema não linear de plasticidade com o campo térmico congelado e um problema de condução de calor. Os dois problemas são acoplados através do aquecimento estrutural e da dissipação mecânica.

Em [7], uma abordagem computacional para estimação da vida últil de estruturas submetidas a carregamentos termomecânicos é apresentada. O trabalho busca simplicidade e robustez em todas as etapas da modelagem, a fim de combinar o método proposto com as restrições encontradas na indústria. No modelo, o comportamento global da estrutura é determinado pelo escoamento de fluidos, difusão térmica transiente e uma análise mecânica, os quais são considerados desacoplados.

Em [8] é apresentado um procedimento sistemático para a obtenção de equações constitutivas termodinamicamente admissíveis, o qual pode ser obtido a partir da definição de dois potenciais termodinâmicos (Energia Livre de Helmholtz e o Potencial de Dissipação) os quais são funções das variáveis de estado. O trabalho propõe uma modelagem termomecanicamente geral para o estudo do acoplamento termomecânico em problemas anisotérmicos. A modelagem proposta, foi particularizada para o caso de materiais elasto-viscoplásticos. O enfoque utilizado possibilita uma análise mais simples do acoplamento fortemente não linear entre os diferentes mecanismos dissipativos (plasticidade, dano, endurecimento e temperatura). Dois tipos de acoplamentos termomecânicos são abordados: o acoplamento nas equações de balanço (acoplamento das equações térmicas e mecânicas) e o acoplamento constitutivo (os coeficientes das equações constitutivas são funções da temperatura). O modelo proposto consiste em uma extensão, para o caso anisotérmico, do modelo para materiais elasto-viscoplásticos proposto em [9]. Essa extensão é feita através da modificação dos potenciais. O efeito da degradação do material foi incorporado à modelagem anisotérmica através da introdução de uma variável de estado chamada de dano.

Um modelo termodinamicamente admissível para simulação do comportamento da deformação plástica em materiais metálicos levando em consideração acoplamentos termomecânicos e efeitos de inércia é proposto em [10]. A partir de um modelo de dano local, a influência dos termos de inércia e da equação da energia no comportamento da deformação de materiais inelásticos como também seus efeitos na fase e na amplitude da onda e tensão são analisados. O modelo proposto, possibilita uma melhor compreensão de fenômenos complexos envolvendo deformações inelásticas tais como endurecimento material, aquecimento de metais, amolecimento de metais devido a solicitações mecânicas, localização da deformação plástica, propagação de onda e transferência de calor em meios inelásticos. Este modelo é uma extensão do modelo proposto em [8], onde termos de inércia são levados em conta. Este modelo é utilizado para estudar como a solicitação mecânica e efeitos dinâmicos associados interferem no comportamento térmico do material.

Neste trabalho, será implementado um modelo baseado em [8] e [10]. As equações que definem o estado do corpo são as seguintes (ver referência [8]):

$$\sigma = (1-D)E[(\varepsilon - \varepsilon^p) - \alpha(\theta - \theta_0)] \tag{2.1}$$

$$Y = (1-D)(b[1-e^{-dp}] + \sigma_p) \tag{2.2}$$

$$X = (1-D)(a_1c_1 + a_2c_2) \tag{2.3}$$

$$B^D = E[\frac{1}{2}(\varepsilon - \varepsilon^p) - \alpha(\theta - \theta_0)](\varepsilon - \varepsilon^p) + b(p + \frac{1}{d}e^{-d}p) + \sigma_p p + \frac{3}{4}[a_1c_1^2 + a_2c_2^2] \tag{2.4}$$

onde as variáveis p (deformação plástica acumulada), c_1 e c_2 (variáveis associadas ao endurecimento cinemático) são responsáveis pela caracterização do comportamento de materiais elasto-viscoplástico. As variáveis Y, $X = X_1 + X_2$ são forças termodinâmicas associadas ao endurecimento isotrópico e ao endurecimento cinemático respectivamente. A variável B^D representa a força termodinâmica associada à variável do dano D. A variável ε representa a deformação, ε^p é a variável que representa a deformação plástica, θ é a temperatura e σ é a tensão. Os parâmetros a_1, a_2, φ_1 e φ_2 são associados ao endurecimento cinemático, os parâmetro b e d são associados ao endurecimento isotrópico, E é o Módulo de Young, α coeficiente de dilatação linear, θ_0 temperatura absoluta de referência e σ_p tensão de proporcionalidade associada ao endurecimento isotrópico.

As leis de evolução do material junto com as equações de estado (Equações 2.1-2.4) formam um conjunto completo de equações constitutivas para materiais elasto-viscoplásticos. Estas equações são apresentadas a seguir [8]:

$$\dot{\varepsilon}^p = \left\langle \frac{f(\sigma,X,Y)}{K} \right\rangle^N \operatorname{sgn}(\sigma - X) \tag{2.5}$$

$$\dot{p} = |\dot{\varepsilon}^p| = \left\langle \frac{f(\sigma,X,Y)}{K} \right\rangle^N \tag{2.6}$$

$$\dot{c}_1 = \dot{\varepsilon}^p - \frac{2}{3}\frac{\varphi_1}{a_1}X_1\dot{p} \tag{2.7}$$

$$\dot{c}_2 = \dot{\varepsilon}^p - \frac{2}{3}\frac{\varphi_2}{a_2}X_2\dot{p} \tag{2.8}$$

$$\dot{D} = \frac{B^D}{S_0}\dot{p} \tag{2.9}$$

onde $f(\sigma,X,Y)$ é a função de escoamento ou função de plastificação, K e N são parâmetros constitutivos associados ao comportamento elasto-viscoplástico e S_0 é um parâmetro associado ao mecanismo de evolução do dano.

A equação da energia para os materiais estudados envolve o acoplamento entre as respostas mecânica e térmica. No modelo considerado tem-se a seguinte equação de energia [8]:

$$\begin{aligned} A\rho c_p\dot{\theta} - \frac{\partial}{\partial x}(Ak\frac{\partial\theta}{\partial x}) + h_L p_M(\theta - \theta_\infty) - Q_N p_N - fA - \sigma\dot{\varepsilon}^p + Y\dot{p} + X_1\dot{c}_1 + X_2\dot{c}_2 - \\ - B^D\dot{D} - \theta\left(\frac{\partial\sigma}{\partial\theta}(\dot{\varepsilon} - \dot{\varepsilon}^p) + \frac{\partial Y}{\partial\theta}\dot{p} + \frac{2}{3}\frac{\partial X_1}{\partial\theta} + \frac{2}{3}\frac{\partial X_2}{\partial\theta} - \frac{\partial B^D}{\partial\theta}\dot{D}\right) = 0 \end{aligned} \tag{2.10}$$

onde A é a área da seção transversal, ρ é a densidade, c_p é o calor específico, $Q_N p_N$ é o termo de fluxo, fA é o termo de fonte, $\frac{\partial}{\partial x}(Ak\frac{\partial\theta}{\partial x})$ é o termo de condução, $h_L p_M(\theta - \theta_\infty)$ é o termo de convecção. Os termos $\sigma\dot{\varepsilon}^p$, $Y\dot{p}$, $B^D\dot{D}$, $X_1\dot{c}_1$, $X_2\dot{c}_2$ são os acoplamentos intrínsecos devido

a deformação plástica, deformação plástica acumulada, endurecimento isotrópico, dano e o endurecimento cinemático. Os termos dados por

$$-\theta\left(\frac{\partial\sigma}{\partial\theta}(\dot{\varepsilon}-\dot{\varepsilon}^{p})+\frac{\partial Y}{\partial\theta}\dot{p}+\frac{2}{3}\frac{\partial X_1}{\partial\theta}+\frac{2}{3}\frac{\partial X_2}{\partial\theta}-\frac{\partial B^{D}}{\partial\theta}\dot{D}\right) \tag{2.11}$$

correspondem ao acoplamento térmico, que está relacionado com acoplamento dos parâmetros constitutivos com a temperatura e das forças termodinâmicas com a temperatura.

2.3 Danos Provocados por Efeitos Ambientais

Os fatores ambientais, tais como ar úmido, água, assim como substâncias utilizadas na indústria intensificam o processo de degradação de materiais. Cerca de um terço de todos os casos de falhas são provocadas por efeitos ambientais, tais como corrosão, fragilização por hidrogênio e corrosão sob tensão fraturante [11]. Na Figura 2.1 [11], pode ser visto uma falha em um tanque de armazenamento de hidrogênio comprimido, esta falha foi provocada pela influência do hidrogênio. Em 2001, uma das mais modernas plataformas de petróleo do mar do norte, a Shell Shearwater, teve seu funcionamento interrompido após dois meses de operação. A paralisação foi conseqüência de uma inesperada fratura em um componente mecânico submerso, fabricado com a liga Inconel 718, de um dos seus poços. Após uma minuciosa investigação, ficou comprovado que um procedimento inadequado durante o tratamento térmico do componente, juntamente com as condições de serviço, tornou-o susceptível a fragilização por hidrogênio resultando no inesperado e indesejável evento de fratura [12]. Devido a esses importantes fatos, existem na literatura uma vasta quantidade de artigos propondo modelos que caracterizam, separadamente, os efeitos de fragilização por hidrogênio, corrosão, difusão etc., no entanto, esse número é bastante pequeno quando se diz respeito a modelos acoplados de difusão-dano-elasticidade.

Figura 2.1 Acidente devido a ruptura e explosão de um tanque de armazenamento de hidrogênio em uma fábrica

A fragilização por hidrogênio consiste em um tipo severo de falha ambiental. Quando o hidrogênio está presente, o material falha em níveis de cargas bem abaixo daquelas que o

material é capaz de suportar. Apesar do extensivo estudo, o mecanismo de fragilização por hidrogênio permanece ainda pouco claro. Vários tipos de mecanismos estão envolidos, cada um suportado por observações experimentais e forte ponto de vista pessoal. Em [13], uma revisão na análise do comportamento mecânico do material na ponta da trinca ou ao redor de um entalhe com difusão de hidrogênio é realizada. O trabalho tenta endereçar o papel dos parâmetros significativos de acordo com que se encontra em observações experimentais, tais como tensão hidrostática, deformação plástica e concentração de hidrogênio.

A fadiga por corrosão e a corrosão por tensão fraturante também são duas importantes falhas ambientais. Em [14], um tratamento do ponto de vista da mecânica dos sólidos desses dois fenômenos é realizado. A análise apresentada é baseada em uma síntese da mecânica da fratura e da mecânica do dano contínuo. A propagação da trinca é considerada como resultado da interação entre o balanço global das forças e energia no sistema fraturado corpo-carregamento e o processo de acumulação de dano disperso. O dano é descrito por meio de funções contínuas de tempo e espaço. Em comparação com a análise de fratura por fadiga ordinária, são incluídos tipos não mecânicos de dano, tais como químicos, eletromecânicos, radiativo ou biológicos. Para cada tipo de dano é indrotuzido uma medida especial de dano e as equações cinéticas que governam suas evoluções. Concentrações, fluxos e temperatura entram nessas equações como parâmetros de controle similar a tensões externas nas equações de acumulação de dano mecânico.

Em [15], é proposto um modelo acoplado de difusão-deformação em sólidos não porosos. O acoplamento entre a deformação e a difusão é introduzido através das equações constitutivas. O acoplamento entre a deformação e a concentração de soluto é o acoplamento termostático (difusão assistida pela tensão e teoria da tensão química). A difusão assistida pela tensão, incorpora o gradiente da tensão como uma força que governa o transporte da massa, no entanto ela despreza a influência da concentração na deformação do sólido. A teoria da tensão química, por outro lado, inclui o gradiente de concentração nas relações tensão-deformação, no entanto ela ignora a influência da deformação no processo de transporte. O processo de ligações químicas entre as espécies que estão difundindo e os "locais" atômicos ou microestruturais dentro do sólido também são considerados no trabalho. A força das ligações são medidas pela magnitude da energia de ativação.

Os aspectos da modelagem e simulações numéricas de processos térmico-químico em sólidos multifase são discutidos em [16]. Neste trabalho, é proposto um modelo o qual descreve a difusão de um soluto diluído em materiais sólidos heterogêneos, suas reações, produção de calor, mudanças no campo de tensões e a evolução das mudanças no material e deformações inelástica dentro do sólido.

Um modelo para deterioração de sólidos elásticos levando em conta a difusão de apenas um soluto é apresentado em [17]. Neste trabalho, é considerado que a deteriorização resulta de processos microescala tais como nucleação, crescimento de microtrincas os quais são provocados por carregamentos mecânicos. Esses processos podem ser influenciados pela presença de soluto, o qual difunde através do sólido. Nesta formulação, reações químicas e efeitos térmicos não são levados em conta. O desenvolvimento do modelo é feito utilizando a mecânica do contínuo, onde em adição ao campo que descreve o movimento do sólido são introduzidos campos microestruturais para representar a medida de dano, a concentração do soluto e o fluxo

de soluto. Utilizando o Princípio da Potência Virtual, quatro sistemas de balanço de forças são introduzidos. A versão mecânica da Segunda Lei (desigualdade da dissipação) também é considerada, onde a noção de potencial químico é introduzida. Após a introdução da teoria constitutiva consistente com a Segunda Lei, as equação que governam o problema são obtidas. Os acoplamentos são obtidos através da função de energia livre.

Neste trabalho, será implementado o modelo baseado em [17]. Neste trabalho, a variável d é responsável pela medida de dano, quando $d = 0$, o material é dito virgem e quando d aproxima-se da unidade o material está completamente danificado. O problema consiste em: conhecido as condições de contorno e inicial, os parâmetros do material, a força de corpo e a concentração de referência, calcular o deslocamento $\mathbf{u}$, o dano d e a concentração c satisfazendo as equações

$$\nabla \cdot \mathbf{S} + \mathbf{b} = 0 \tag{2.12}$$

$$\mathbf{S} = (1-d)\{\bar{\lambda}\mathrm{tr}(\mathbf{E})\mathbf{I} + 2\bar{\mu}\mathbf{E} - e(3\bar{\lambda} + 2\bar{\mu})(c - c_0)\mathbf{I}\} \tag{2.13}$$

$$\nabla \cdot \mathbf{J} + \dot{c} = 0 \tag{2.14}$$

$$\mathbf{J} = -D(\nabla c - \frac{c}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S}) + \omega\lambda d)) \tag{2.15}$$

$$\dot{d} = \frac{1}{b} < \hat{\varepsilon}(\mathbf{E}, 0, c) - \omega(1 - \lambda c) + \kappa\triangle d > \tag{2.16}$$

$$\hat{\varepsilon}(\mathbf{E}, 0, c) = \frac{1}{2}\{\bar{\lambda}(\mathrm{tr}(\mathbf{E}_e))^2 + 2\bar{\mu}|\mathbf{E}_e|^2\} \tag{2.17}$$

onde $\mathbf{S}$ é a tensão, $\mathbf{b}$ é a força de corpo, $\mathbf{J}$ é o fluxo de soluto, d é a medida de dano, c é a medida de concentração do soluto, as constantes e, α, D, λ, κ, ω, b são constantes positivas do material, $\bar{\lambda}$ e $\bar{\mu}$ são os parâmetros de Lamé, $\hat{\varepsilon}(\mathbf{E}, 0, c)$ é a energia armazenada do material virgem, c_0 é a concentração de referência e $\mathbf{E}$ é a deformação a qual pode ser decomposta em

$$\mathbf{E} = \mathbf{E}_e + e(c - c_0)\mathbf{I} \tag{2.18}$$

sendo $\mathbf{E}_e$ a parte elástica e $e(c - c_0)\mathbf{I}$ corresponde a parte química.

2.4 Sistemas para Construção de Simuladores

Em virtude da grande relevância das simulações em diferentes áreas de aplicações, a comunidade de pesquisadores em simulação computacional é bastante ativa, em virtude disso, pode-se enumerar uma grande quantidade de sistemas que estão relacionados com este trabalho (simuladores MEF e outros tipos de simuladores). No entanto, quando se fala em simuladores para problemas multi-física este número se reduz acentuadamente. Nesta seção serão discutidos alguns sistemas que lidam com a implementação de problemas multi-física.

O SCIRun foi desenvolvido pelo SCI (Scientific Computing and Imaging Institute) e consiste em um ambiente de programação científica que possibilita a construção e depuração interativa de cálculos científicos em larga escala. Permite o usuário controlar interativamente simulações cientificas enquanto o cálculo está em progresso. Este controle permite ao usuário variar condições de contorno e parâmetros computacionais durante a simulação. Uma aplicação

específica é construída através da conexão de elementos computacionais (módulos) para formar um programa. Este programa pode conter vários elementos computacionais assim como vários elementos de visualização, todos trabalhando em conjunto orquestrando uma solução para um problema específico. Entre o conjunto de ferramentas estão uma linguagem de programação visual, dispositivos de manipulação visual, bibliotecas de componentes para domínios específicos, um ambiente de desenvolvimento e um sistema otimizado de execução [18].

O Sierra, desenvolvido pela Sandia, fornece uma infra-estrutura comum de software para aplicações da mecânica computacional massivamente paralelas. O Sierra consolida serviços computacionais independentes os quais são necessários em diversas aplicações da mecânica. A consolidação desses serviços elimina seus desenvolvimentos redundantes e esforços de manutenção e organiza, em aplicações multi-mecânicas integradas, o acoplamento de potencialidades da mecânica computacional desenvolvidas de maneira independente. O Sierra possui uma biblioteca de elemento finito que fornece todos os métodos de cálculos discretos necessários para as aplicações (operador gradiente, interpolação, regras de integrações numéricas, por exemplo). O Sierra também possibilita que aplicações em uma única física sejam rapidamente acopladas produzindo pacotes multi-física [19].

O CTL (Component Template Library) é uma biblioteca do C++ para construir sistemas baseados em componentes. Esta biblioteca foi desenvolvida no Instituto de Computação Científica (Institute of Scientific Computing) da Universidade de Braunschweig. Aqui, componente é uma parte de software que consiste de uma interface bem definida e uma implementação os quais são conectados através de um canal de comunicação. A idéia do CTL é proporcionar um mecanismo para tornar o desenvolvimento de sistemas distribuídos tão fácil quanto possível. O CTL define uma descrição da interface de uma biblioteca que fornece toda as informações que os componentes devem compartilhar a fim de se comunicarem. Estas informações incluem nomes e assinaturas de funções, classes e seus métodos [20].

Apesar desses sistemas suportarem práticas e fundamentos úteis para definir notáveis sistemas de simulação, eles não fornecem um ambiente integrado que satisfaz as características necessárias para ambientes de simulações com fenômenos acoplados. Existem ainda importante questões relacionadas com a abstração de algoritmos numéricos que ainda continuam sem solução, tais como:

- Mecanismos que permitem uma fácil troca de métodos numéricos e estratégias;
- Organização e gerenciamento do repositório e suas relações com instâncias da simulação;
- Abstrações satisfatórias de acoplamento, as quais podem ser definidas independentemente da implementação atual do fenômeno participante;
- Abstrações para representar a relação fenômenos-geometria com a finalidade de decidir se cada fenômeno compartilhará ou não a sua malha com outro fenômeno (definido em uma mesma geometria), etc.

CAPÍTULO 3

Desenvolvimento do Simulador

Neste capítulo será discutido detalhes sobre a implementação do simulador. Para o melhor entendimento faz-se necessário a introdução de alguns conceitos:

Padrão: Os padrões descrevem idéias e perspectivas. Um padrão é uma pequena coleção de "estruturas" e uma descrição de suas relações. Um padrão deve expressar um tema geral recorrente o qual tem se mostrado útil [2].

Classe: Uma classe encapsula dados e funções as quais operam em dados da classe definindo uma dada estrutura e seu comportamento. Uma classe é um tipo e um escopo com uma especificação exata a qual é feita na declaração da classe [21].

Objeto: Um objeto é uma instância de uma classe [21].

Fenômeno: É uma ferramenta de abstração complexa responsável por produzir as contribuições de um fenômeno natural para as equações serem resolvidas em cada instante do processo de solução da simulação. O fenômeno também é responsável por outros tipos de operações, tais como, operações de pós-processamento, operações com vetores e matrizes no nível do elemento finito, etc [2].

Estado: Um conjunto de variáveis que representam o fenômeno. Os valores que essas variáveis assumem é que definem o estado do fenômeno em um determinado instante [2].

Skeleton: São partes do processo de solução os quais podem ser substituídos, possibilitando a construção de estratégias de solução diferente [2].

3.1 MPhyScas

O MPhyScas (Multi-Physics Multi-Scale Solver Environment) é um ambiente dedicado ao desenvolvimento automático de simuladores baseados no método do elemento finito. Deve-se entender o termo multi-física como um qualificador para um conjunto de interações entre fenômenos, no tempo e no espaço. Estes fenômenos são, usualmente, de naturezas diferentes (deformação de sólidos, transferência de calor, campos eletromagnéticos, etc.) e podem ser definidos em diferentes escalas de comportamento (comportamento macro e micro mecânico de materiais, por exemplo). Um sistema multi-física também é chamado um sistema com fenômenos acoplados. Se o exemplo descrito na seção 2.1 faz parte de um sistema multi-física, isto significa que B_Ω e L_Ω podem depender de outros fenômenos. Se este for o caso, o cálculo

das matrizes e vetores locais (no nível do elemento finito), item 2 da descrição apresentada na seção 2.1, dependem de dados de outros fenômenos, isto é, ele está acoplado com outros fenômenos. Outro tipo de dependência de dados, é o caso em que dois ou mais fenômenos são definidos em uma mesma componente geométrica e compartilham a mesma malha geométrica. As dificuldades em trabalhar com problemas acoplados serão descritas aqui, bem como a tecnologia elaborada para contornar estas dificuldades.

Os simuladores baseados no método do elemento finito podem ser moldados em uma arquitetura de camadas (ver Figura 3.1).

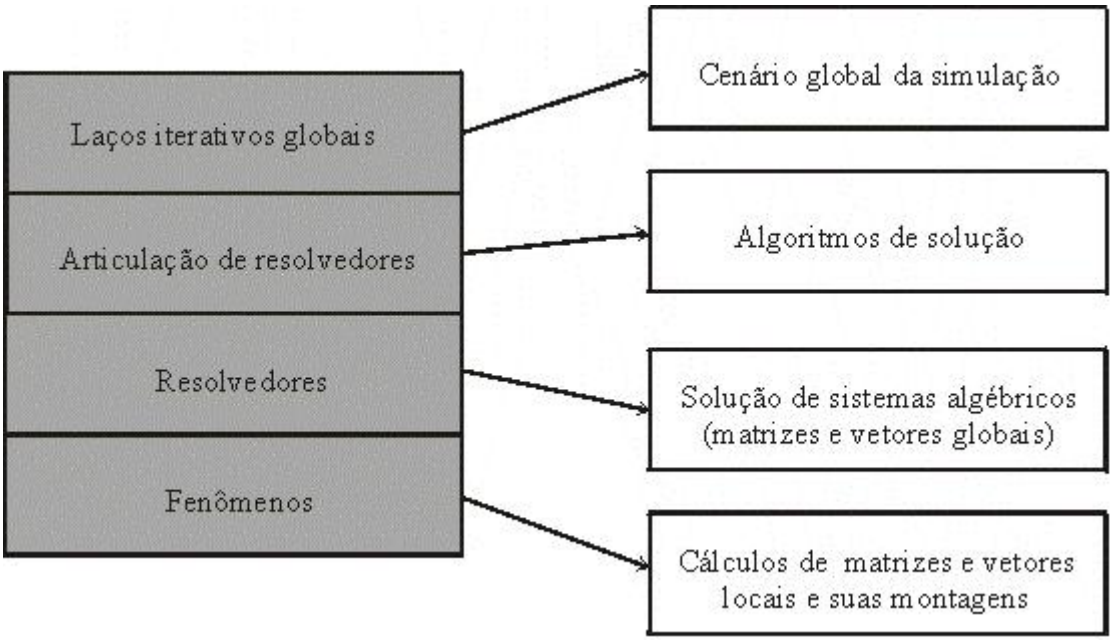


Figura 3.1 Camadas dos simuladores baseados no MEF

Na camada do topo têm-se os laços iterativos globais, os quais envolvem progressão no tempo, adaptação de modelos e discretização. Esta camada corresponde ao cenário global da simulação. Na camada seguinte tem-se a articulação entre os resolvedores, que corresponde ao algoritmo de solução. Esta camada articula soluções de diferentes sistemas algébricos. A próxima camada, resolvedores, é responsável pela montagem e solução de sistemas algébricos. A última camada, fenômeno, é responsável pelo cálculo e montagem de matrizes e vetores no nível do elemento finito (matrizes e vetores locais).

A definição dessas camadas é importante no sentido da modularização do software, no entanto, isto não fornece uma visão de como os procedimentos pertencentes a diferentes camadas interagem, nem como é realizada a troca de dados e, enfim, como pode ser descrita a dependência entre essas camadas. Isto de fato é muito importante na definição de abstrações, as quais podem padronizar a forma como processos e dados nas camadas se comportam e interagem. Com respeito a problemas desacoplados, existem bibliotecas para o método do elemento finito, as quais fornecem suficiente poder de representação computacional (abstração) com a finalidade de facilitar a construção de simuladores em um tempo razoável e com alto grau de

reusabilidade. Infelizmente, isso não é verdade quando se trata de fenômenos acoplados, ou seja, sempre que um fenômeno depende de dados de outro fenômeno, as abstrações destas bibliotecas se tormam ineficazes no sentido da reusabilidade e manutenabilidade. Isto ocorre pelo fato de que essas bibliotecas não fornecem abstrações para interação entre fenômenos e nem para a interação entre fenômeno e o algoritmo de solução.

Problemas multi-físicas acoplados tornam as coisas bastante complexas no nível do elemento finito e no nível da solução. As abstrações utilizadas em sistemas desacoplados não podem ser empregadas com eficiência - no que diz respeito a reusabilidade, adaptabilidade e manutenabilidade - em sistemas acoplados. Estas abstrações não representam de forma adequada a troca de dados e a dependência entre as camadas, as quais podem ocorrer em problemas acoplados. Isto acontece devido a existência de uma relação muito forte entre as decisões no nível de solução e os cálculos no nível do elemento finito.

A razão desta dificuldade pode ser esclarecida analisando os procedimentos no nível mais baixo, isto é, no nível do elemento finito. Estes procedimentos são relacionados com a produção e montagem das correspondentes matrizes e vetores locais para cada fenômeno. As matrizes e vetores locais calculadas para cada fenômeno podem estar acopladas com outros fenômenos, significando que o cálculo dessas quantias necessitam de informações de outros fenômenos. Essas informações são definidas no nível da solução, fazendo com que mudanças do algoritmo de solução possam exigir extensa reprogramação em ambas as camadas.

A fim de exemplificar, considere que um fenômeno P_i seja capaz de calcular um conjunto de quantias $\{Q_j(P_i)\}_{j=1..n_j}$. Durante a simulação, cada fenômeno possui um número predefinido fixo de estados, os quais são representados por dados calculados durante os vários estágios da simulação. Assumindo que uma dada quantia $Q_j(P_i)$, para um j qualquer seja acoplada a outro fenômeno P_k, a definição de acoplamento significa que um certo número de estados de P_k são utilizados no cálculo de $Q_j(P_i)$. Assuma que P_k possua o conjunto $\{St_s(P_k)\}_{s=1..m_k}$ como o conjunto de seus estados. P_i não sabe a priori quais estados de P_k serão usados no cálculo de $Q_j(P_i)$ até que o algoritmo de solução determine isto (no momento em que ele solicitar a P_i o cálculo de $Q_j(P_i)$). Esta escolha pode ser alterada, quando um algoritmo de solução diferente é utilizado. Supondo que apenas um estado seja requerido para o cálculo de $Q_j(P_i)$, isto é, o estado $St_r(P_k)$, para um certo r. Esta informação é então fornecida para P_i no momento em que ele é solicitado para calcular $Q_j(P_i)$, assim como a estrutura onde ele deve montar esta quantia. P_i deve solicitar o estado $St_r(P_k)$ e então ele será capaz de calcular a quantia $Q_j(P_i)$ (em cada elemento finito da malha) e montar esta quantia na estrutura de dado fornecida. É importante notar que, a decisão sobre o estado a ser utilizado no cálculo de uma quantia acoplada é feita a um nível substancialmente mais alto do que o nível do elemento finito. Além disto, o estado deve ser recuperado a fim de ser utilizado por P_i. Estes aspectos, produzem grandes dificuldades para o desenvolvimento de simuladores para fenômenos acoplados. No entanto, existem outros requisitos os quais fazem o problema ainda mais difícil de ser solucionado:

1. O acoplamento pode ocorrer em apenas uma parte do domínio geométrico (por exemplo, em alguma parte do contorno), ou ele pode ocorrer dinamicamente, como em problemas de contato;

2. Dois fênomenos, os quais são acoplados em um componente geométrico podem possuir

malhas geométricas diferentes, significando que o uso de um estado acoplado por um fenômeno necessita da transferência deste estado de uma malha para outra.

As dificuldades descritas geram o seguinte problema: Quais abstrações podem adequadamente representar e encapsular as informações, as relações e os processos pertencentes ao nível do elemento finito e que sejam do interesse da modelagem numérica de um fenômeno, a fim de descrever e implementar abstrações dos fenômenos no contexto de sistemas multi-físicos acoplados? Ainda, como representar abstratamente os processos e dados inerentes a cada uma das camadas de cálculo, garantindo que alterações em uma determinada camada impliquem em pouquíssima ou nenhuma alteração nas outras camadas? Nas próximas seções serão apresentadas soluções para estes problemas.

3.1.1 Arquitetura do MPhyScas

A arquitetura do MPhyScas, que foi proposta em [2], propõe uma representação computacional para as camadas utilizando padrões (ver Figura 3.2) onde, o **Kernel** representa os laços iterativos globais, a articulação entre os resolvedores é representada pelo **Block**, os resolvedores são representados pelo **Group** e por fim, os fenômenos são representados pelo **Phenomenon**. A definição desta estrutura tem a finalidade de melhorar a qualidade dos projetos de simuladores. A arquitetura definida é uma tentativa de preencher o vazio que existe no desenvolvimento de simuladores baseados no MEF. Os principais requisitos desta arquitetura são as seguintes:

- Flexibilidade no desenvolvimento de simuladores;
- Possibilidade de extensão do sistema através da integração de componentes;
- Reusabilidade de processos, dados e modelos.

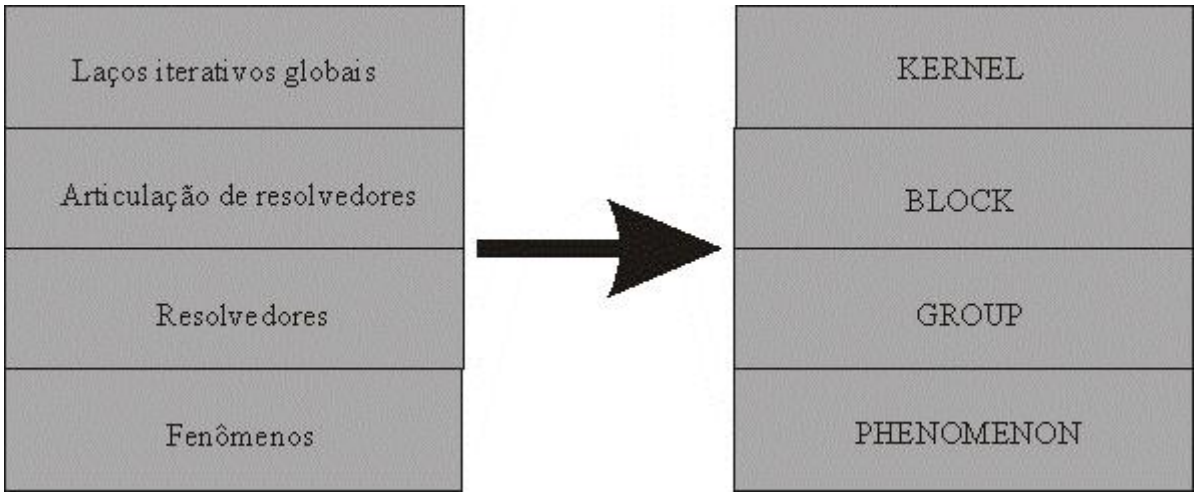


Figura 3.2 Representação computacional das camadas do simulador

A arquitetura do MPhyScas pode ser vista na Figura 3.3. A Biblioteca Estática possibilita a manutenção dos dados empregados na construção dos simuladores e simulações. Estes dados incluem: métodos (métodos de geração de malha, métodos de integração numérica, por

exemplo), funções (parâmetros constitutivos, por exemplo), algoritmos e fenômenos. O Pré-Processador tem a finalidade de produzir os Dados para Simulação e de construir o Simulador utilizando os dados da Biblioteca Estática (uma discussão mais detalhada sobre a construção do Simulador pelo Pré-Processador será realizada na seção 3.1.6). Os Dados para Simulação representam os dados de entrada da simulação a serem utilizados pelo Simulador. O Simulador é responsável pela simulação baseado no MEF de modelos com fenômenos acoplados (na seção 3.1.5 será apresentado um detalhamento do funcionamento do simulador). O Simulador utiliza os Dados para Simulação para realizar a simulação e produzir os Resultados da Simulação. O Visualizador utiliza os Resultados da Simulação e os Dados para Simulação para produzir a visualização dos resultados obtidos na simulação.

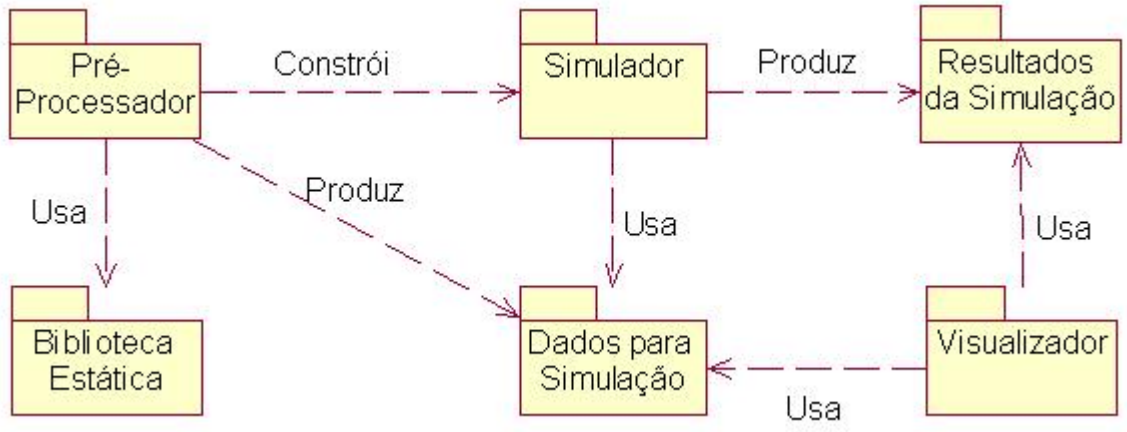


Figura 3.3 Arquitetura do MPhyScas

O Simulador aqui é considerado como um padrão [22], [23] e [24] (ver Figura 3.4) o qual, de uma forma simplificada, é um workflow na forma de uma árvore e dividido em quatro camadas:

- **Kernel**: É responsável por inicializar procedimentos (transfere para o **Block** em um nível mais baixo), por laços e iterações globais no tempo, por iterações adaptativas globais e por articulações de atividades a serem executadas pelos **Block**s em um nível mais baixo. O **Kernel** deve armazenar os dados do sistema relacionados com parâmetros para as suas iterações e laços;

- **Block**: É responsável pela transferência de solicitações do **Kernel** para os **Group**s pertencentes ao **Block** (procedimentos de inicialização, por exemplo), por iterações e laços locais do **Block** (iterações e laços internos dentro de um passo no tempo global, restritos a grupos de fenômenos), iterações locais do **Block** (restrito a grupos de fenômenos, iteração tipo Newton-Raphson, por exemplo), por procedimentos para cálculo do passo para avanço no tempo, por iterações adaptativas locais do **Block** (retrito a grupos de fenômenos) e por operações com quantias globais (remetidas para o **Group**, os quais são os donos das quantias globais, em um nível mais baixo). O **Block** serve ao **Kernel**. Cada **Block** é responsável por uma certa quantidade de **Group**s, os quais não podem pertencer

a outro **Block**. As solicitações de um **Block** para um nível mais baixo devem ser endereçadas aos seus **Group**s. O **Block** armazena parâmetros relacionados com suas iterações e laços e parâmetros relacionados com seus procedimentos;

- **Group**: É responsável pela transferência das solicitações do seu **Block** para o **Phenomenon** em um nível mais baixo (inicialização de procedimentos, por exemplo), por resolvedores de sistemas lineares (métodos diretos ou iterativos), por operações com quantias globais (solicitadas a partir de seu **Block**), pela articulação de atividades a serem executadas pelos **Phenomena** em um nível mais baixo (está basicamente relacionado com o cálculo e montagem de matrizes e vetores globais a partir da contribuição dos **Phenomena** pertencentes ao **Group**). Os **Group**s servem a seus respectivos **Block**s. Cada **Group** é responsável por uma certa quantidade de **Phenomena**, os quais não podem pertencer a outro **Group**. As solicitações a partir de um **Group** para um nível mais baixo devem ser endereçadas aos seus **Phenomena**. O **Group** armazena matrizes, vetores e escalares globais, armazena também **GroupTask**s, as quais são objetos que encapsulam procedimentos, onde a articulação dos **Phenomena** do **Group** são necessárias. As **GroupTask**s são programáveis. Os dados do **GroupTask** são informações padronizadas e dependem do seu tipo.

- **Phenomenon**: É responsável pelo cálculo de matrizes, vetores e escalares locais (quantias do **Phenomenon**), por operações envolvendo matrizes e vetores no nível do elemento finito e a montagem em matrizes e vetores globais fornecidos. Os **Phenomena** servem a seus respectivos **Group**s. O **Phenomenon** armazena dados relacionados a parâmetros constitutivos ou outros parâmetros os quais são específicos de um respectivo **Phenomenon**, armazena também a geometria onde o **Phenomenon** é definido (diferentes **Phenomena** podem compartilhar uma mesma geometria ou parte desta geometria), armazenam **WeakForm**s os quais são ferramentas para cálculo e montagem de quantias em uma determinada parte da geometria. A **WeakForm** representa partes da lei de comportamento discreto, condições de contorno e outras informações as quais são utilizadas pelo algoritmo de solução. Uma **WeakForm** pode ser ativada ou não, apenas **WeakForm**s ativadas podem ser utilizadas durante uma simulação. Uma **WeakForm** pode armazenar parâmetros, os quais estão relacionados com dados específicos da simulação (por exemplo, funções para a definição de condições de contorno, parâmetros para o cálculo de uma quantia, os quais podem ser fornecidos juntamente com um conjunto de dados da simulação). O **Phenomenon** também deve armazenar métodos, os quais são ferramentas utilizadas em tarefas específicas do **Phenomenon**, por exemplo, estas tarefas podem gerar as malhas geométrica e do **Phenomenon**, integração numérica no nível do elemento, funções de forma, etc.

A simulação começa com a execução da raiz do **Kernel**, o qual utiliza serviços providos pelo conjunto de **Block**s, os quais utilizam serviços de um conjunto de **Group**s. Cada **Group** possui um conjunto de objetos **Phenomenon**, os quais são utilizados na produção de matrizes e vetores locais e na montagem dessas quantias em matrizes e vetores globais fornecidos pelo seu **Group**.

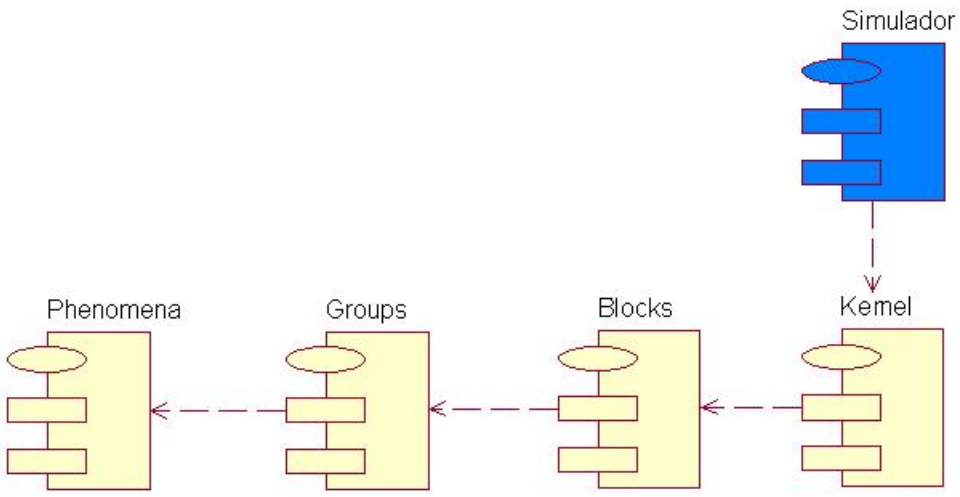


Figura 3.4 Diagrama do Simulador

3.1.2 O Padrão GIG - Generic Interface Graph

O padrão GIG (Generic Interface Graph) foi proposto em [2] e tem como finalidade representar o algoritmo na forma de um grafo acíclico orientado. O GIG lida com a definição e controle do fluxo de processos, tornando fácil a sua definição a partir da linguagem natural do algoritmo. Este padrão ajuda na reutilização de programas.

Como um exemplo, considere o algoritmo a seguir:

1. Para cada **Block** i $= 1$ até n_b
   (a) Recuperar o estado inicial para o **Block** i
   (b) Calcular o incremento inicial de tempo Δt para o **Block** i
   (c) Calcular os dados auxiliares iniciais para o **Block** i
2. Calcule o Δt inicial $\Delta t = \min_{1 \leq i \leq n_b}\{\Delta t_i\}$ e fazer o instante de tempo $t_1 = 0$
3. Enquanto $t_1 \leq T_{max}$ faça:
   (a) Faça $t_0 = t_1$ e $t_1 = t_0 + \Delta t$
   (b) Para o **Block** i $= 1$ até n_b
      i. Resolva para o **Block** i
      ii. Calcule o próximo Δt_i para o **Block** i
   (c) Calcule o próximo $\Delta t = \min_{1 \leq i \leq n_b}\{\Delta t_i\}$
4. Fim da simulação

A representação em forma de grafo deste algoritmo, utilizando o padrão GIG pode ser vista na Figura 3.5.

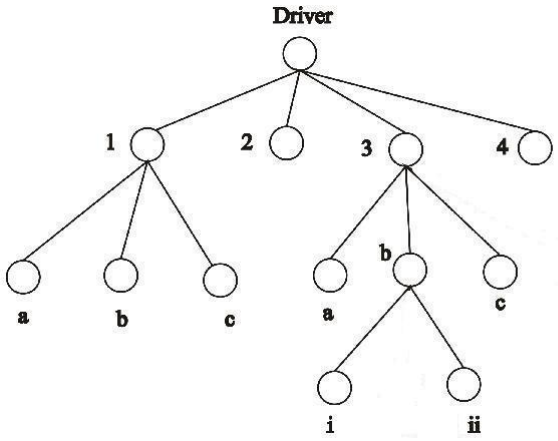


Figura 3.5 Representação do algoritmo na forma de grafo

3.1.3 O Padrão Block

No **Block** procedimentos relacionados com a articulação de soluções de diferentes sistemas algébricos são implementados. Na Figura 3.6 pode ser visto o diagrama do **Block**, onde observa-se que o **Block** possui objetos **Group**, os quais são solicitados pelo **Block** para montagem e solução de sistemas algébricos. O **Block** também possui um conjunto de algoritmos organizados na forma do Padrão GIG (**Skeleton**s). Estes algoritmos são os procedimentos que articulam as soluções de diferentes sistemas algébricos.

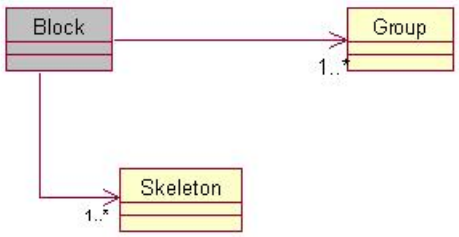


Figura 3.6 Diagrama do **Block**

3.1.4 O Padrão Group

No **Group** procedimentos relacionados com a montagem e solução de sistemas algébricos são implementados. No **Group** também são implementados operações com matrizes e vetores (blas 1, 2 e 3). Na Figura 3.7 pode ser visto a estrutura do **Group**. O **Group** possui um conjunto de estruturas de dados indexadas (ver GlobalStates na Figura 3.7) os quais podem ser compartilhadas pelos objetos **Phenomenon** deste **Group**. Ele também possui um conjunto de tabelas, os quais podem ser dinamicamente programados a fim de conter informações sobre o cálculo de matrizes e vetores, para os quais os objetos **Phenomenon** contibuem e como eles devem fazer isso (quais quantias do **Phenomenon** são utilizadas e, para cada quantia, quais os

estados acoplados).

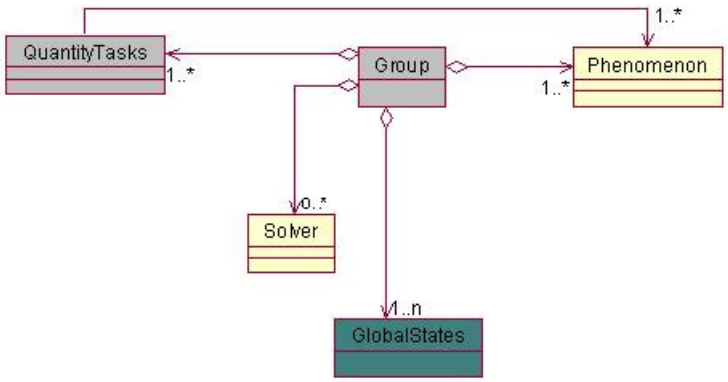


Figura 3.7 Diagrama do **Group**

O **Group** possui um conjunto de tarefas programáveis (ver QuantityTasks na Figura 3.7), estas tarefas podem ser de dois tipos (ver Figura 3.8):

- **GrpSolverTask**: Esta tarefa é responsável pela solução de sistemas algébricos. Os dados utilizados para a sua programação serão explicados na seção 3.1.6;
- **GrpBuilderTask**: Esta tarefa é responsável pela montagem de matrizes e vetores. Os dados utilizados para a sua programação serão explicados na seção 3.1.6.

É o nível do **Block** que determina o que deve ser calculado. No entanto, o **Block** transfere esta informação para o **Group** que transfere para cada objeto **Phenomenon** durante a demanda de cálculo de uma certa quantia. Todas as classes **Phenomenon** e **Group** devem ser implementadas de uma forma que poucas modificações devam ser realizadas, quando o algoritmo de solução é alterado (nível do **Block**). Algumas pequenas reprogramações dos procedimentos do **Block** e do **Kernel** fazem-se necessárias, mas as suas codificações como conjuntos de procedimentos articulados como workflows acomodará estas alterações sem muito trabalho.

3.1.5 O Padrão Phenomenon

Por simplicidade, um fenômeno é considerado como uma máquina para cálculo de matrizes e vetores locais e para a montagem dessas quantias em matrizes e vetores globais conhecidos. O cálculo dessas matrizes e vetores locais é realizado em cada elemento finito da malha, onde os seguintes requisitos devem ser satisfeitos:

1. O cálculo de matrizes e vetores locais pode estar relacionado com qualquer componente geométrica entre um conjunto de componentes geométricas, as quais definem o domínio físico de um fenômeno (uma parte do contorno, por exemplo);
2. Cada fenômeno deve possuir uma lista de quantias, as quais o fenômeno é capaz de calcular e montar;

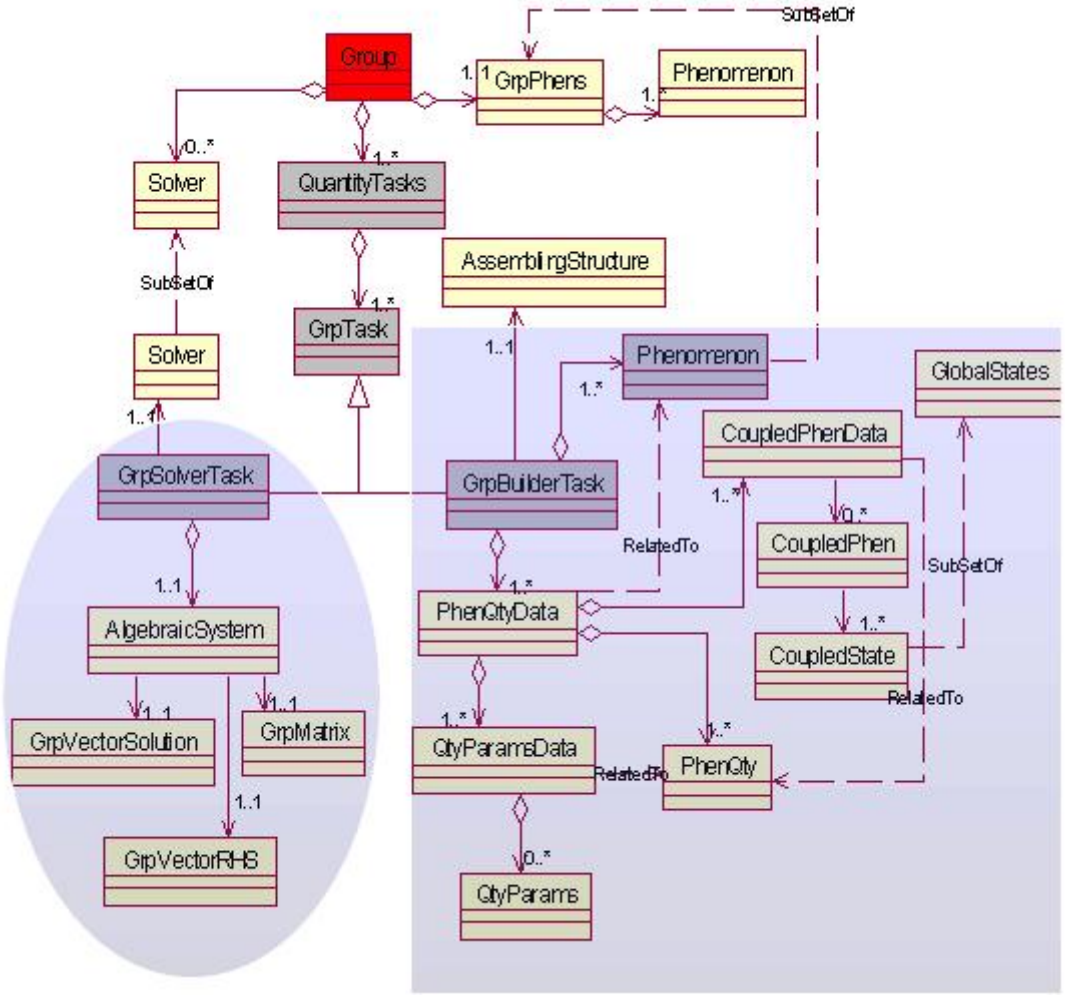


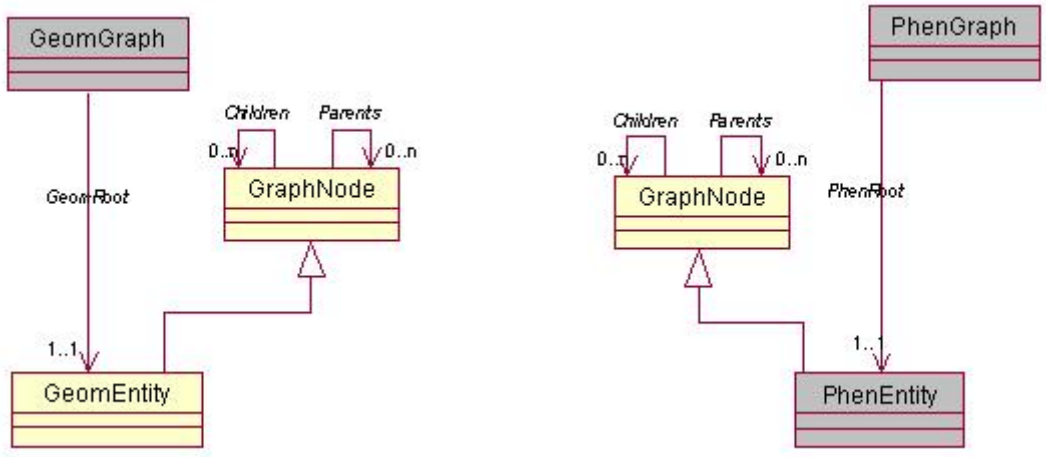
Figura 3.8 Diagrama do **Group** mostrando as QuantityTasks

3. A definição de estados acoplados necessária para o cálculo de uma determinada quantia de um fenômeno é determinado pelo nível do algoritmo de solução (nível dos **Block**s) e encapsulados no nível dos resolvedores (nível dos **Group**s). O fenômeno não deve ser informado sobre a maneira como ele é utilizado em uma simulação particular;

4. Dois fenômenos podem compartilhar malhas em qualquer parte geométrica que tenham em comum;

5. Um fenômeno é acoplado através de campos vetoriais (os quais podem ser dele mesmo ou de outros fenômenos) os quais tem uma definição fixa (dimensão do campo e do valor nodal, por exemplo). O fenômeno não é acoplado a priori a um fenômeno específico. A definição de acoplamento entre fenômenos é retardada até a definição dos dados do problema em uma simulação específica.

A solução proposta, a qual deve satisfazer os requisitos listados anteriormente, é o Padrão **Phenomenon** (será chamado de **Phenomenon** a partir de agora)[5], [25], [26], [27]. O **Phenomenon** representa uma abstração da coleção de atributos comuns encontrados nos conceitos e processos para representar simulações de fenômenos através do MEF. Esta abstração possibilita a representação do compartilhamento de dados e a dependência entre diferentes fenômenos de forma fácil e intuitiva. O **Phenomenon** pode ser visto como um "container" possuindo dois grafos acíclicos: o **PhenGraph** e o **GeomGraph** (ver Figuras 3.9, 3.10 e 3.11). Ambos possuem exatamente a mesma estrutura como grafo, mas as informações armazenadas em cada **GraphNode** são diferentes de um grafo para outro. O **PhenNode** e o **GeomNode** são os



(a) Grafo da geometria (b) Grafo do **Phenomenon**

Figura 3.9 Grafos da geometria e do **Phenomenon**

GraphNodes do **PhenGraph** e do **GeomGraph**, respectivamente. No **GeomGraph** os dados sobre a geometria são armazenados em uma estrutura do tipo representação pelo contorno (brep, boundary representation), isto é partindo da raiz do **GeomGraph** até as folhas são percorridos os entes geométricos de maior dimensão (volume, por exemplo) até os entes de menor

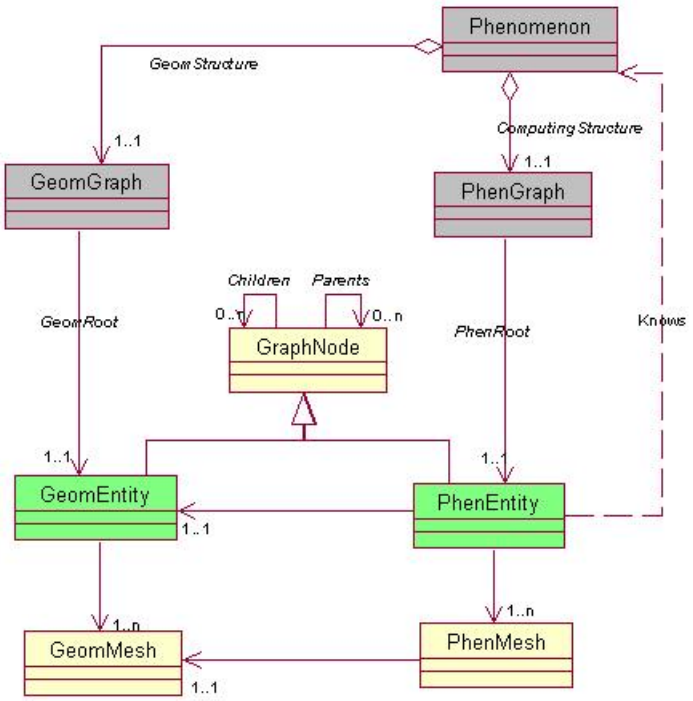


Figura 3.10 Diagrama do **Phenomenon**

dimensão (pontos). Por outro lado, no **PhenGraph** encontra-se em cada **PhenNode** um conjunto de **WeakForm**s, os quais são calculados no ente geométrico do respectivo **GeomNode**. Uma vez que a geometria pode ser compartilhada com outros fenômenos, é importante existir uma separação entre esses grafos. Cada **PhenNode** conhece (através de uma referência) seu respectivo **GeomNode** e portanto possui acesso aos dados do **GeomNode**, mas o contrário não é verdade.

O **Phenomenon** também contém um conjunto de quantias indexadas $\{Q\} = \{Q_i\}_{i=1,..,n}$ (ver Figura 3.11), o qual representa o conjunto de todas as quantias (matrizes e vetores locais) que podem ser calculadas e montadas em estruturas fornecidas (matrizes e vetores globais). Aqui, n é o número total de quantias (**WeakForm**s) que o **Phenomenon** é capaz de produzir e montar. Os procedimentos responsáveis para o cálculo dessas quantias devem estar armazenados no **PhenNode**, isto implica na necessidade de cada **PhenNode**, o j-ésimo por exemplo, possuir um conjunto indexado $q_j = \{Q_{jk}\}_{k=1,..,nj} \subset Q$ (ver PhenQty na Figura 3.11) que representa todas as quantias que o **PhenNode** j é capaz de calcular.

Cada **PhenNode** necessita de dados a fim de calcular uma determinada quantia. Esses dados são:

1. Dados gerais (servem para todas as quantias):

   (a) **GeomMesh**: é a malha de um ente geométrico armazenado em um **GeomNode**. É uma estrutura de dados que contém todos os elementos finitos geométricos (por exemplo, triângulos, quadrilateros, tetraedros, hexaedros, etc; depende da dimensão

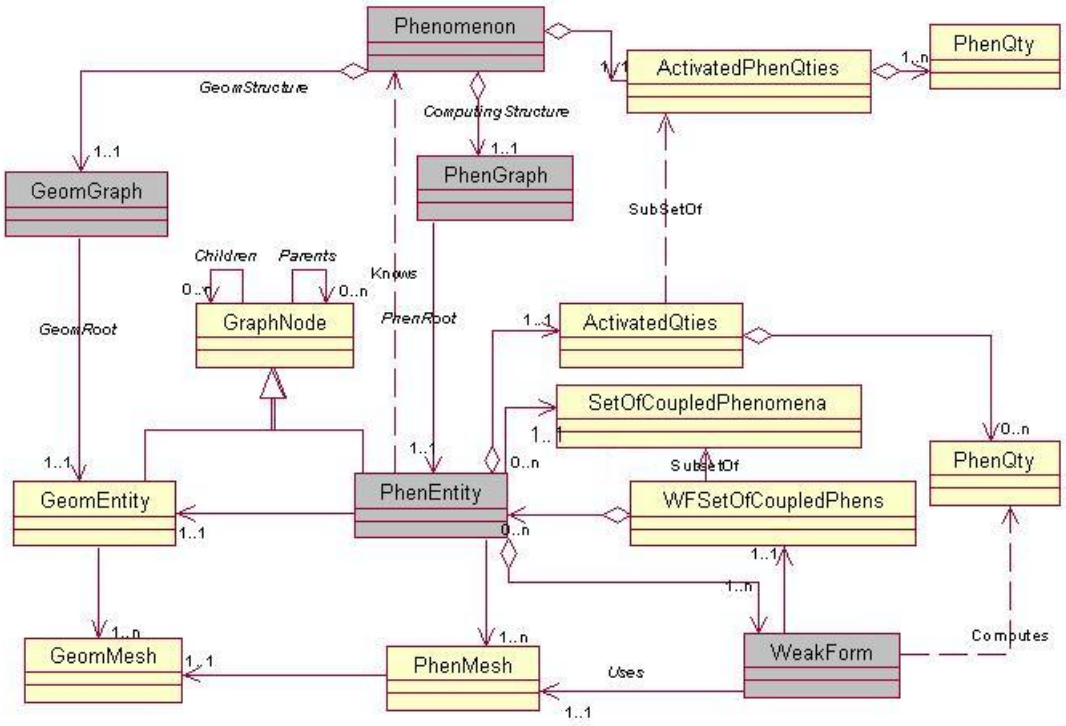


Figura 3.11 Diagrama do **Phenomenon**

geométrica e dos métodos de geração de malha utilizados). A malha geométrica pertence ao respectivo **GeomNode**;

(b) **PhenMesh**: é a malha de um fenômeno no nível do respectivo **PhenNode**. É uma estrutura de dados que contém informações sobre a aproximação do campo vetorial do fenômeno em cada elemento finito geométrico (por exemplo, a ordem da aproximação polinomial) da malha geométrica do **GeomNode** associado ao respectivo **PhenNode**. A malha do fenômeno que pertence ao respectivo **PhenNode** é fortemente relacionada com o **GeomMesh** do **GeomNode** associado ao respectivo **PhenNode**.

2. Dados específicos para uma certa quantia $Q_{jk,k=1,..,nj}$ do j-ésimo **PhenNode**:

(a) **CoupledPhenNodes** $CPN_{jk} = \{C_{jks}\}_{s=1,..,njk}$: é um conjunto de **GraphNode**s acoplados a esta quantia. Eles são utilizados a fim de obter os dados de outros fenômenos, os quais são importantes para o cálculo da quantia Q_{jk};

(b) **CoupledStates** $CPS_{jks} = \{S_{jksr}\}_{r=1,..,njks}$: é um conjunto de estados, os quais podem ser recuperados a partir do **CoupledGraphNode** C_{jks} (pois pertencem ao dono deste **PhenNode**). Esses estados normalmente representam campos vetoriais discretos. Q_{jk} sabe como relacionar os conjuntos $CPS_{jks,s=1,..,njk}$ com CPN_{jk}

A referência para os **PhenNode**s dos objetos **Phenomenon** acoplados que serão utilizados no cálculo de uma determinada quantia é armazenada no **PhenNode** do **Phenomenon** que a está calculando. O **PhenNode** pode possuir um conjunto de **WeakForm**s. Cada **WeakForm** é responsável pelo cálculo e montagem de uma única quantia (ver Figura 3.11).

É importante notar que o uso (por um determinado fenômeno) de um campo vetorial de outro fenômeno levanta um número de questões. Por exemplo, note que o uso de um campo vetorial implica na recuperação dos seguintes dados pertencentes ao outro **Phenomenon**:

- malha do **Phenomenon** e malha geométrica do **PhenNode** correspondente à região onde ocorre o acoplamento (as funções de forma estão encapsuladas nestes dados);
- campo vetorial discreto (valores nodais) do campo vetorial a ser utilizado.

Uma questão é a possibilidade de ambos **GeomMesh**s serem diferentes (este problema não é tratado neste trabalho). Neste caso objetos do tipo **Phenomenon** são construídos com o objetivo de transferir dados de uma malha para outra. Este **Phenomenon** possui as seguintes características:

- Compartilha a parte relevante do **GeomGraph** com o **Phenomenon** acoplado;
- A raiz do seu **PhenGraph** possui apenas as quantias necessárias para transferência de dados;
- Estas quantias estão acopladas com os **GraphNode**s dos **Phenomena** envolvidos;
- Como dados em uma malha (a acoplada) são utilizadas para cálculos em outra, existe a necessidade de uma estrutura para procura a fim de, dado um ponto qualquer, encontrar o elemento finito na malha do fenômeno acoplado que o possui. Isto é importante para realizar procedimentos de integração numérica em uma malha utilizando dados de outra malha.

Será tratado agora o problema onde os dados necessários para procedimentos em um nível mais baixo (cálculo de quantias em cada elemento finito) são determinados por um nível mais alto (o nível do **Block**) e armazenados (encapsulados) no nível do **Group** (nas **GroupTask**s). Antes de tratar desse problema faz-se importante uma melhor explicação de como os objetos **Phenomenon** são utilizados. As classes **Phenomenon** são altamente reutilizáveis. Essas classes contém informações as quais podem ser utilizadas em diferentes contextos e geometrias. Com a finalidade de alcançar esse estado, fez-se necessário que as definições sobre os estados acoplados fossem realizadas dinamicamente em tempo de execução. Esta definição depende do algoritmo de solução utilizado na simulação.

Nesse momento, pode-se fazer uma descrição mais detalhada da utilização de um objeto **Phenomenon** (**P**) pelo seu dono, o objeto **Group** (**G**) e pelo dono do **Group**, o objeto **Block** (**B**).

1. **B**, sabendo o que deve ser calculado, solicita a **G** que calcule uma determinada quantia. Esta solicitação é feita repassando para **G** um código que identifica a produção de uma determinada quantia (suponha a quantia Q_g);

2. **G**, solicitado por **B**, reconhece o código da quantia e então executa um conjunto de **GroupTask**s (**GT**s) os quais são responsáveis pela produção de Q_g;

3. Cada **GT** recupera, a partir de uma de suas tabelas, as referências para todos os objetos **Phenomenon** que contribuem para Q_g, onde **P** está entre eles;

4. **GT** recupera a partir de outra tabela um conjunto de dados para cada objeto **Phenomenon** que está contribuindo. Este conjunto de dados para o i-ésimo objeto contém as informações:

   (a) um conjunto $qt_i = \{q_{ij}\}_{j=1,..,ni}$ contendo todas as quantias do i-ésimo **Phenomenon** que contribuem para formar Q_g;

   (b) para cada q_{ij}, um conjunto $CS_{ij} = \{S_{ijk}\}_{k=1,..,nij}$, contendo os estados acoplados a serem utilizados neste cálculo - a ordem na qual esses estados são fornecidos é importante, visto que eles podem pertencer a fenômenos diferentes. O **Phenomenon** fará a associação dos estados com fenômenos acoplados pela ordem;

   (c) uma referência para a estrutura de dados (K_g) onde a quantia deve ser montada.

5. Supondo que **P** é solicitado para calcular uma determinada quantia q_r, juntamente com o conjunto de códigos para os estados acoplados ($CS_{pr} = \{S_{prk}\}_{k=1,..,npr}$) e a referência para K_g;

6. Então, **P** transfere a demanda para o seu **PhenGraph**, o qual transfere para a sua raiz (**R**);

7. **R** verifica se é capaz de calcular a quantia desejada. Em caso afirmativo, a demanda é enviada para um de seus objetos tipo **WeakForm** (**W**), o qual é responsável para o cálculo e montagem de q_r;

8. **W** já possui referências para todos os **PhenNodes** acoplados (cada um de um fenômeno diferente). A cada estado acoplado $S_{prk,k=1,..,npr}$ de CS_{pr}, **W** associa um dos **PhenNodes** acoplados;

9. Seja o **PhenNode** acoplado PN_k dono do estado acoplado S_{prk}, **W** solicita a PN_k os dados relativos a S_{prk}: **PhenMesh** e **GeomMesh** da região de acoplamento e o campo vetorial discreto dado pelo código global de S_{prk}. PN_k recebe esta solicitação. PN_k solicita o campo vetorial discreto ao seu **Group** e juntamente com as suas (de PN_k) **PhenMesh** e **GeomMesh**, envia os dados solicitados à **W**. Isto é feito para cada estado aoplado de CS_{pr}.

10. **W** percorre o **PhenMesh** (do seu **PhenNode**, neste caso, **R**) juntamente com os estados acoplados e respectivos **PhenMesh**es e calcula para cada elemento finito, a quantia q_r e monta em K_g;

11. Se **R** não é capaz de calcular a quantia desejada, **R** transfere a solicitação recursivamente para os seus **GraphNode**s filhos juntamente com os dados recebidos. O processo segue recursivamente até que não exista mais **GraphNode**s filhos a serem alcançados. Se ninguém pode calcular a quantia, uma exceção é lançada.

É importante notar que, na descrição do cálculo de q_r por **W** foi considerado que todos os objetos **PhenMesh** (de **R** e dos estados acoplados) compartilham o mesmo objeto **GeomMesh**, por esta razão eles podem ser percorridos ao mesmo tempo. Caso contrário, para cada ponto de integração em um elemento o valor do campo vetorial acoplado dependerá de uma procura na malha acoplada.

3.1.6 Construção do Simulador Pelo Pré-Processador

Nesta seção serão descritos os arquivos de dados necessários para a construção do Simulador pelo Pré-Processador e como o Pré-Processador utiliza esses dados para construir o Simulador.

Para o Pré-Processador realizar a construção do Simulador faz-se necessário que alguns dados sejam fornecidos para o Pré-Processador. Estes dados fornecem informações sobre a geometria, quais métodos devem ser utilizados no algoritmo de solução, quais quantias serão ativadas para cada **Phenomenon** e etc. Os arquivos de dados são escritos em formato texto e devem ser lidos pelos métodos do Pré-Processador. Os arquivos de dados serão descritos a seguir:

GeometricIn: este arquivo contém os dados utilizados na construção da geometria do problema;

PhenIn: este arquivo contém os dados utilizados na construção dos objetos **Phenomenon**;

PhenGeomIn: este arquivo contém os dados utilizados na construção da relação entre os objetos **Phenomenon** e a geometria;

ConsParIn: este arquivo contém os dados utilizados na construção dos parâmetros constitutivos (objetos **ConstPar**);

PhenConsParIn: este arquivo contém os dados utilizados na construção da relação entre os objetos **Phenomenon** e os objetos **ConstPar**;

GeomMeshIn: este arquivo contém os dados utilizados na geração da malha geométrica;

PhenMeshIn: este arquivo contém os dados utilizados na geração da malha dos objetos **Phenomenon**;

PhenMeshStateIn: este arquivo contém os dados utilizados na construção da relação entre a malha dos objetos **Phenomenon** e os seus estados;

GroupIn: este arquivo contém os dados utilizados na construção dos objetos **Group**;

PhenGroupsIn: este arquivo contém os dados utilizados na construção da relação entre os objetos **Group** e os objetos **Phenomenon**;

GrpStructureIn: este arquivo contém os dados utilizados na construção das estruturas de dados do objeto **Group**;

PhenQtyDataIn: este arquivo contém os dados utilizados na construção dos objetos **PhenQtyData**;

GroupTaskIn: este arquivo contém os dados utilizados na construção dos objetos **GroupTask**;

ActiveQtyIn: este arquivo contém os dados utilizados na ativação das quantias da simulação;

BoundCondIn: este arquivo contém os dados utilizados na ativação das condições de contorno da simulação;

PhenPhenIn: este arquivo contém os dados utilizados na construção da relação de acoplamento entre os objetos **Phenomenon**;

GroupGroupTaskIn: este arquivo contém os dados utilizados na construção da relação entre os objetos **Group** e os objetos **GroupTask**

AlgthmsIn: este arquivo contém os dados utilizados na construção dos algoritmos;

AlgthAlgthIn: este arquivo contém os dados utilizados na construção da relação entre os algoritmos;

BlockIn: este arquivo contém os dados utilizados na construção dos objetos **Block**;

AlgthBlocksIn: este arquivo contém os dados utilizados na construção da relação entre os objetos **Block** e os algoritmos (os dados do **Kernel** estão incluídos neste arquivo);

ProblemDataIn: este arquivo contém os dados utilizados na construção dos dados do problema;

WFParameterIn: este arquivo contém os dados utilizados na construção das funções que são utilizadas no cálculo das quantias.

Na Figura 3.12 são mostrados todos os dados necessários para construção do simulador, bem como a ordem em que estes dados são utilizados. Estes dados são utilizados pelos métodos do Pré-Processador que têm a finalidade de construir o Simulador. Esses métodos são descritos a seguir:

BuildGeometry: este método lê os dados no arquivo *GeometricIn*. A partir desses dados, o método constrói a geometria, partindo dos entes de menor dimensão (pontos) até os entes de maior dimensão (volumes). Durante a construção de cada ente geométrico é fornecido um código para sua identificação, este código é especificado no arquivo de dados. Após

a construção dos entes geométricos, o método constrói as relações entre os entes, ou seja, relaciona os pontos com as curvas, as curvas com as superfícies e as superfícies com volumes. Por fim, o método constrói os objetos **GeomGraph**, fornecendo um código para cada objeto e, então, relaciona os entes geométricos com o seu respectivo **GeomGraph**;

BuildPhenomena: este método lê os dados no arquivo *PhenIn*. A partir desses dados, o método constrói uma certa quantidade de objetos **Phenomenon** vazios, ou seja, sem nenhum dado (os dados de cada objeto serão adicionados por outros métodos). Para cada objeto **Phenomenon** é fornecido uma código o qual é utilizado na sua identificação. A quantidade de objetos **Phenomenon** bem como o código de cada objeto é especificado no arquivo de dados utilizado por este método;

BuildConstParameters: este método lê os dados no arquivo *ConstParIn*. A partir desses dados, o método obtém os códigos dos objetos **ConstPar** (os quais repsresentam os parâmetros constitutivos) e os dados desses objetos. O método então constrói os objetos **ConstPar** e fornece a estes os seus dados;

BuildPhenConstParameters: este método lê os dados no arquivo *PhenConstParIn*. A partir desses dados, o método obtém os códigos dos objetos **ConstPar** e os códigos dos objetos **Phenomenon**. O método então recupera os objetos **Phenomenon** (produzidos em **BuildPhenomena**) e os objetos **ConstPar** (produzidos em **BuildConstParameters**) e então relaciona os objetos **Phenomenon** com seus objetos **ConstPar**;

BuildPhenGeom: este método lê os dados no arquivo *PhenGeomIn*. A partir desses dados, o método é capaz de recuperar os objetos **Phenomenon** que foram produzidos pelo método **BuildPhenomena** e os objetos **GeomGraph** que foram produzidos pelo método **BuildGeometry**. Então, o método relaciona cada objeto **Phenomenon** com um objeto **GeomGraph**. O objeto **Phenomenon** se encarrega de construir o objeto **PhenGraph**;

GenGeomMesh: este método lê os dados no arquivo *GeomMeshIn*. A partir desses dados, o método cria, através do seu código, o gerador de malha geométrica que será utilizado. A raiz dos objetos **GeomGraph**, os quais foram produzidos no método **BuildGeometry**, é então fornecida para o gerador para que este gere a malha de toda a geometria;

GenPhenMesh: este método lê os dados no arquivo *PhenMeshIn*. A partir desses dados, o método cria, através do seu código, o gerador de malha do fenômeno que será utilizado. O método então recupera os objetos **PhenGraph** (produzidos no método **BuildPhenGeom**) através dos objetos **Phenomenon** (produzidos em **BuildPhenomena**). Os objetos **Phenomenon** são obtidos através de seus códigos. A raiz de cada objeto **PhenGraph** é então fornecida para o gerador de malha, para que a sua malha seja gerada. Cada malha é identificada através de um código o qual é fornecido no arquivo de dados;

BuildPhenMeshStates: este método lê os dados no arquivo *PhenMeshStateIn*. A partir desses dados, o método recupera, através dos seus códigos, o objeto **Phenomenon** (produzido em **BuildPhenomena**) que terá sua malha relacionada a um determinado estado. O código de cada estado e o código da malha (produzida em **GenPhenMesh**), os quais são

fornecidos no arquivo de dados, são repassados para o objeto **Phenomenon** que, por sua vez, se encarrega de fazer a relação entre a sua malha e o estado;

BuildGroups: este método lê os dados no arquivo *GroupIn*. A partir desses dados, o método cria uma certa quantidade de objetos **Group** vazios (sem nenhum dado). Para cada objeto **Group** é fornecido um código utilizado para sua identificação. A quantidade de objetos a serem construídos e os seus códigos são fornecidos pelo arquivo de dados.

BuildPhenGroups: este método lê os dados no arquivo *PhenGroupsIn*. A partir desses dados, o método recupera, através de seus códigos, os objetos **Group** (produzidos em **BuildGroups**) e os objetos **Phenomenon** (produzidos em **BuildPhenomena**). Para cada objeto **Group** são então fornecida os objetos **Phenomenon** dos quais ele é dono.

BuildGroupStructure: este método lê os dados no arquivo *GrpStructureIn*. O método obtém através do arquivo de dados o código global de cada estrutura de dado, o código do objeto **Group** onde uma determinada estrutura vai estar armazenada, o código dos objetos **Phenomenon** que contribuirão para a montagem da estrutura e o código do tipo da estrutura (escalar, vetor ou matriz). O método então recupera o objeto **Group** (produzido em **BuildGroups**) e fornece a este o código da estrutura, o código dos objetos **Phenomenon** e o tipo da estrutura que deverá ser construída. O **Group** por sua vez se encarrega de construir a estrutura.

BuildPhenQtyData: este método lê os dados no arquivo *PhenQtyDataIn*. Este método é responsável pela construção dos objetos **PhenQtyData**. Estes objetos armazenam informações sobre o cálculo e montagem de uma determinada quantia, ou seja, os códigos das quantias dos objetos **Phenomenon** que contribuirão para o cálculo e a montagem desta quantia, e, para cada quantia dos objetos **Phenomenon**, são armazenados os parâmetros utilizados no cálculo da quantia (se houver) e o código dos estados dos objetos **Phenomenon** acoplados (se houver). Estas informações estão contidas no arquivo de dados, bem como o código que será utilizado para identificação de cada objeto **PhenQtyData**. O método então cria os objetos **PhenQtyData** vazios (sem informação) e fornece o seu código. E, por fim, o método repassa todas as informações sobre o cálculo de uma determinada quantia para o respectivo objeto **PhenQtyData**;

BuildGroupTask: este método lê os dados no arquivo *GroupTaskIn*. O método obtém através do arquivo de dados o código do objeto **GroupTask**, o tipo do objeto (**GroupSolverTask** ou **GroupBuilderTask**) e os dados para cada objeto. O método cria o objeto **GroupTask** a partir do seu tipo e fornece a este o seu código. Se o objeto for um **GroupBuilderTask** deve ser fornecido os objetos **Phenomenon** (criado em **BuildPhenomena**), os objetos **PhenQtyData** (criado em **BuildPhenQtyData**) e o código da estrutura onde deverão ser montadas as quantias. Se ele for um **GroupSolverTask** deve ser fornecido a matriz, o vetor de solução, o vetor do lado direito (que constituem o sistema linear) e o resolvedor para o sistema linear;

ActiveQuantities: este método lê os dados no arquivo *ActiveQtyIn*. A partir do arquivo de dados o método obtém os códigos das quantias a serem ativadas, os códigos dos objetos

Phenomenon onde as quantias serão ativadas e os códigos dos entes geométricos onde as quantias serão produzidas. O método então cria a quantia com o código fornecido e recupera o objeto **Phenomenon** (produzido em **BuildPhenomena**). O método então fornece ao objeto **Phenomenon** a quantia a ser ativada e o código da geometria onde a quantia será produzida e este se encarrega da ativação da quantia;

SetBoundaryConditions: este método lê os dados no arquivo *BoundCondIn*. A ativação das condições de contorno pode ser vista como uma ativação de uma determinada quantia. Este método é semelhante ao método **ActiveQuantities**. O arquivo de dados utilizado por este método também é semelhate ao arquivo de dados *ActiveQtyIn*;

BuildPhenPhen: este método lê os dados no arquivo *PhenPhenIn*. A partir do arquivo de dados o método obtém os códigos dos objetos **Phenomenon**, os códigos dos objetos **Phenomenon** acoplados, os códigos das quantias através das quais os acoplamentos ocorrem e os códigos dos entes geométricos onde os acoplamentos ocorrem. O método então recupera o objeto **Phenomenon** (produzido em **BuildPhenomena**) e os objetos **Phenomenon** acoplados. O método fornece ao objeto **Phenomenon** o código da quantia e o código do ente geométrico onde o acoplamento ocorre juntamente com os objetos **Phenomenon** acoplados e este se encarrega de produzir o acoplamento. Isto significa fornecer para a **WeakForm**, que for responsável pelo cálculo da quantia, as referências dos **PhenNode**s dos fenômenos acoplados. Estes **PhenNode**s devem ser associados à região geométrica onde o acoplamento ocorre.

BuildGroupGroupTask: este método lê os dados no arquivo *GroupGroupTaskIn*. A partir do arquivo de dados o método obtém os códigos dos objetos **Group** e os códigos dos objetos **GroupTask**. O método então recupera os objetos **Group** (produzidos em **BuildGroups**) e fornece para cada objeto os seus objetos **GroupTask** (produzidos em **BuildGroupTask**).

BuildAlgthms: este método lê os dados no arquivo *AlgthmsIn*. A partir do arquivo de dados o método obtém os códigos dos algoritmos, os códigos das estruturas de dados dos algoritmos e as relações entre as estruturas de dados e os seus algoritmos. O método então constrói os algoritmos, constrói as estruturas de dados dos algoritmos (os dados para estas estruturas são fornecidas por outro método) e relaciona cada algoritmo com suas estruturas de dados.

BuilAlgthAlgth: este método lê os dados no arquivo *AlgthAlgthIn*. A partir do arquivo de dados o método obtém os códigos dos algoritmos (pais e filhos). O método recupera o algoritmo pai (produzido em **BuildAlgthms**) e os seus algoritmos filhos (produzidos em **BuildAlgthms**) e então relaciona estes algoritmos. Em seguida, faz o mesmo procedimento (recursivamente) com os seus filhos até que não haja mais algoritmo filhos.

BuildBlocks: este método lê os dados no arquivo *BlockIn*. A partir do arquivo de dados o método obtém os códigos dos objetos **Block**. O método então cria os objetos **Block** sem dados e fornece a cada objeto o seu código que será utilizado para sua identificação;

BuildAlgthBlocks: este método lê os dados no arquivo *AlgthBlocksIn*. A partir do arquivo de dados o método obtém os códigos dos objetos **Block** e os códigos dos algoritmos. O método então recupera cada um dos objetos **Block** (produzidos em **BuildBlocks**). Para cada objeto **Block** são recuperados os seus algoritmos (produzidos em **BuildAlgthms**). O método então fornece esses algoritmos para o objeto **Block**.

SetProblemData: este método lê os dados no arquivo *ProblemDataIn*. A partir do arquivo de dados são obtidos os códigos das estruturas de dados dos algoritmos e os seus dados a serem construídos. O método então recupera as estruturas de dados dos algoritmos (produzidos em **BuildAlgthms**) e fornece os dados para cada estrutura de dados. Os dados são referências para os objetos **Block**, para os objetos **Group** além dos parâmetros para os laços e iterações dos algoritmos.

SetWFParameters: este método lê os dados no arquivo *WFParameterIn*. A partir do arquivo de dados são obtidos os códigos dos parâmetros, os códigos dos objetos **Phenomenon**, os códigos das quantias onde os parâmetros serão utilizados e os códigos dos entes geométricos onde as quantias são produzidas. O método cria o parâmetro através do seu código (fornecido no arquivo de dados) e recupera o objeto **Phenomenon** (produzido em **BuildPhenomena**) dono da quantia onde o parâmetro será utilizado. O método então fornece para o objeto **Phenomenon** o parâmetro, o código da quantia e o código do ente geométrico onde a quantia é produzida e este se encarrega de fornecer o parâmetro para a quantia.

3.1.7 Um Exemplo Hipotético

Nesta seção será apresentado um exemplo hipotético para ilustração do sistema. Considere a geometria mostrada na Figura 3.13(a). Na Figura 3.13(b) pode ser visto que o fenômeno 0 (**Ph0**) é definido na curva **C7**, o fenômeno 1 (**Ph1**) é definido na superfície **S1** e o fenômeno 2 (**Ph2**) é definido na superfície **S2**.

Supondo que as equações algébricas discretas para **Ph0**, para **Ph1** e para **Ph2** sejam definidas pelas Equações 3.1, 3.2 e 3.3 respectivamente. Destas equações observa-se que **Ph0** deve ser capaz de produzir as quantias M_{21}, M_{22} e f_w, **Ph1** deve ser capaz de produzir as quantias M_{11}, M_{12} e f_u e **Ph2** deve ser capaz de produzir as quantias N e g, onde a produção dessas quantias estão acopladas com estados do próprio fenômeno e/ou estados de outros fenômenos.

$$\left[\begin{array}{cc} M_{21} & M_{22} \end{array}\right] \cdot \left[\begin{array}{c} u \\ w \end{array}\right] = [f_w] \tag{3.1}$$

$$\left[\begin{array}{cc} M_{11} & M_{12} \end{array}\right] \cdot \left[\begin{array}{c} u \\ w \end{array}\right] = [f_u] \tag{3.2}$$

$$[N] \cdot [v] = [g] \tag{3.3}$$

Suponha ainda que o grupo 0 (**Group0**) seja responsável pela montagem e solução do sistema mostrado na Equação 3.4 e o grupo 1 (**Group1**) seja responsável pela montagem e

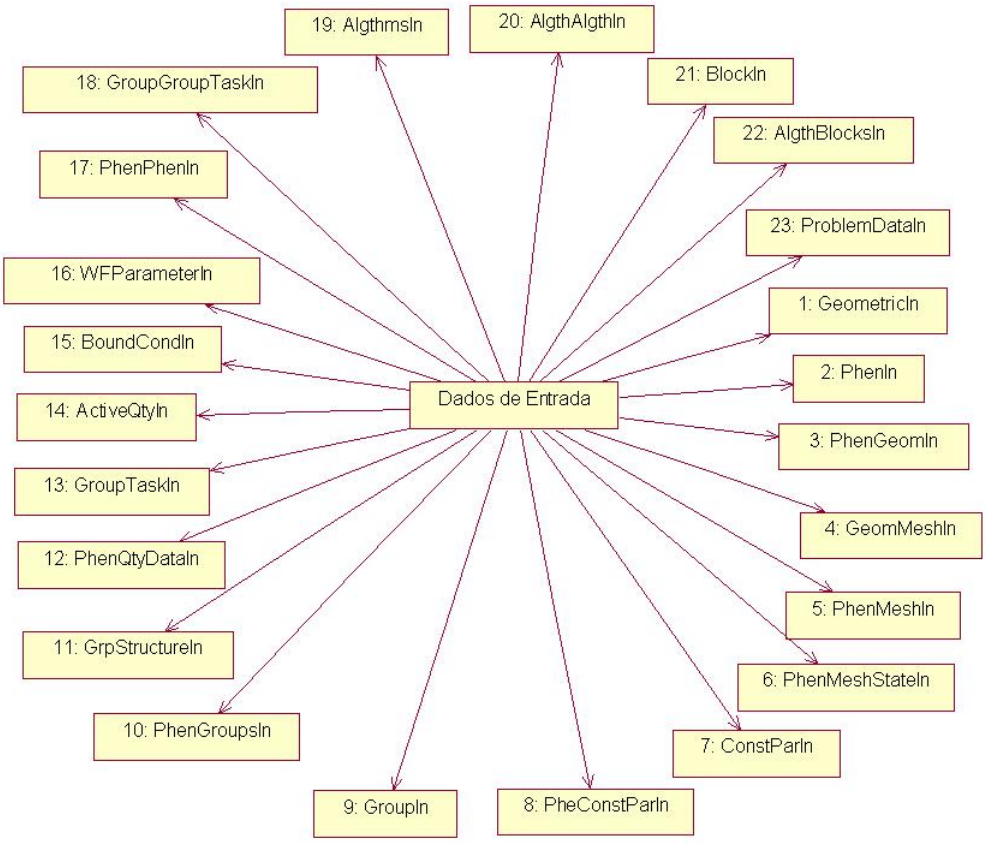
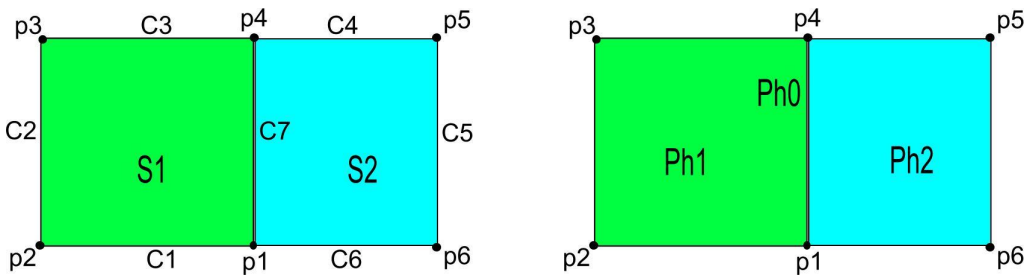


Figura 3.12 Arquivos de dados



(a) Geometria (b) Geometria × Fenômeno

Figura 3.13 Geometria e fenômenos para o exemplo hipotético

solução do sistema na Equação 3.5 e que o bloco 0 (**Block0**) articula o **Group0** e o bloco 1 (**Block1**) articula o **Group1**.

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot \begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} f_u \\ f_w \end{bmatrix} \tag{3.4}$$

$$[N] \cdot [v] = [g] \tag{3.5}$$

Considere hipoteticamente o seguinte algoritmo de solução:

```
calcule z_0
calcule v_0
obter Tol
obter Nmax
v_a = v_0
cnt = 0;
Error = 1.0;
z_kp1 = z_0
while(Error > Tol && cnt < Nmax)
{
  z_k = z_kp1
  calcule:
    M += M_11(u_k); // em S1
    M += M_12(w_k,v_a); // em C7
    M += M_21(u_0,v_a); // em C7
    M += M_22(u_k); // em C7
  calcule:
    f += f_u(u_k); // em S1
    f += f_w(u_k,w_k,v_a); // em C7
  resolva:
    M.z_kp1 = f;
  calcule:
    Du = z_kp1 - z_k;
    Error = ||D_u||;
  cnt++;
}
calcule:
  N += N(v_a); // em S2
calcule:
  g += g(u_k,u_kp1); // em C7
resolva:
  N.v_b = g;
```

Do algoritmo mostrado acima, observam-se os processos de inicialização (antes da entrada no laço), de cálculo e montagem das matrizes M e N, de cálculo e montagem dos vetores f e

g, e o processo de solução dos sistemas $M \cdot z_{kp1} = f$ e $N \cdot v_b = g$. Neste algoritmo, z_k e z_{kp1} representam o vetor z no passo anterior e no passo atual da iteração, respectivamente, onde z_k é composto pelos vetores u_k e w_k e z_{kp1} é composto pelos vetores u_{kp1} e w_{kp1}. Como pode ser visto, as quantias que são calculadas no processo de montagem da matriz M dependem de estados do próprio fenômeno e de estados de outros fenômenos. As quantias que são produzidas para montagem de M são as seguintes

- M_{11}, que é produzida em **S1** por **Ph1**, depende do estado u_k que está relacionado com **Ph1**;
- M_{12}, que é produzida em **C7** por **Ph1**, depende dos estados w_k, que está relacionado com **Ph0**, e v_a, que está relacionado com **Ph2**;
- M_{21}, que é produzida em **C7** por **Ph0**, depende dos estados u_0, que está relacionado com **Ph1** e v_a, que está relacionado com **Ph2** e;
- M_{22}, que é produzida em **C7** por **Ph0**, depende do estado u_k que está relacionado com **Ph1**.

As quantias que são produzidas para montagem de f são as seguintes

- f_u, que é produzida em **S1** por **Ph1**, depende do estado u_k que está relacionado com **Ph1**;
- f_w, que é produzida em **C7** por **Ph1**, depende dos estados u_k que está relacionado com **Ph1**, w_k, que está relacionado com **Ph0**, e v_a, que está relacionado com **Ph2**;

As quantias que são produzidas para a montagem do sistema $N \cdot v_b = g$ são as seguintes

- A quantia N é produzida em **S2** por **Ph2**, depende do estado v_a que está relacionado com o **Ph2**;
- A quantia g é produzida em **C7** por **Ph2**, depende dos estados u_k e u_{kp1} que estão relacionados com **Ph1**.

Como pode ser observado, o cálculo de uma determinada quantia pode estar acoplado com estados relacionados a outros fenômenos, onde esses estados podem estar localizados em grupos diferentes (caso os fenômenos perteçam a grupos diferentes), por esta razão faz-se necessário que cada estado receba um código global, através do qual o estado é conhecido nas várias partes e níveis do simulador e um código local, através do qual o estado é conhecido no grupo. Os códigos globais e locais para os estados bem como os grupos onde eles estão localizados podem ser vistos na Tabela 3.1.

Na Tabela 3.2 observa-se as relações de acoplamento, de onde pode ser visto, por exemplo, que o cálculo de f_w está acoplado com os estados z_k (u_k e w_k) e v_a. Portanto, faz-se necessário que essas informações sejam fornecidas pelo algoritmo para o fenômeno responsável pelo cálculo da quantia. Na Tabela 3.3 podem ser vistas as informações que devem ser repassadas para **Ph0** para que este calcule a quantia f_w.

Com isso, o indicativo dos processos do simulador fica da seguinte forma:

Kernel

```
Block0->Compute(8); // Calculando z_0
Block1->Compute(17); // Calculando v_0
Block0->Compute(10); // Calculando z_kp1
Block1->Compute(16); // Calculando v_b
```

Block0

```
bool Compute(int a)
{
 switch (a)
 {
  case 8:
  {
   return Group0->Compute(8);
  }
  case 10:
  {
   cnt = 0;
   Error = 1.0;
   Group0->Replace(a,8);
   Tol = Group0->RetrieveParam(0);
   Nmax = Group0->RetrieveParam(1);
   while(Error > Tol && cnt < Nmax)
   {
    Group0->Replace(9,10);
    Group0->Compute(0); // Calculando M
    Group0->Compute(5); // Calculando f
    Group0->Compute(10); // Resolvendo o sistema para z_kp1
    Group0->Compute(12); // Calculando o Erro
    Error = Group0->RetrieveState(12);
     cnt++;
   }
   }
  }
 if (cnt ≥ Nmax)
  return FALSE;
 else
  return TRUE;
}
```

Block1

```
bool Compute(int a)
{
 switch (a)
 {
```

```
  case 17:
  {
   return Group1->Compute(17);
  }
  case 16:
  {
   Group1->Compute(13); // Calculando N
   Group1->Compute(14); // Calculando g
   Group1->Compute(15); // Resolvendo o sistema para v_b
  }
 }
}
```

No nível do grupo, cada comando `Group0->Compute(i)` executa uma rotina da seguinte forma

```
bool Compute(int e)
{
 int c = getQtyGrpCode(e);
 int a = QuantityTasks[c]->GetNumberOfTasks();
 bool b;
 for(int i = 0;i < a; i++)
 {
   b = QuantityTasks[c]->Tasks[i]->ExecuteTask();
   if(!b)
     return b;
 }
 return b;
}
```

onde pode ser obeservado a execução das tarefas do grupo (no retângulo). Cada comando `ExecuteTask()` executa uma rotina da seguinte forma

```
bool ExecuteTask()
{
 //alguns ponteiros já foram definidos
 //Phenom = vetor dos fenômenos que contribuem para a quantia
 //desta tarefa
 //QuantityData = matriz onde QuantityData[i] contém um vetor
  //de QttyDataVec e i = fenômeno contribuinte
 //QttyDataVec = é um vetor de QttyData*
 //QttyData = dados para cálculo de uma quantia
 //AssemblingStruct = estrutura de dados (escalar,
```

```
 //vetor, ou matriz) onde a quantia será montada
 //CPPhenData = Dados de acoplamento de uma quantia a
 //ser calculada
 ParameterSet *Param;
 CouplingData *CpPhenData;
 QttyData *Qty;
 Phenomenon *Phen;
 vector<QttyData *> *QtyData;//QttyDataVec
 bool a = FALSE;
 int sizePhen = Phenom->giveSize();
 for(int i = 0;i < sizePhen; i++)
 {
   Phen = Phenom[i];
   QtyData = QuantityData[i];
   sizeQties = QtyData->giveSize();
   for(int j = 0; j < sizeQties; j++)
   {
     Qty = QtyData[j];
     cpPhenData = QtyData->giveCouplingData(j);
     Param = QtyData->giveParameters(j);
     a = Phen->Compute(Qty,cpPhenData,Param,AssemblingStruct);
     if(!a)
       return a;
   }
 }
 return a;
}
```

onde é observado as informações de acoplamentos sendo recuperadas (primeiro retângulo) e os objetos **Phenomenon** sendo demandados para calcular suas contribuições (segundo retângulo).

3.1.8 Algumas Informações sobre a Implementação

O sistema apresentado neste capítulo foi todo implementado neste trabalho. A linguagem de programação utilizada foi o C++. Esta linguagem foi escolhida pelo fato de suportar abstrações de dados, suportar programação orientada a objetos, suportar programação genérica, além de ser uma linguagem amplamente utilizada em processamento científico [28].

Na implementação desse sistema foram utilizados cerca de seiscentos arquivos, totalizando $1,5$ Mbytes. Cerca de duzentas classes foram implementadas, as quais podem ser destacadas: **Block**, **Group**, **Phenomenon**, **PreProcessor**, **PhenGraph**, **GeomGraph**, **WeakForm**, etc.

Tabela 3.1 Código dos estados para o exemplo hipotético

Nome	GblCd	Grupo	GrpCd
M	0	0	0
M_{11}	1	0	1
M_{12}	2	0	2
M_{21}	3	0	3
M_{22}	4	0	4
f	5	0	5
f_u	6	0	6
f_w	7	0	7
z_0	8	0	8
z_k	9	0	9
z_{kp1}	10	0	10
Du	11	0	11
$Error$	12	0	12
N	13	1	0
g	14	1	1
v_a	15	1	2
v_b	16	1	3
v_0	17	1	4

3.1.9 Considerações sobre a Formulação de Problemas

O MPhyScas foi desenvolvido com a finalidade de melhorar a qualidade dos projetos de simuladores baseado no MEF, tornando-os menos custosos. A arquitetura do MPhyScas não depende da formulação do problema, ou seja, ela pode ser utilizada para implementação dos mais diversos problemas. Esta característica foi observada neste trabalho através da implementação no MPhyScas de dois problemas diferentes (serão apresentados no próximo capítulo) utilizando a mesma arquitetura.

As modificações que são necessárias quando pretende-se implementar problemas diferentes devem ser realizadas apenas na Biblioteca Estática e nos arquivos de configuração do sistema, ou seja, devem ser inseridos na Biblioteca Estática novos métodos, novas **WeakForm**s, novas funções e novos algoritmos de solução e os arquivos de configuração devem ser modificados para a obtenção do novo problema.

Nota-se então que não são necessárias alterações na arquitetura do MPhyScas quando está sendo implementado problemas diferentes, ou até mesmo quando formulações diferentes para o mesmo problema são utilizadas, isto é um ponto forte do MPhyScas. A diferença de um programa procedural em relação a uma implementação no MPhyScas é que uma alteração no algoritmo de solução de um programa procedural resultará em uma modificação de praticamente toda a estrutura do programa enquanto que no MPhyScas esta alteração não faz-se necessário.

Tabela 3.2 Relações de acoplamento para o exemplo hipotético

										z_0	z_k	z_{kp1}					v_a		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	0																		
M_{11}	1										1								
M_{12}	2										1						1		
M_{21}	3									1							1		
M_{22}	4										1								
	5																		
f_u	6										1								
f_w	7										1						1		
	...										⋮								
$Error$	12										1	1							
N	13																1		
g	14										1	1							
	15																		
	16																		
	17																		

Tabela 3.3 Informações de acoplamento para cálculo de f_w no exemplo hipotético

Código do fenômeno no grupo	Código da quantia para o grupo	Código da quantia para o fenômeno	Código dos fenômenos acoplados	Código global dos estados acoplados
0	$f_w = 7$	2	$1 =$ **Ph1**	9
			$0 =$ **Ph0**	9
			$2 =$ **Ph2**	15

CAPÍTULO 4

Modelos e Implementações

Neste capítulo serão apresentadas as aplicações que foram desenvolvidas para o simulador. Neste trabalho, foram implementados dois modelos. O primeiro, consistiu em um modelo termodinâmico unidimensional levando-se em conta fenômenos tais como deformação plástica, temperatura, dano, endurecimentos cinemático e isotrópico. Uma primeira versão mais simplificada do simulador foi utilizada na implementação deste modelo. Este modelo foi escolhido pois consistia em um problema conhecido e que já havia sido implementado em MATLAB. O segundo modelo implementado (mais complexo) foi baseado em um modelo contínuo para deformação-difusão-dano em sólidos elásticos. Este modelo foi implementado em uma segunda versão, mais sofisticada, do simulador.

4.1 Modelo Elasto-Viscoplástico com Dano

Será implementado, utilizando o MPhyScas, um modelo unidimensional capaz de descrever acoplamentos em processos anisotérmicos de materiais elasto-viscoplásticos [8], [10], [29], [30]. A degradação do material será considerada através da introdução de uma variável interna chamada de dano $D \in [0,1)$. A variável D pode ser interpretada como uma medida da degradação do material, quando $D = 0$, o material é considerado virgem e quando D aproxima-se da unidade, o material está completamente danificado.

4.1.1 Geometria e Equação de Equilíbrio

O problema de uma barra cilíndrica solicitada axialmente será considerado. A barra tem um comprimento igual a L e uma área de seção transversal igual a A. A barra é engastada em uma das extremidades e a extremidade livre será submetida a dois tipos de carregamento: deslocamento prescrito ou força prescrita, ver Figura 4.1.

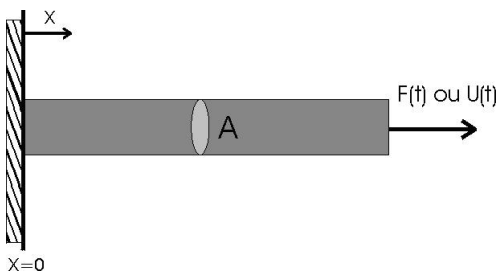


Figura 4.1 Geometria do problema unidimensional

A equação de equilíbrio para o problema proposto é dada por

$$-\frac{\partial}{\partial x}(\sigma A) + ku = f \tag{4.1}$$

onde A é a área da seção transversal, k é a constante do meio elástico, f a força de corpo, σ a tensão e u é o deslocamento.

4.1.2 Equações de Estado e Leis de Evolução

As equações de estado para este modelo foram obtidas utilizando uma formulação termodinamicamente admissível para a geometria considerada. Estas equações foram apresentadas nas Equações 2.1, 2.2, 2.3 e 2.4.

As leis de evoluções consideradas neste modelo podem ser vistas nas Equações 2.5, 2.6, 2.7, 2.8 e 2.9, onde as funções $\text{sgn}(x)$ e $\langle x \rangle$ são definidas por

$$\text{sgn}(x) = \frac{x}{|x|} \tag{4.2}$$

$$\langle x \rangle = \text{máximo}\{0, x\} \tag{4.3}$$

4.1.3 Equação da Energia

A equação de energia para a geometria considerada foi apresentada na Equação 2.10, onde

$$\frac{\partial \sigma}{\partial \theta} = -(1-D)E\alpha \tag{4.4}$$

$$\frac{\partial Y}{\partial \theta} = 0 \tag{4.5}$$

$$\frac{\partial X_1}{\partial \theta} = 0 \tag{4.6}$$

$$\frac{\partial X_2}{\partial \theta} = 0 \tag{4.7}$$

$$\frac{\partial B^D}{\partial \theta} = -E\alpha(\varepsilon - \varepsilon^p) \tag{4.8}$$

O termo de fluxo $Q_N p_N$, o termo de fonte fA, o termo de difusão $\frac{\partial}{\partial x}(Ak\frac{\partial \theta}{\partial x})$ e o termo de convecção $h_L p_M(\theta - \theta_\infty)$ não serão considerados, com isso a Equação 2.10 fica da forma

$$A\rho c_p \dot{\theta} \; - \; \sigma\dot{\varepsilon}^p \; + \; Y\dot{p} \; + \; X_1\dot{c}_1 \; + \; X_2\dot{c}_2 \; - \; B^D\dot{D} \; - \; \theta\left(\frac{\partial \sigma}{\partial \theta}(\dot{\varepsilon} - \dot{\varepsilon}^p) - \frac{\partial B^D}{\partial \theta}\dot{D}\right) \; = \; 0 \tag{4.9}$$

onde A é a área da seção transversal, ρ é a densidade e c_p é o calor específico.

4.1.4 Problema a Ser Resolvido

Estabelecidas as equações e a geometria utilizada, o problema a ser resolvido consiste em dada as condições de contorno, as condições iniciais, calcular para cada passo no tempo o deslocamento, a deformação, a deformação plástica, a deformação plástica acumulada, o endurecimento isotrópico, o endurecimento cinemático, o dano, a tensão e a temperatura que satisfazem a equação de equilíbrio (Equação 4.1), as equações de estados (Equações 2.1, 2.2, 2.3 e 2.4), as leis de evoluções (Equações 2.5, 2.6, 2.7, 2.8 e 2.9) e a equação da energia (Equação 4.9).

4.1.5 Utilização do Método de Decomposição do Operador

O método de decomposição do operador [10] será utilizado para solução do problema proposto. Essa ferramenta permite que técnicas numéricas realmente simples sejam utilizadas para obter uma solução aproximada do problema descrito, conseguindo-se uma boa estabilidade e precisão nos resultados obtidos.

A idéia básica do método consiste em uma decomposição aditiva do problema original em uma seqüência de outros problemas mais simples do tipo preditor/elástico e corretor/termoplástico de forma que possam ser aplicados métodos clássicos dos quais se conheçam bem o comportamento de estabilidade e convergência. Utilizando o método de decomposição do operador no problema implica que a cada passo será resolvido um problema elástico, no qual será aplicado o método do elemento finito, seguido da aplicação de um algoritmo termoplástico consistindo na solução de equações diferenciais ordinárias cujo o método de solução utilizado será o método de Runge-Kutta de quarta ordem. O método de Runge-Kutta foi utilizado por ser um método de fácil implementação e com uma boa precisão.

Considere a derivada da Equação 2.1, ou seja a taxa da tensão

$$\dot{\sigma} = -(1-D)E\alpha\dot{\theta} + (1-D)E(\dot{\varepsilon} - \dot{\varepsilon}^p) - E[(\varepsilon - \varepsilon^p) - \alpha(\theta - \theta_0)]\dot{D} \tag{4.10}$$

No preditor elástico será considerado que $\dot{\varepsilon}^p = 0$, $\dot{p} = 0$, $\dot{D} = 0$, $\dot{c}_1 = 0$, $\dot{c}_2 = 0$, então:

$$\dot{\sigma} = -(1-D)E\alpha\dot{\theta} + (1-D)E\dot{\varepsilon} \tag{4.11}$$

$$A\rho c_p\dot{\theta} = \theta\frac{\partial\sigma}{\partial\theta}\dot{\varepsilon} \tag{4.12}$$

utilizando a Equação 4.4 na Equação 4.12 então:

$$A\rho c_p\dot{\theta} = -\theta(1-D)E\alpha\dot{\varepsilon} \tag{4.13}$$

No corretor plástico será considerado que $\dot{\varepsilon} = 0$, então a Equação 4.10 fica da forma:

$$\dot{\sigma} = -(1-D)E\alpha\dot{\theta} - (1-D)E\dot{\varepsilon}^p - E[(\varepsilon - \varepsilon^p) - \alpha(\theta - \theta_0)]\dot{D} \tag{4.14}$$

e a Equação 4.9 fica da forma

$$A\rho c_p\dot{\theta} = \sigma\dot{\varepsilon}^p - Y\dot{p} - X_1\dot{c}_1 - X_2\dot{c}_2 + B^D\dot{D} + \theta\left(-\frac{\partial\sigma}{\partial\theta}\dot{\varepsilon}^p - \frac{\partial B^D}{\partial\theta}\dot{D}\right) \tag{4.15}$$

utilizando as Equações 4.4 e 4.8 na Equação 4.15, então

$$A\rho c_p\dot{\theta} = \sigma\dot{\varepsilon}^p - Y\dot{p} - X_1\dot{c}_1 - X_2\dot{c}_2 + B^D\dot{D} + \theta\left((1-D)E\alpha\dot{\varepsilon}^p + E\alpha(\varepsilon - \varepsilon^p)\dot{D}\right) \tag{4.16}$$

4.1.6 Formulação Usando o Método do Elemento Finito

Estabelecidas as considerações no preditor e no corretor, pode-se agora apresentar a solução numérica para o problema considerado. O primeiro problema a ser resolvido é o cálculo dos deslocamentos nodais. Multiplicando a Equação 4.1 pela função teste v e integrando no domínio tem-se

$$\int_0^L -\frac{\partial}{\partial x}(\sigma A)v dx + \int_0^L kuv dx = \int_0^L fv dx \tag{4.17}$$

sabe-se que

$$\int_0^L w_1 \frac{\partial w_2}{\partial x} dx = w_1 w_2|_0^L - \int_0^L w_2 \frac{\partial w_1}{\partial x} dx \tag{4.18}$$

utilizando a Equação 4.18 na Equação 4.17, então

$$\int_0^L [\sigma A \frac{\partial v}{\partial x} + ku] dx = \int_0^L fv dx + \sigma A v|_0^L \tag{4.19}$$

Substituindo a Equação 2.1 na Equação 4.19 e considerando apenas os termos do domínio

$$\int_0^L (1-D)EA\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}dx + \int_0^L kuv = \int_0^L (1-D)EA\varepsilon^p \frac{\partial v}{\partial x}dx + \\ + \int_0^L (1-D)EA\alpha(\theta-\theta_0)\frac{\partial v}{\partial x}dx \tag{4.20}$$

e o termo do contorno é dado por

$$\sigma A v|_0^L \tag{4.21}$$

O processo de discretização pelo método do elemento finito, envolve a definição de funções as quais serão utilizadas para interpolação do deslocamento (funções de forma). As funções de forma serão definidas em um elemento de referência, elemento onde serão realizadas as integrações da Equação 4.20. Na Figura 4.2 pode ser visto como é feito o mapeamento dos nós no elemento real para o elemento de referência e vice-versa.

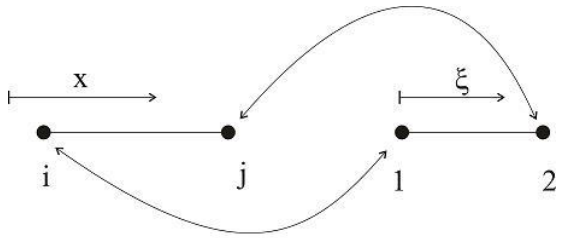


Figura 4.2 Mapeamento entre o elemento de referência e o elemento real

As funções de forma no elemento de referência são definidas como

$$\hat{\psi}_1(\xi) = 1 - \xi \tag{4.22}$$

$$\hat{\psi}_2(\xi) = \xi \tag{4.23}$$

e a função que mapeia o elemento de referência no elemento real é definida por

$$x = x_i + h_e \xi \tag{4.24}$$

sendo x_i, a coordenada do nó i e h_e o comprimento do elemento. Utilizando as funções de forma definidas e a transformação descrita e considerando E, A, k, f, ε^p, α, θ e θ_0 constantes em cada elemento pode-se calcular as contribuição de cada elemento.

$$\mathbf{K_1^e} = (1-D)\frac{EA}{h_e}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{4.25}$$

$$\mathbf{K_2^e} = \frac{kh_eA}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \tag{4.26}$$

$$\mathbf{f^e} = \frac{fh_e}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{4.27}$$

$$\mathbf{f^{ep}} = (1-D)EA\varepsilon^p\begin{pmatrix} -1 \\ 1 \end{pmatrix} \tag{4.28}$$

$$\mathbf{f^{e\theta}} = (1-D)EA\alpha(\theta-\theta_0)\begin{pmatrix} -1 \\ 1 \end{pmatrix} \tag{4.29}$$

onde h_e é o comprimento de cada elemento. A partir das contribuições de cada elemento pode-se montar um sistema de equações algébricas da forma:

$$\mathbf{Ku} = \mathbf{f} \tag{4.30}$$

4.1.7 Preditor Elástico e Corretor Plástico

Através da solução do sistema mostrado na Equação 4.30 obtém-se os deslocamentos nodais. A partir dos deslocamentos nodais é feito o cálculo da deformação em cada elemento. Considere o elemento mostrado na Figura 4.3, o cálculo da deformação no elemento é dado

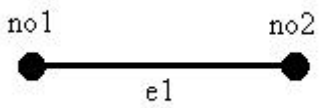


Figura 4.3 Elemento Finito 1D

por:

$$\varepsilon = \frac{u_{no_2} - u_{no_1}}{h_e} \tag{4.31}$$

onde u_{no_1} e u_{no_2} são os deslocamentos nodais dos nós 1 e 2 do elemento e h_e é o comprimento do elemento.

A evolução da temperatura e da tensão no preditor elástico será feita utilizando o método de Euler explícito, ou seja

$$\theta^{n+1} = \theta^n + \Delta\theta \tag{4.32}$$

$$\sigma^{n+1} = \sigma^n + \Delta\sigma \tag{4.33}$$

onde θ^{n+1} e σ^{n+1} são, respectivamente, a temperatura e a tensão no tempo atual e θ^n e σ^n são, respectivamente, a temperatura e a tensão no tempo anterior. Os termos $\Delta\theta$ e $\Delta\sigma$ serão obtidos utilizando a aproximação

$$\dot{(\bullet)} = \frac{\Delta(\bullet)}{\Delta t} \tag{4.34}$$

onde $\Delta(\bullet)$ é dado por

$$\Delta(\bullet) = (\bullet)^{n+1} - (\bullet)^n \tag{4.35}$$

sendo $(\bullet)^{n+1}$ é o valor da variável no tempo atual, $(\bullet)^n$ é o valor da variável no tempo anterior e Δt é o valor do incremento de tempo. Utilizando a aproximação mostrada na Equação 4.34 nas Equações 4.11 e 4.13, então:

$$\Delta\theta = -\theta^n(1-D)E\alpha\Delta\varepsilon \tag{4.36}$$

$$\Delta\sigma = -(1-D)E\alpha\Delta\theta + (1-D)E\Delta\varepsilon \tag{4.37}$$

com isso pode-se evoluir a temperatura e a tensão. Para finalizar o preditor, deve ser realizado o cálculo dos endurecimentos isotrópico e cinemático utilizando as Equações 2.2 e 2.3.

O corretor plástico apenas será realizado se a seguinte condição de escoamento for alcançada

$$f(\sigma, X, Y) > 0 \tag{4.38}$$

onde $f(\sigma, X, Y)$ é chamada de função de plastificação e é calculada por

$$f(\sigma, X, Y) = |\sigma - X| - Y \tag{4.39}$$

A função teste deve ser calculada para cada elemento e onde a condição mostrada na Equação 4.38 for satisfeita deve ser calculado o problema de evolução. Este problema consiste em calcular novos valores para ε^p, p, c_1, c_2, D, σ e θ utilizando $\dot{\varepsilon}^p$, $\dot{p}$, $\dot{c}_1$, $\dot{c}_2$, $\dot{D}$, $\dot{\sigma}$ e $\dot{\theta}$ dados pelas Equações 2.5, 2.6, 2.7, 2.8, 2.9, 4.14 e 4.16 respectivamente. Como foi mencionado antes, este problema será resolvido utilizando o método de Runge-Kutta de quarta ordem o qual para o problema proposto é resolvido da seguinte forma

1. $\mathbf{h_n}$ conhecido
2. Calcular $\mathbf{k_1} = \mathbf{g}(\mathbf{h_n}, t_n)$
3. Calcular $\mathbf{k_2} = \mathbf{g}(\mathbf{h_n} + \mathbf{k_1}\frac{\Delta t}{2}, t_n + \frac{\Delta t}{2})$
4. Calcular $\mathbf{k_3} = \mathbf{g}(\mathbf{h_n} + \mathbf{k_2}\frac{\Delta t}{2}, t_n + \frac{\Delta t}{2})$
5. Calcular $\mathbf{k_4} = \mathbf{g}(\mathbf{h_n} + \mathbf{k_3}\Delta t, t_n + \Delta t)$

6. Calcular $\mathbf{h_{n+1}} = \mathbf{h_n} + \frac{\Delta t}{6}(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4})$

onde

$$\mathbf{h_n} = \left[\begin{array}{cccccccc} \varepsilon^p & p & c_1 & c_2 & D & \sigma & \theta \end{array} \right]^T \tag{4.40}$$

$$\mathbf{g}(\mathbf{h_n}, t) = \left[\begin{array}{cccccccc} \dot{\varepsilon}^p & \dot{p} & \dot{c}_1 & \dot{c}_2 & \dot{D} & \dot{\sigma} & \dot{\theta} \end{array} \right]^T \tag{4.41}$$

Para finalizar o preditor, deve ser realizado o cálculo dos endurecimentos isotrópico e cinemático utilizando as Equações 2.2 e 2.3 nos elementos onde a condição da Equação 4.38 for satisfeita.

4.1.8 Algoritmo de Solução

O algoritmo para solução desse problema é mostrado a seguir:

1. Em cada instante de tempo zerar $W^{k+1}, W^k, \overline{W}_n, W_{n+1}$

2. Conhecendo

$$W_n = \left[\begin{array}{cccccccccccc} u_n & \sigma_n & \theta_n & p_n & \varepsilon_n & \varepsilon_n^p & c1_n & c2_n & D_n & X_n & Y_n \end{array} \right]^T$$

   (a) Calcular u_{n+1} usando $D_n, \varepsilon_n^p, \theta_n$ e salvar em W_{k+1}

   (b) Calcular ε_{n+1} usando u_{n+1} e salvar em W_{k+1} e em $\overline{W}_n$

   (c) Calcular $\Delta\varepsilon = \varepsilon_{n+1} - \varepsilon_n$

   (d) Calcular $\Delta\theta$ usando θ_n, D_n e $\Delta\varepsilon$ calculado em 2c

   (e) Calcular $\overline{\theta}_{n+1}$ usando $\Delta\theta$ calculado em 2d e armazenar em $\overline{W}_n$

   (f) Calcular $\Delta\sigma$ usando $D_n, \Delta\varepsilon$ calculado em 2c e $\Delta\theta$ calculado em 2d

   (g) Salvar $p_n, \varepsilon_n^p, c1_n, c2_n, D_n, X_n, Y_n$ de W_n para $\overline{W}_n$

   (h) Calcular $\overline{\sigma}_{n+1}$ usando σ_n e $\Delta\sigma$ calculado em 2f e armazenar em $\overline{W}_n$

   (i) Calcular F usando $\overline{\sigma}_{n+1}, X_n, Y_n$ que estão em $\overline{W}_n$

   (j) Se $F > 0$

      i. Corretor usando

$$\overline{W}_n = \left[\begin{array}{cccccccccccc} u_n & \overline{\sigma}_{n+1} & \overline{\theta}_{n+1} & p_n & \varepsilon_{n+1} & \varepsilon_n^p & c1_n & c2_n & D_n & X_n & Y_n \end{array} \right]^T$$

      completando W_{k+1}, o qual fica da seguinte forma

$$W_{k+1} = \left[\begin{array}{cccccccccccc} u_{k+1} & \sigma_{k+1} & \theta_{k+1} & p_{k+1} & \varepsilon_{k+1} & \varepsilon_{k+1}^p & c1_{k+1} & c2_{k+1} & D_{k+1} & X_n & Y_n \end{array} \right]^T$$

      ii. Zerar W_{k+1} em X e Y

(k) Senão

i. faz-se

$$W_{k+1} = [u_{k+1} \quad \sigma_{k+1} = \overline{\sigma}_{n+1} \quad \theta_{k+1} = \overline{\theta}_{n+1} \quad p_{k+1} = p_n \quad \varepsilon_{k+1} \quad \varepsilon^p_{k+1} = \varepsilon^p_n \\ c1_{k+1} = c1_n \quad c2_{k+1} = c2_n \quad D_{k+1} = D_n \quad X_n \quad Y_n]^T$$

(l) Fim Se

(m) Calcular X_{k+1} usando $c1_{k+1}, c2_{k+1}, D_{k+1}$ onde $F > 0$ e salvar em W_{k+1}

(n) Calcular Y_{k+1} usando D_{k+1}, p_{k+1} onde $F > 0$ e salvar em W_{k+1} o qual agora fica da forma

$$W_{k+1} = \left[\begin{array}{cccccccccc} u_{k+1} & \sigma_{k+1} & \theta_{k+1} & p_{k+1} & \varepsilon_{k+1} & \varepsilon^p_{k+1} & c1_{k+1} & c2_{k+1} & D_{k+1} & X_{k+1} \quad Y_{k+1} \end{array} \right]^T$$

3. $erro = 2 * tol$

4. $niter = 0$

5. while($erro > tol$ && $niter < nmax$) {

(a) Fazer $W_k = W_{k+1}$ e salvar em W_{k+1}

(b) Zerar $\overline{W}_n$ em $\sigma, \theta, \varepsilon, X, Y$

(c) Zerar W_{k+1}

(d) Calcular u_{k+1} usando $D_k, \varepsilon^p_k, \theta_k$ e salvar em W_{k+1}

(e) Calcular ε_{k+1} usando u_{k+1} e salvar em W_{k+1} e em $\overline{W}_n$

(f) Calcular $\Delta\varepsilon = \varepsilon_{k+1} - \varepsilon_n$

(g) Calcular $\Delta\theta$ usando $\theta_k, D_k, \Delta\varepsilon$ calculado em 5f

(h) Calcular $\Delta\sigma$ usando $D_k, \Delta\varepsilon$ calculado em 5f e $\Delta\theta$ calculado em 5g

(i) Calcular $\overline{\sigma}_{k+1}$ usando σ_n e $\Delta\sigma$ calculado em 5h

(j) Calcular $\overline{\theta}_{k+1}$ usando θ_n e $\Delta\theta$ calculado em 5g

(k) Alterar $\overline{W}_n$ com novos valores $\overline{\sigma}_{k+1}$ e $\overline{\theta}_{k+1}$

(l) Colocar X_k, Y_k em $\overline{W}_n$ o qual fica da forma

$$\overline{W}_n = \left[\begin{array}{cccccccccc} u_n & \overline{\sigma}_{k+1} & \overline{\theta}_{k+1} & p_n & \varepsilon_{k+1} & \varepsilon^p_n & c1_n & c2_n & D_n & X_k \quad Y_k \end{array} \right]$$

(m) Calcular F usando σ_k, X_k, Y_k

(n) Se $F > 0$

i. Corretor usando $\overline{W}_n$ completando W_{k+1} o qual fica

$$W_{k+1} = \left[\begin{array}{cccccccccc} u_{k+1} & \sigma_{k+1} & \theta_{k+1} & p_{k+1} & \varepsilon_{k+1} & \varepsilon^p_{k+1} & c1_{k+1} & c2_{k+1} & D_{k+1} & X_k \quad Y_k \end{array} \right]^T$$

ii. Zerar W_{k+1} em X e Y

(o) Senão

i. faz-se

$$W_{k+1} = [u_{k+1} \quad \sigma_{k+1} = \overline{\sigma}_{k+1} \quad \theta_{k+1} = \overline{\theta}_{k+1} \quad p_{k+1} = p_k \quad \varepsilon_{k+1} \quad \varepsilon^p_{k+1} = \varepsilon^p_k \\ c1_{k+1} = c1_k \quad c2_{k+1} = c2_k \quad D_{k+1} = D_k \quad X_k \quad Y_k]^T$$

(p) Fim Se

(q) Calcular X_{k+1} usando $c1_{k+1}, c2_{k+1}, D_{k+1}$ e salvar em W_{k+1} onde $F > 0$

(r) Calcular Y_{k+1} usando D_{k+1}, p_{k+1} e salvar em W_{k+1} onde $F > 0$ o qual fica da forma

$$W_{k+1} = \left[\; u_{k+1} \quad \sigma_{k+1} \quad \theta_{k+1} \quad p_{k+1} \quad \varepsilon_{k+1} \quad \varepsilon^p_{k+1} \quad c1_{k+1} \quad c2_{k+1} \quad D_{k+1} \quad X_{k+1} \quad Y_{k+1} \; \right]^T$$

(s) Calcular erro

(t) niter++

6. }

7. Se $niter < nmax$

(a) Fazer

$$W_{n+1} = [u_{n+1} = u_{k+1} \quad \sigma_{n+1} = \sigma_{k+1} \quad \theta_{n+1} = \theta_{k+1} \quad p_{n+1} = p_{k+1} \quad \varepsilon_{n+1} = \varepsilon_{k+1} \quad \varepsilon^p_{n+1} = \varepsilon^p_{k+1} \\ c1_{n+1} = c1_{k+1} \quad c2_{n+1} = c2_{k+1} \quad D_{n+1} = D_{k+1} \quad X_{n+1} = X_{k+1} \quad Y_{n+1} = Y_{k+1}]^T$$

8. Senão

(a) Erro

9. Fim Se

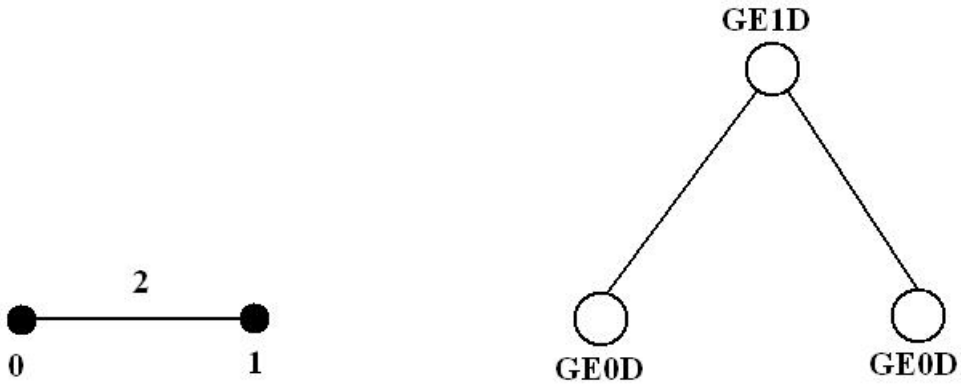
4.1.9 Implementação no MPhyScas

Nesta seção será mostrado como o problema foi implementado no MPhyScas. A representação da geometria na forma de grafo bem como os códigos dos entes geométricos podem ser vistos na Figura 4.4

O problema foi dividido em um Bloco, seis Grupos e vinte Fenômenos. Na Tabela 4.1 pode ser visto a relação entre o Bloco e os seus Grupos e na Tabela 4.2 pode ser visto a relação entre os Grupos e seus Fenômenos.

A descrição de cada Fenômeno pode ser vista na Tabela 4.3.

Em cada simulação, um determinado Fenômeno pode ter quantias ativadas no domínio e no contorno. O Fenômeno Deslocamento possui quantias ativadas no contorno e no domínio.



(a) Geometria e códigos (b) Representação da geometria

Figura 4.4 Geometria do problema unidimensional

Tabela 4.1 Relação entre Bloco e Grupo no problema unidimensional

Códigos do Bloco	Códigos dos Grupos
0	0, 1, 2, 3, 4, 5

Tabela 4.2 Relação entre Grupo e Fenômeno no problema unidimensional

Códigos dos Grupos	Códigos dos Fenômenos
0	0
1	11
2	6
3	1 e 4
4	2, 3, 5, 7, 8, 9, 10
5	12, 13, 14, 15, 16, 17, 18, 19

Tabela 4.3 Descrição dos Fenômenos

Códigos dos Fenômenos	Descrição
0	Fenômeno Deslocamento
1	Fenômeno Endurecimento Isotrópico
2	Fenômeno que representa a variável c_1
3	Fenômeno que representa a variável c_2
4	Fenômeno Endurecimento Cinemático
5	Fenômeno Deformação Plástica
6	Fenômeno Função de Plastificação
7	Fenômeno Tensão
8	Fenômeno Temperatura
9	Fenômeno Deformação Plástica Acumulada
10	Fenômeno Dano
11	Fenômeno Deformação
12	Fenômeno responsável pelo pós-processamento da tensão
13	Fenômeno responsável pelo pós-processamento da deformação
14	Fenômeno responsável pelo pós-processamento da deformação plástica
15	Fenômeno responsável pelo pós-processamento do dano
16	Fenômeno responsável pelo pós-processamento da temperatura
17	Fenômeno responsável pelo pós-processamento da deformação plástica acumulada
18	Fenômeno responsável pelo pós-processamento do endurecimento isotrópico
19	Fenômeno responsável pelo pós-processamento do endurecimento cinemático

Tabela 4.4 Quantias ativadas no contorno - Fenômeno Deslocamento

Ente geométrico	Código da quantia	Quantia	Descrição da quantia
0	0	WFDirichlet	Responsável pela condição de contorno de Dirichlet
1	1	WFNeumann	Responsável pela condição de contorno de Neumann

Na Tabela 4.4 são mostradas as quantias que são ativadas no contorno e na Tabela 4.5 são mostradas as quantias que são ativadas no domínio para o Fenômeno Deslocamento.

Os fenômenos que restam apenas possuem quantias ativadas no domínio. Os Fenômenos Endurecimento Isotrópico, Endurecimento Cinemático, Deformação Plástica, Função de Plastificação, Deformação Plástica Acumulada, Dano, Deformação, além dos Fenômenos responsáveis pelo pós-processamento da Tensão, da Deformação, da Deformação Plástica Acumulada, do Dano, da Temperatura, do Endurecimento Cinemático, do Endurecimento Isotrópico e os Fenômenos responsáveis pelas variáveis c_1 e c_2 só possuem uma quantia ativada no domínio, por esta razão a quantia ativada para cada um desses Fenômenos será mostrada em uma única tabela. Estas quantias podem ser vistas na Tabela 4.6 (todas as quantias possuem código zero).

As quantias ativadas para o Fenômeno Tensão podem ser vistas na Tabela 4.7 e as quantias ativadas para o Fenômeno Temperatura podem ser vistas na Tabela 4.8.

4.2 Modelo de Difusão Bidimensional

Um modelo contínuo para deformação-difusão-dano em sólidos elásticos proposta em [17] também será implementado para validação do MPhyScas (Equações 2.12, 2.13, 2.14, 2.15, 2.16, 2.17 e 2.18).

Para solução deste problema, será necessário uma discretização espacial e uma discretização temporal. A discretização espacial será realizada utilizando o método do elemento finito. O ponto de partida para a discretização espacial consiste em obter a forma fraca das equações. Substituindo a Equação 2.15 na Equação 2.14 e multiplicando a equação resultante pela função teste v e integrando no domínio Ω tem-se que

$$\int_\Omega \dot{c}v \,\mathrm{d}\Omega - \int_\Omega \nabla\cdot(D\nabla c)v \,\mathrm{d}\Omega + \int_\Omega \nabla\cdot(\frac{cD}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S})))v \,\mathrm{d}\Omega - \\ - \int_\Omega \nabla\cdot(\frac{cD}{\alpha}\nabla(\omega\lambda d))v \,\mathrm{d}\Omega = 0 \quad (4.42)$$

sabe-se que

$$\phi\nabla\cdot\mathbf{u} = \nabla\cdot(\Phi\mathbf{u}) - \nabla\Phi^T\cdot\mathbf{u} \quad (4.43)$$

e o teorema do divergente

$$\int_{\partial\Omega}\mathbf{n}^T\cdot\mathbf{u}\,\mathrm{dl} = \int_\Omega \nabla\cdot\mathbf{u}\,\mathrm{d}\Omega \quad (4.44)$$

Usando 4.43 e 4.44 na Equação 4.42 e organizando, tem-se

$$\int_\Omega \dot{c}v \,\mathrm{d}\Omega + \int_\Omega \nabla v^T\cdot(D\nabla c)\,\mathrm{d}\Omega - \int_\Omega \nabla v^T\cdot(\frac{cD}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S})))\,\mathrm{d}\Omega \\ + \int_\Omega \nabla v^T\cdot(\frac{cD}{\alpha}\nabla(\omega\lambda d))\,\mathrm{d}\Omega - \int_{\partial\Omega}\mathbf{n}^T\cdot(D\nabla c)v\,\mathrm{dl} + \int_{\partial\Omega}\mathbf{n}^T\cdot(\frac{cD}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S})))v\,\mathrm{dl} \\ - \int_{\partial\Omega}\mathbf{n}^T\cdot(\frac{cD}{\alpha}\nabla(\omega\lambda d))v\,\mathrm{dl} = 0 \quad (4.45)$$

Organizando os termos do contorno em uma só integral, a Equação 4.45 fica da forma

$$\int_\Omega \dot{c}v \, \mathrm{d}\Omega + \int_\Omega \nabla v^T \cdot [D(\nabla c - \frac{c}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S}) - \omega\lambda d))] \, \mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{n}^T \cdot [D(-\nabla c + \frac{c}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S}) - \omega\lambda d))]v \, \mathrm{dl} = 0 \quad (4.46)$$

que é a forma fraca para o problema da difusão.

Multiplicando a Equação 2.12 pela função teste $\mathbf{w}$ e integrando no domínio Ω obtém-se

$$\int_\Omega (\nabla \cdot \mathbf{S}) \cdot \mathbf{w} \, \mathrm{d}\Omega + \int_\Omega \mathbf{b} \cdot \, \mathrm{d}\Omega = 0 \quad (4.47)$$

Sabendo que

$$(\nabla \cdot \mathbf{S}) \cdot \mathbf{w} = \nabla \cdot (\mathbf{S}\mathbf{w}) - \mathbf{S} : \nabla \mathbf{w} \quad (4.48)$$

Usando a Equação 4.44 e a Equação 4.48 na Equação 4.47, resulta em

$$\int_\Omega \mathbf{S} : \nabla \mathbf{w} \, \mathrm{d}\Omega - \int_\Omega \mathbf{b} \cdot \mathbf{w} \, \mathrm{d}\Omega - \int_{\partial\Omega} \mathbf{n}^T \cdot \mathbf{S}\mathbf{w} \, \mathrm{dl} = 0 \quad (4.49)$$

que é a forma fraca para o problema de elasticidade.

Na obtenção da forma fraca para a equação da evolução do dano, a função $\langle \, \rangle$ será desconsiderada. Multiplicando a Equação 2.16 pela função teste r e integrando no domínio Ω obtém-se

$$\int_\Omega \dot{d}r \, \mathrm{d}\Omega - \int_\Omega \frac{1}{b}\hat{\varepsilon}r \, \mathrm{d}\Omega + \int_\Omega \frac{\omega}{b}(1 - \lambda c)r \, \mathrm{d}\Omega - \int_\Omega \frac{\kappa}{b}\triangle d r \, \mathrm{d}\Omega = 0 \quad (4.50)$$

Sabe-se que

$$r\nabla \cdot \mathbf{u} = \nabla \cdot (r\mathbf{u}) - \mathbf{u} \cdot \nabla r \quad (4.51)$$

Usando a Equação 4.44 e a Equação 4.51 na Equação 4.50, resulta em

$$\int_\Omega \dot{d}r \, \mathrm{d}\Omega - \int_\Omega \frac{1}{b}\hat{\varepsilon}r \, \mathrm{d}\Omega + \int_\Omega \frac{\omega}{b}(1 - \lambda c)r \, \mathrm{d}\Omega + \int_\Omega \nabla d \cdot \nabla(\frac{\kappa}{b}r) \, \mathrm{d}\Omega - \int_{\partial\Omega} \mathbf{n}^T \cdot \frac{\kappa}{b} r \nabla d \, \mathrm{dl} = 0 \quad (4.52)$$

que é a forma fraca para a equação de evolução do dano.

Devido a complexidade do problema, a parte referente ao dano não será considerado neste momento. Com isso, as equações que governam o problema são reduzidas a

$$\nabla \cdot \mathbf{S} + \mathbf{b} = 0 \quad (4.53)$$

$$\mathbf{S} = \bar{\lambda}\mathrm{tr}(\mathbf{E})\mathbf{I} + 2\bar{\mu}\mathbf{E} - e(3\bar{\lambda} + 2\bar{\mu})(c - c_0)\mathbf{I} \quad (4.54)$$

$$\nabla \cdot \mathbf{J} + \dot{c} = 0 \quad (4.55)$$

$$\mathbf{J} = -D(\nabla c - \frac{c}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S}))) \quad (4.56)$$

com as formas fracas dadas por

$$\int_\Omega \mathbf{S} : \nabla \mathbf{w} \, \mathrm{d}\Omega - \int_\Omega \mathbf{b} \cdot \mathbf{w} \, \mathrm{d}\Omega - \int_{\partial\Omega} \mathbf{n}^T \cdot \mathbf{S}\mathbf{w} \, \mathrm{dl} = 0 \quad (4.57)$$

$$\int_\Omega \dot{c}v \, \mathrm{d}\Omega + \int_\Omega \nabla v^T \cdot [D(\nabla c - \frac{c}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S})))] \, \mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{n}^T \cdot [D(-\nabla c + \frac{c}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S})))]v \, \mathrm{dl} = 0 \quad (4.58)$$

O método de solução utilizado constitui-se de dois problemas. O primeiro a ser resolvido será o problema de elasticidade (Equação 4.53), onde a difusão do soluto será considerada congelada. O segundo será o problema de difusão (Equação 4.55), o qual utilizará como dado a solução do problema de elasticidade.

4.2.1 Solução do Problema de Elasticidade

Neste seção, será descrito os procedimentos utilizados na solução do problema de elasticidade. A tensão elástica total $\mathbf{S}$, como pode ser visto na Equação 4.54, é decomposta em duas parcelas (parcela elástica sem difusão e a parcela devido a difusão). A parcela referente a tensão elástica sem difusão será identificada pela variável $\sigma_\mathbf{e}$ e a parcela devido a difusão será identificada pela variável $\sigma_\mathbf{c}$, onde

$$\sigma_\mathbf{e} = \bar{\lambda}\,\mathrm{tr}(\mathbf{E})\mathbf{I} + 2\bar{\mu}\mathbf{E} \quad (4.59)$$

$$\sigma_\mathbf{c} = -e(3\bar{\lambda} + 2\bar{\mu})(c - c_0)\mathbf{I} \quad (4.60)$$

A Equação 4.54 pode ser reescrita na forma

$$\mathbf{S} = \sigma_\mathbf{e} + \sigma_\mathbf{c} \quad (4.61)$$

Substituindo a Equação 4.61 na Equação 4.57, resulta em

$$\int_\Omega \sigma_\mathbf{e} : \nabla\mathbf{w} \, \mathrm{d}\Omega = \int_\Omega \mathbf{b}\cdot\mathbf{w} \, \mathrm{d}\Omega - \int_\Omega \sigma_\mathbf{c} : \nabla\mathbf{w} \, \mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{n}^T \cdot (\sigma_\mathbf{e} + \sigma_\mathbf{c})\mathbf{w} \, \mathrm{dl} \quad (4.62)$$

Conforme mencionado ateriormente, o problema será discretizado pelo método do elemento finito. O esquema adotado para solução será o de resolver um problema de elasticidade a partir do qual serão obtidos os deslocamentos nodais de cada elemento finito. A partir dos deslocamentos nodais, será realizado o cálculo da deformação em cada elemento. De posse da deformação em cada elemento, o cálculo das tensões nos elementos será realizado. Com as tensões em cada elemento, será realizada a recuperação das tensões nodais (tensão elástica sem difusão) e a partir desta obtém-se a tensão elástica total em cada nó. O termo referente a difusão na tensão elástica total, contribuirá como uma força no problema de elasticidade, como pode ser visto na Equação 4.62.

O tensor de deformação é dado por

$$\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \quad (4.63)$$

onde no plano este tensor pode ser escrito na forma vetorial, ou seja

$$\varepsilon(\mathbf{u}) = \begin{pmatrix} \varepsilon_{xx}(\mathbf{u}) \\ \varepsilon_{yy}(\mathbf{u}) \\ \varepsilon_{xy}(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \end{pmatrix} \quad (4.64)$$

No plano, o tensor de tensão $\sigma_{\mathbf{e}}$ também pode ser escrito na forma vetorial, ou seja

$$\sigma_{\mathbf{e}} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{pmatrix} \tag{4.65}$$

A tensão $\sigma_{\mathbf{e}}$, na forma vetorial, e a deformação ε se relacionam através da equação constitutiva

$$\sigma_{\mathbf{e}} = \mathbf{C}\varepsilon(\mathbf{u}) \tag{4.66}$$

onde $\mathbf{C}$ é a matriz de elasticidade. A matriz de elasticidade é definida por

$$\mathbf{C} = \begin{bmatrix} \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} & \frac{E\nu}{(1+\nu)(1-2\nu)} & 0 \\ \frac{E\nu}{(1+\nu)(1-2\nu)} & \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} & 0 \\ 0 & 0 & 2G \end{bmatrix} \tag{4.67}$$

quando se tem estado plano de deformações e

$$\mathbf{C} = \begin{bmatrix} \frac{E}{(1-\nu^2)(1-2\nu)} & \frac{\nu}{(1-\nu^2)} & 0 \\ \frac{\nu}{(1-\nu^2)} & \frac{E}{(1-\nu^2)(1-2\nu)} & 0 \\ 0 & 0 & 2G \end{bmatrix} \tag{4.68}$$

quando se tem estado plano de tensões. Aqui, E é o módulo de young, ν é o coeficiente de poison e G é dado por

$$G = \frac{E}{2(1+\nu)} \tag{4.69}$$

É possível relacionar a deformação na forma vetorial com o deslocamento através de

$$\varepsilon(\mathbf{u}) = \mathbf{D}\mathbf{u} \tag{4.70}$$

onde

$$\mathbf{D} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \tag{4.71}$$

A partir da Equação 4.71 pode-se escrever a Equação 4.66 como

$$\sigma_{\mathbf{e}} = \mathbf{C}\mathbf{D}\mathbf{u} \tag{4.72}$$

O produto interno $\sigma_{\mathbf{e}} : \nabla\mathbf{w}$ na Equação 4.62 é dado por

$$\sigma_{\mathbf{e}} : \nabla\mathbf{w} = \mathrm{tr}(\nabla\mathbf{w}^T \sigma_{\mathbf{e}}) \tag{4.73}$$

que para o caso bidimensional, resulta em

$$\sigma_{\mathbf{e}} : \nabla\mathbf{w} = \sigma_{xx}\frac{\partial w_x}{\partial x} + \tau_{xy}\left(\frac{\partial w_x}{\partial y} + \frac{\partial w_y}{\partial x}\right) + \sigma_{yy}\frac{\partial w_y}{\partial y} \tag{4.74}$$

Utilizando a Equação 4.64 na Equação 4.74, então

$$\sigma_{\mathbf{e}} : \nabla \mathbf{w} = \sigma_{xx}\varepsilon_{xx}(\mathbf{w}) + \tau_{xy}(2\varepsilon_{xy}(\mathbf{w})) + \sigma_{yy}\varepsilon_{yy}(\mathbf{w}) \tag{4.75}$$

Redefinindo a Equação 4.64 da seguinte forma

$$\varepsilon(\mathbf{u}) = \begin{pmatrix} \varepsilon_{xx}(\mathbf{u}) \\ \varepsilon_{yy}(\mathbf{u}) \\ 2\varepsilon_{xy}(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} \varepsilon_{xx}(\mathbf{u}) \\ \varepsilon_{yy}(\mathbf{u}) \\ \gamma_{xy}(\mathbf{u}) \end{pmatrix} \tag{4.76}$$

então pode-se escrever a Equação 4.75 como sendo

$$\sigma_{\mathbf{e}} : \nabla \mathbf{w} = \varepsilon(\mathbf{w})^T \cdot \sigma_{\mathbf{e}} \tag{4.77}$$

Para se utilizar a Equação 4.76, as Equações 4.67 e 4.68 devem ser modificadas substituindo $2G$ por G. Utilizando as Equações 4.72 e 4.77 na Equação 4.62, então

$$\int_{\Omega} (\mathbf{Dw})^T \cdot \mathbf{CDu} = \int_{\Omega} \mathbf{b} \cdot \mathbf{w} \, \mathrm{d}\Omega - \int_{\Omega} \sigma_{\mathbf{c}} : \nabla \mathbf{w} \, \mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{n}^T \cdot (\sigma_{\mathbf{e}} + \sigma_{\mathbf{c}})\mathbf{w} \, \mathrm{dl} \tag{4.78}$$

que corresponde a forma fraca para o problema de elasticidade em função do deslocamento.

Neste problema, em cada parte do contorno, é possível prescrever as seguintes condições de contorno:

- Condição de Dirichlet: Podem ser prescrito deslocamentos na direção normal e na direção tangencial;
- Condição de Neumann: Podem ser prescrita tensões na direção normal e na direção tangencial;
- Condição Mista: Também existe a possibilidade de se prescrever tensão na direção normal e deslocamento na direção tangencial ou vice-versa.

O processo de discretização pelo método do elemento finito, envolve a definição de funções as quais serão utilizadas para interpolação do deslocamento (funções de forma). As funções de forma serão definidas em um elemento de referência, elemento onde serão realizadas as integrações da Equação 4.78. Na Figura 4.5 pode ser visto como é feito o mapeamento dos nós no elemento real para o elemento de referência e vice-versa.

As funções de forma no elemento de referência são definidas como

$$\hat{\psi}_1(\xi, \eta) = 1 - \xi - \eta \tag{4.79}$$

$$\hat{\psi}_2(\xi, \eta) = \xi \tag{4.80}$$

$$\hat{\psi}_3(\xi, \eta) = \eta \tag{4.81}$$

e a função que mapeia o elemento de referência no elemento real é definida por

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{B} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \mathbf{x_i} \tag{4.82}$$

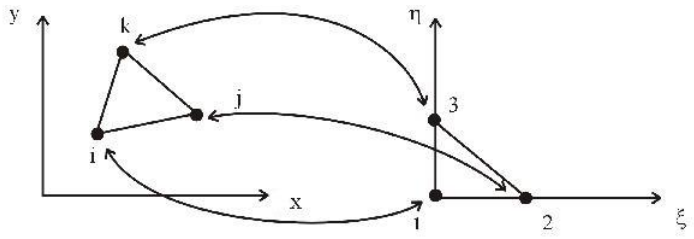


Figura 4.5 Mapeamento entre o elemento de referência e o elemento real

onde

$$\mathbf{B} = [\mathbf{x_j} - \mathbf{x_i} | \mathbf{x_k} - \mathbf{x_i}] \tag{4.83}$$

sendo $\mathbf{x_i}$, $\mathbf{x_j}$ e $\mathbf{x_k}$ são os vetores de coordenadas dos nós i, j e k respectivamente. Utilizando as funções de forma definidas e a transformação descrita, pode-se calcular a contribuição de cada elemento para o sistema global. A contribuição da primeira integral na Equação 4.78 é dada por

$$\mathbf{K^e} = \frac{\det(\mathbf{B})}{2} \mathbf{D_\psi}^T \mathbf{C} \mathbf{D_\psi} \tag{4.84}$$

onde

$$\mathbf{D_\psi} = \begin{bmatrix} \mathbf{G_\psi}(1,1) & 0 & \mathbf{G_\psi}(1,2) & 0 & \mathbf{G_\psi}(1,3) & 0 \\ 0 & \mathbf{G_\psi}(2,1) & 0 & \mathbf{G_\psi}(2,2) & 0 & \mathbf{G_\psi}(2,3) \\ \mathbf{G_\psi}(2,1) & \mathbf{G_\psi}(1,1) & \mathbf{G_\psi}(2,2) & \mathbf{G_\psi}(1,2) & \mathbf{G_\psi}(2,3) & \mathbf{G_\psi}(1,3) \end{bmatrix} \tag{4.85}$$

sendo

$$\mathbf{G_\psi} = \mathbf{B}^{-T} \mathbf{A_\psi} \tag{4.86}$$

com

$$\mathbf{A_\psi} = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.87}$$

As contribuição para o vetor força são dadas por (segunda e terceira inegral na Equação 4.78, respectivamente)

$$\mathbf{f_1^e} = \int_e \hat{\psi}_{\hat{i}}(\xi,\eta) \hat{\mathbf{b}}(\xi,\eta) \det(\mathbf{B}) \, \mathrm{d}\xi \mathrm{d}\eta \;\; \text{para } \hat{i} = 1,2,3 \tag{4.88}$$

$$\mathbf{f_2^e} = \frac{e(3\bar{\lambda} + 2\bar{\mu}) \det(\mathbf{B})}{6} \mathbf{K_c} \begin{bmatrix} c_1 - c_0 \\ c_2 - c_0 \\ c_3 - c_0 \end{bmatrix} \tag{4.89}$$

onde

$$\mathbf{K_c} = \begin{bmatrix} \mathbf{k_1}(1) & \mathbf{k_1}(1) & \mathbf{k_1}(1) \\ \mathbf{k_1}(2) & \mathbf{k_1}(2) & \mathbf{k_1}(2) \\ \mathbf{k_2}(1) & \mathbf{k_2}(1) & \mathbf{k_2}(1) \\ \mathbf{k_2}(2) & \mathbf{k_2}(2) & \mathbf{k_2}(2) \\ \mathbf{k_3}(1) & \mathbf{k_3}(1) & \mathbf{k_3}(1) \\ \mathbf{k_3}(2) & \mathbf{k_3}(2) & \mathbf{k_3}(2) \end{bmatrix} \tag{4.90}$$

sendo

$$\mathbf{k_1} = \mathbf{B}^{-T} \begin{pmatrix} -1 \\ -1 \end{pmatrix} \tag{4.91}$$

$$\mathbf{k_2} = \mathbf{B}^{-T} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{4.92}$$

$$\mathbf{k_3} = \mathbf{B}^{-T} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.93}$$

onde c_1, c_2 e c_3 corresponde ao valor da concentração de soluto nos nós 1, 2 e 3 respectivamente e c_0 é a concentração de referência. A integral na Equação 4.88 será calculada numericamente, uma vez que a força de corpo **b** pode depender de x e y. A partir dessas contribuições, pode-se montar o sistema algébrico para o cálculo dos deslocamentos nodais

$$\mathbf{Ku} = \mathbf{f} \tag{4.94}$$

A solução do sistema algébrico na Equação 4.94 fornece os deslocamentos nodais. A partir dos deslocamentos nodais, calcula-se a deformação em cada elemento finito utilizando a Equação 4.63. A deformação é colocada na forma vetorial mostrada na Equação 4.64. A partir da deformação na forma vetorial, calcula-se a tensão elástica em cada elemento, na forma vetorial, utilizando a Equação 4.66. Conhecendo a tensão elástica em cada elemento, pode-se realizar a recuperação das tensões nodais. Considere uma parte da malha geométrica mostrada na Figura 4.6 a tensão nodal claculada no nó i será dada por

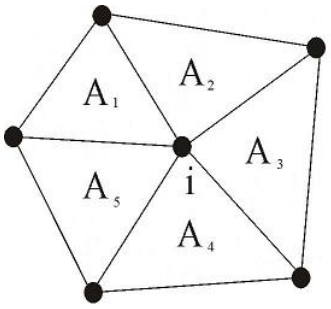


Figura 4.6 Recuperação da tensão elástica nodal

$$\sigma_{\mathbf{e}noi} = \frac{\sum_j \sigma_{\mathbf{e}\,j} A_j}{\sum_j A_j} \tag{4.95}$$

onde $\sigma_{\mathbf{e}\,j}$ é a tensão elástica do elemento j e A_j é a área do elemento j. Com as tensões elásticas nodais recuperadas, pode-se calcular o traço da tensão elástica total em cada nó, isto é feito através de

$$\mathrm{tr}(\mathbf{S})_{noi} = \mathrm{tr}(\sigma_{\mathbf{e}noi}) - 2e(3\bar{\lambda} + 2\bar{\mu})(c_{noi} - c_0) \tag{4.96}$$

onde $\sigma_{\mathbf{e}noi}$ é a tensão elástica no nó i e c_{noi} é o valor da concentração no nó i. Com isso, o problema de elasticidade é finalizado.

4.2.2 Solução do Problema de Difusão

O processo de discretização espacial para o problema de difusão é feito de forma semelhante ao processo de discretização do problema de elasticidade. Aqui, serão utilizadas as mesmas funções de forma e a mesma transformação entre o elemento de referência e o elemento real. A partir da forma fraca na Equação 4.58 calcula-se as seguintes contribuições em cada elemento. A contribuição referente a primeira integral na Equação 4.58 é dada por

$$\mathbf{M^e} = \frac{\det(\mathbf{B})}{24}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{4.97}$$

As contribuições referentes a segunda integral na Equação 4.58 são dadas por

$$\mathbf{K_1^e} = \frac{D\det(\mathbf{B})}{2}{\mathbf{A}_\psi}^T\mathbf{B}^{-1}\mathbf{B}^{-T}\mathbf{A}_\psi \tag{4.98}$$

$$\mathbf{K_2^e} = -\frac{De\det(\mathbf{B})}{6\alpha}{\mathbf{A}_\psi}^T\mathbf{B}^{-1}\mathbf{B}^{-T}\mathbf{A}_\psi\begin{bmatrix} tr_1 & tr_1 & tr_1 \\ tr_2 & tr_2 & tr_2 \\ tr_3 & tr_3 & tr_3 \end{bmatrix} \tag{4.99}$$

onde tr_1, tr_2 e tr_3, correspondem ao traço da tensão elástica total dos nós 1, 2 e 3 respectivamente. Com essas contribuições, pode-se montar o sistema algébrico

$$\mathbf{M\dot{c}} + \mathbf{Kc} = 0 \tag{4.100}$$

É necessário uma discretização temporal no sistema algébrico da Equação 4.100. Essa discretização será feita utilizando o método de Euler implícito [31], então

$$\dot{\mathbf{c}} = \frac{\mathbf{c^{n+1}} - \mathbf{c^n}}{\Delta t} \tag{4.101}$$

onde $\mathbf{c^{n+1}}$ é a concentração de soluto do tempo atual, $\mathbf{c^n}$ é a concentração de soluto no tempo anterior e Δt é o incremento de tempo. Substituindo a Equação 4.101 na Equação 4.100 o sistema algébrico fica na forma

$$\left(\frac{\mathbf{M}}{\Delta t} + \mathbf{K}\right)\mathbf{c^{n+1}} = \frac{\mathbf{M}}{\Delta t}\mathbf{c^n} \tag{4.102}$$

Resolvendo o sistema na Equação 4.102 obtém-se o concentração de soluto em cada nó no instante de tempo atual e com isso o problema está finalizado.

4.2.3 Algoritmo de Solução

O algoritmo para solução desse problema é mostrado a seguir:

1. Calcular a matriz de elasticidade
2. Calcular os deslocamentos nodais

3. Calcular a deformação
4. Calcular tensão nos elementos
5. Recuperar a tensão nodal
6. Calcular tr($\mathbf{S}$)
7. Calcular difusão

4.2.4 Implementação no MPhyScas

Neste seção, será apresentado como o problema foi implementado no MPhyScas. O problema foi dividido em um Bloco, em seis Grupos e seis Fenômenos. Na Tabela 4.9 pode ser visto a relação entre o Bloco e os seus Grupos e na Tabela 4.10 pode ser visto a relação entre os Grupos e seus Fenômenos.

A descrição de cada Fenômeno pode ser vista na Tabela 4.11.

Neste problema foram realizadas várias simulações, onde as quantias ativadas para cada fenômeno variaram em cada simulação, por este motivo as tabelas com as quantias ativadas para cada fenômeno foram deixadas para o capítulo de soluções numéricas. Na implementação deste modelo, apenas foram inseridos na Bliblioteca Estática as **WeakForm**s para produção das quantias necessárias para este problema bem como os métodos e algoritmos aqui utilizados, permanecendo a estrutura do MPhyScas inalterada.

4.3 Gerador de Malha no MPhyScas

A geração de malha corresponde a um dos processos envolvidos na discretização de um problema utilizando o método do elemento finito. A geração de malha é um pré-requisito essencial para qualquer simulação envolvendo o método do elemento finito, além disso, este processo pode ser visto como um gargalo em processos numéricos, no sentido que uma falha no processo de geração resulta em uma subsequente impossibilidade na simulação numérica.

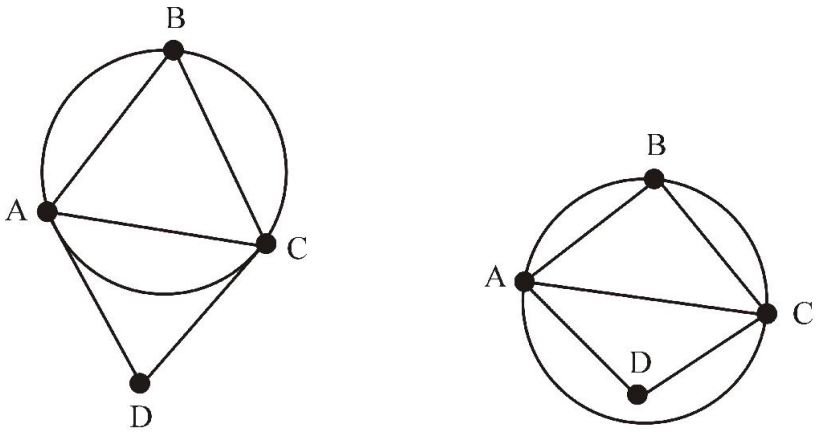
Em [32] uma variedade de processos de geração de malha são descritos. O método escolhido neste trabalho se baseia na triangulação Delaunay. Essa triangulação é caracterizada pelo fato de que o círculo circunscrito a um triângulo qualquer pertencente a tringulação não possui pontos internos a ele, na Figura 4.7 pode ser visto um exemplo de triagulação Delauney e outra triangulação que não é Delauney.

A idéia principal deste seção é de apresentar os passos do gerador de malha bidimensional que foi implementado, portanto os detalhes teóricos sobre a triangulação Delauney serão deixados de lado. A geometria será representada na forma de grafo, conforme foi mencionado no capítulo 3. Um exemplo de representação da geometria pode ser visto na Figura 4.8.

Cada nó do grafo de geometria recebe o nome de **GeomEntity**. Os **GeomEntities** de menor dimensão (pontos) recebem o nome de **GE0D2D**, as curvas recebem o nome de **GE1D2D** (contorno) e as superfícies recebem o nome de **GE2D2D**.

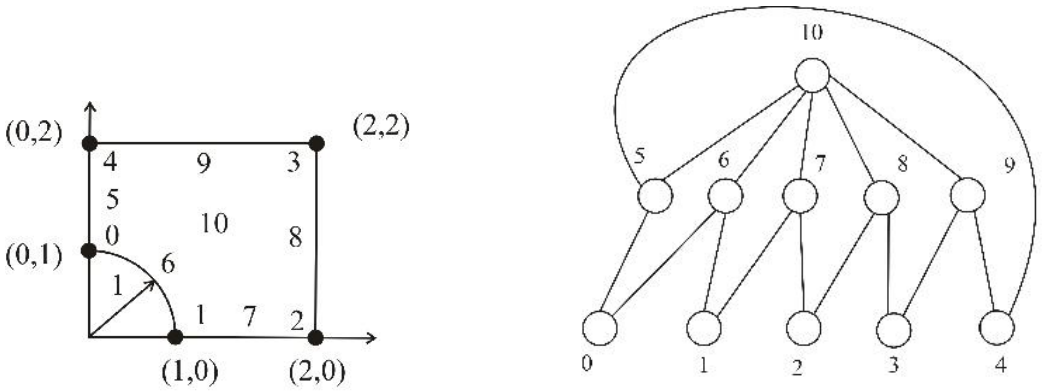
Tabela 4.5 Quantias ativadas no domínio - Fenômeno Deslocamento

Código da quantia	Quantia	Descrição da quantia
0	WFStiffness	Responsável pela montagem da matriz de rigidez
1	WFLoad	Responsável pela montagem da força de corpo
2	WFTempLoad	Responsável pela montagem da força devido a temperatura
3	WFPlasDefLoad	Responsável pela montagem da força devido a deformação plástica



(a) Triangulação Delauney (b) Triangulação não Delauney

Figura 4.7 Exemplos de triangulação



(a) Exemplo de geometria (b) Representação da geometria

Figura 4.8 Exemplo de representação da geometria

Tabela 4.6 Quantias ativadas no domínio para os fenômenos com apenas uma quantia

Código do Fenômeno	Quantia	Descrição da quantia
1	WFIsoHar	Responsável pelo cálculo do endurecimento isotrópico
2	WFKinC1	Responsável pelo cálculo da taxa da variável c_1
3	WFKinC2	Responsável pelo cálculo da taxa da variável c_2
4	WFKinHar	Responsável pelo cálculo do endurecimento cinemático
5	WFPlasDef	Responsável pelo cálculo da taxa da deformação plástica
6	WFPlasTes	Responsável pelo cálculo da função de plastificação
9	WFCumulPlas	Responsável pelo cálculo da taxa da deformação plástica acumulada
10	WFDamage	Responsável pelo cálculo da taxa do dano
11	WFDeform	Responsável pelo cálculo da deformação
12	WFPosStress	Responsável pelo pós-processamento da tensão
13	WFPosDeformation	Responsável pelo pós-processamento da deformação
14	WFPosPlasDef	Responsável pelo pós-processamento da deformação plástica
15	WFPosDamage	Responsável pelo pós-processamento do dano
16	WFPosTemp	Responsável pelo pós-processamento da temperatura
17	WFPosCPDeform	Responsável pelo pós-processamento da deformação plástica acumulada
18	WFPosIsoHar	Responsável pelo pós-processamento do endurecimento isotrópico
19	WFPosKinHar	Responsável pelo pós-processamento do endurecimento cinemático

Tabela 4.7 Quantias ativadas no domínio - Fenômeno Tensão

Código da quantia	Quantia	Descrição da quantia
0	WFStrVar	Responsável pelo cálculo de $\Delta\sigma$
1	WFStress	Responsável pelo cálculo da taxa da tensão

Tabela 4.8 Quantias ativadas no domínio - Fenômeno Temperatura

Código da quantia	Quantia	Descrição da quantia
0	WFTempVar	Responsável pelo cálculo de $\Delta\theta$
1	WFTemperature	Responsável pelo cálculo da taxa da temperatura

Tabela 4.9 Relação entre Bloco e Grupo no problema bidimensional

Códigos do Bloco	Códigos dos Grupos
0	0, 1, 2, 3, 4, 5

Tabela 4.10 Relação entre Grupo e Fenômeno no problema bidimensional

Códigos dos Grupos	Códigos dos Fenômenos
0	0
1	1
2	2
3	3
4	4
5	5

Tabela 4.11 Descrição dos Fenômenos

Códigos dos Fenômenos	Descrição
0	Fenômeno Deslocamento
1	Fenômeno Deformação
2	Fenômeno Matriz de Elasticidade
3	Fenômeno Tensão
4	Fenômeno Difusão
5	Fenômeno Pós-Processamento

Ao receber o grafo da geometria, o gerador de malha deve gerar a malha iniciando geração de malha dos entes geométricos de menor dimensão estrutural até os entes de maior dimensão estrutural, ou seja, primeiro é gerada a malha dos **GE0D2D**, em seguida é gerada a malha dos **GE1D2D** (sem alterar a malha dos **GE0D2D**) e por fim é gerada a malha do **GE2D2D** (sem alterar a malha dos **GE0D2D**). A geração de malha de um ente de maior dimensão não deve modificar a malha de um ente de menor dimensão pois estas malhas podem estar sendo compartilhadas por fenômenos diferentes. A malha do **GE2D2D** é armazenada em uma árvore binária entrelaçada onde a função de ordenamento é dada pela "medida de qualidade" do triângulo (esta medida pode ser de várias formas, um exemplo de medida pode ser visto na Equação 4.104).

A geração de malha dos **GE0D2D** consiste apenas na inclusão de pontos com as mesmas coordenadas destes entes geométricos. A geração de malha dos **GE1D2D**(os pontos da malha dos **GE0D2D** devem ser utilizados na geração desta malha) consiste na inclusão de pontos ao longo destas curvas (as arestas ligando esses pontos só serão criadas na geração da malha da superfície). O quantidade de pontos que deverá ser incluído em uma curva será especificado por uma função (por exemplo, distância em relação a um ponto, distância em relação ao segmento de reta). Esta função é chamada de densidade do ponto. No exemplo que está sendo apresentado (geração de malha da geometria na Figura 4.8) está sendo utilizada como função densidade do ponto, a distância em relação ao ponto de coordenada $(0,0)$, esta função encontra-se na Equação 4.103

$$d = d_0 e^{(\sqrt{(x-x_0)^2+(y-y_0)^2})} \tag{4.103}$$

onde $d_0 = 0.005$, d é o valor da densidade do ponto, x e y são as coordenadas do ponto que se quer calcular a densidade e x_0 e y_0 são as coordenadas do ponto a qual se calcula a distância, neste exemplo $x_0 = 0$ e $y_0 = 0$.

Depois de criados todos os pontos do contorno inicia-se o processo de geração da malha do **GE2D2D**. O primeiro passo consiste na construção do *BoundingBox*. O *BoundingBox*, neste caso, corresponde a um quadrado que envolve todos os pontos criados, este quadrado será representado pela sua diagonal. Esta diagonal é criada pegando-se a maior coordena x e a maior coordenada y dentre os pontos criados para formar um extremo da diagonal e a menor coordena x e a menor coordenada y dentre os pontos criados para formar o outro extremo da diagonal. A representação do *BoundingBox* para o exemplo pode ser visto na Figura 4.9.

Após a criação do *BoundingBox* cria-se a malha inicial, esta malha é formada por dois triângulos e é construída alongando-se *BoundingBox*, na Figura 4.10 pode ser vista a malha inicial para o exemplo considerado.

A partir da malha inicial, inicia-se o processo de inclusão dos pontos do contorno. A inclusão é feita ponto por ponto. Para cada ponto incluído, deve-se primeiro achar o triângulo afetado (ponto incluído deve está dentro deste triângulo). A procura pelo triângulo afetado é feita da seguinte forma:

1. Pega-se o último triângulo afetado (caso não exista, o triângulo com pior "medida de qualidade" é utilizado como último triângulo afetado) e verifica-se se este continua sendo o triângulo afetado, caso afirmativo a procura é finalizada, caso contrário seguir para 2;

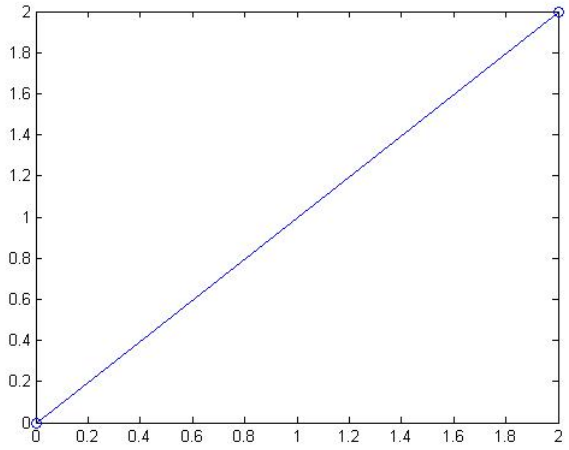


Figura 4.9 *BoundingBox* para o exemplo

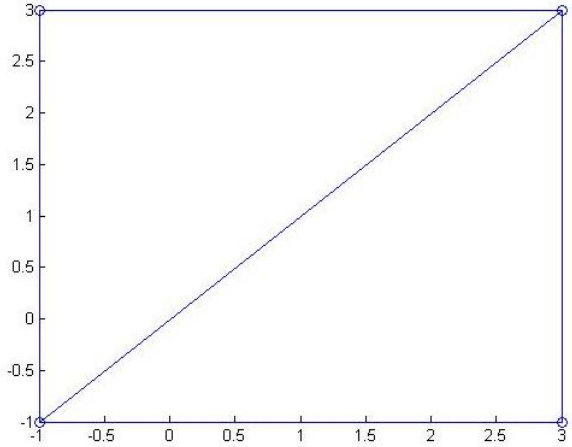


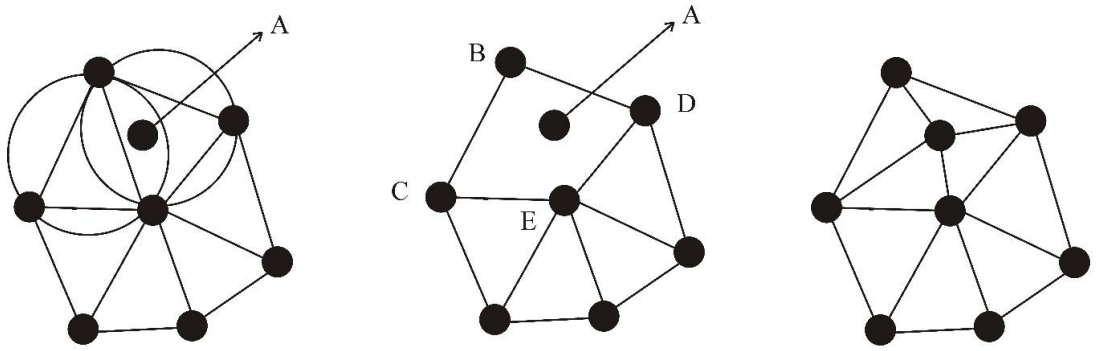
Figura 4.10 Malha inicial para o exemplo

2. Cria-se um segmento de reta entre o baricentro do triângulo afetado e ponto incluído (SEGMENTO1);

3. Pega-se a aresta do triângulo afetado (ARESTA1) que é cortada por SEGMENTO1;

4. Pega-se o outro triângulo (TRIANG1) do qual ARESTA1 faz parte e verifica-se se este triângulo é o triângulo afetado, em caso afirmativo a procura é finaliza caso contrário coloca-se TRIANG1 como último triângulo afetado e retorna-se para 2

Após o triângulo afetado ser encontrado, este triângulo é destruído, no entanto suas arestas são coletadas e inseridas em uma lista circular. Os triângulos vizinhos ao triângulo afetado, cujo o círculo circunscrito contém o ponto a ser inserido, também devem ser destruídos e suas arestas devem ser coletadas e inseridas na lista circular. Esta lista circular representa um polígono onde os novos triângulos devem ser criados. Estes triângulos terão agora como um de seus vértices o ponto que foi inserido. Este processo pode ser visto na Figura 4.11. Na Figura 4.11(a), pode ser visto a triangulação Delauney e o novo ponto a ser iserido, o ponto A, onde são mostrados apenas os círculos circunscritos dos triângulos afetados. Na Figura 4.11(b) pode ser visto o polígono BCDE formado pelas arestas dos triângulos destruídos e, por fim na Figura 4.11 pode ser vista a nova triangulação.



(a) Triangulação com o ponto a ser incluído

(b) Polígono formado pelas arestas

(c) Nova triangulação

Figura 4.11 Exemplos de inclusão de pontos

Este processo é repetido até que todos os pontos do contorno sejam inseridos. Na Figura 4.12 pode ser visto a malha com todos os pontos do contorno inserido para o exemplo considerado.

O próximo passo consiste na verificação do contorno. A verificação é feita pegando-se dois pontos consecutivos do contorno e então é verificado se existe uma aresta na malha ligando os dois pontos. Caso esta aresta não exista deve-se realizar o processo de recuperação do contorno, este processo utiliza o processo de *EdgeFlip*. O processo de *EdgeFlip* é mostrado na Figura 4.13.

A recuperação do contorno é mostrada na Figura 4.14. Na Figura 4.14(a) a aresta AB faz parte do contorno, no entanto este contorno foi violado. Deve-se então, encontrar uma

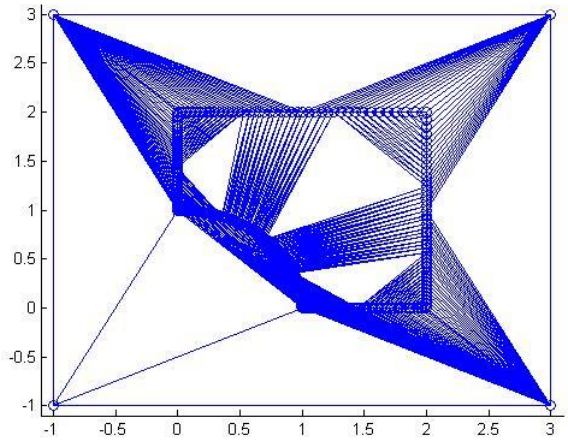
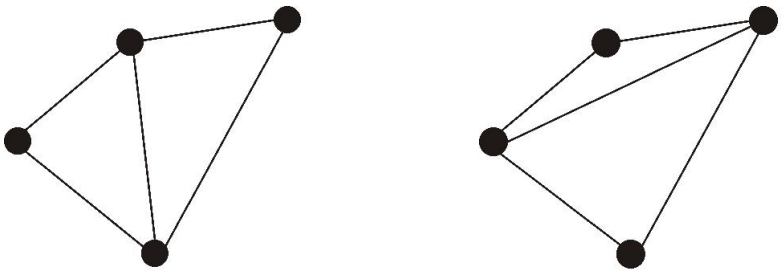


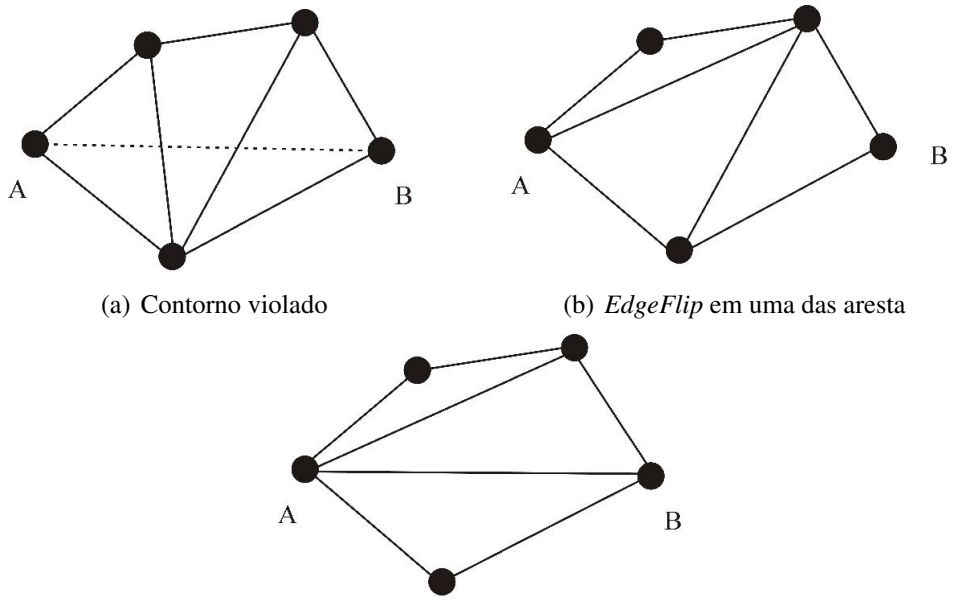
Figura 4.12 Malha com os pontos do contorno para o exemplo



(a) Antes de realizar *EdgeFlip*

(b) Após a realização do *EdgeFlip*

Figura 4.13 Processo de *EdgeFlip*



(a) Contorno violado

(b) *EdgeFlip* em uma das aresta

(c) *EdgeFlip* na outra aresta

Figura 4.14 Recuperação do contorno

das aresta que é interceptada pela aresta AB e então realizar o *EdgeFlip* nesta aresta (Figura 4.14(a)). Este processo é repetido até que o contorno não esteja mais violado (Figura 4.14(a)).

O próximo passo consiste na classificação dos triângulos como internos e externos. Esta calssificação é realizada percorrendo-se o contorno no sentido anti-horário. Os triângulos que estão a direita do contorno são marcados como externos e os triângulos a esquerda do contorno são marcados como interno. Na Figura 4.15 pode ser visto a triangulação apenas com os triângulos marcados como internos para o exemplo considerado.

O próximo passo consiste no refinamento da malha. Este processo consiste em incluir pontos internos a malha. A inclusão de pontos de penderá da "medida de qualidade" de cada elemento. A inclusão de pontos é feita recursivamente até que o triângulo com pior "medida de qualidade" ("pior triângulo") atinja um valor pré-estabelecido (aqui a inclusão de pontos é feita de forma semelhante a inclusão de pontos do contorno). O ponto escolhido para inclusão corresponde ao baricentro do triângulo com pior "medida de qualidade". Para o exemplo considerado, a "medida de qualidade" é dado pela Equação 4.104

$$Q = \sqrt{3}R \tag{4.104}$$

onde Q é a medida de qualidade e R é o raio do círculo circunscrito. Neste exemplo, o valor estabelecido para parada é mostrado na Equação 4.105

$$Q_w \leq 1.5d \tag{4.105}$$

sendo Q_w é o valor da "medida de qualidade" do "pior triângulo" e d é a densidade do último ponto inserido, a qual é calculada pela função na Equação 4.103. Na Figura 4.16 pode vista a malha refina para o exemplo considerado.

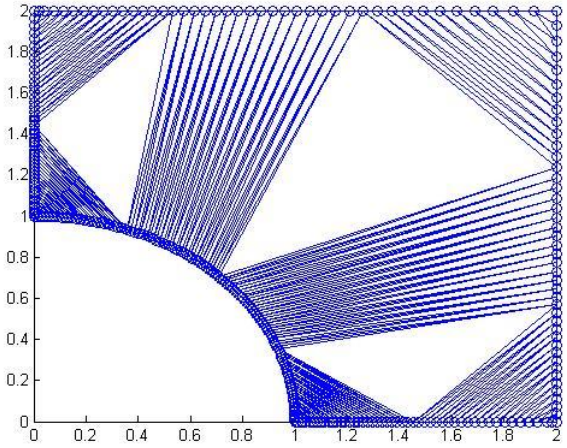


Figura 4.15 Malha apenas com os triângulos internos

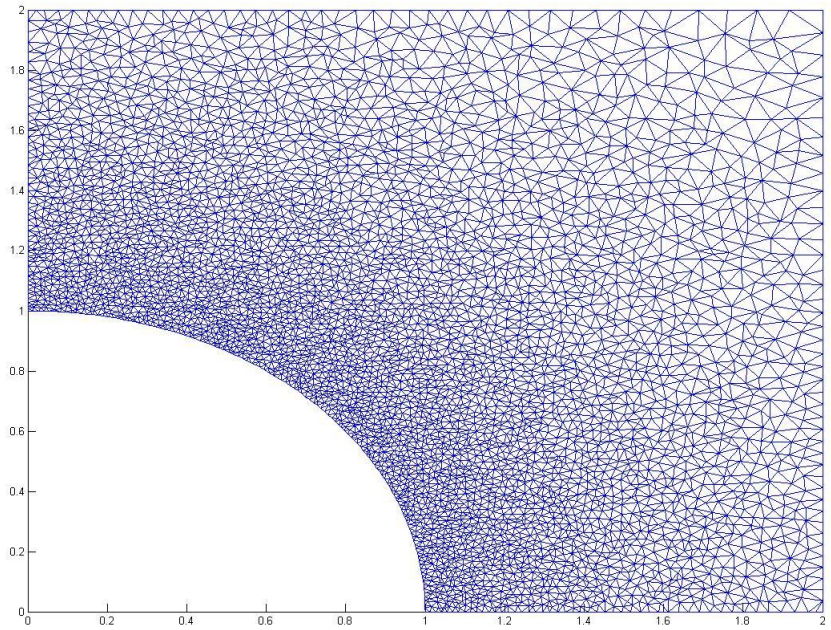


Figura 4.16 Malha refinada

O último passo, consiste na regularização da malha. O processo de regularização consiste em percorrer ponto por ponto (pontos do contorno não são considerados) modificando as suas coordenadas. Este processo é mostrado na Figura 4.17, onde as novas coordenadas x_i e y_i para o ponto i mostrado na figura são calculadas pelas Equações 4.106 e 4.107. A malha regularizada para o exemplo considerado pode ser vista na Figura 4.18.

$$x_i = \frac{x_0 + x_1 + x_2 + x_3}{4} \tag{4.106}$$

$$y_i = \frac{y_0 + y_1 + y_2 + y_3}{4} \tag{4.107}$$

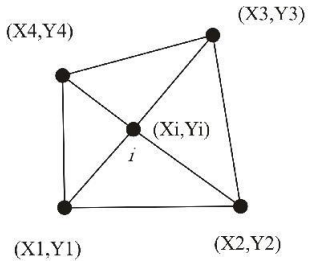


Figura 4.17 Regularização da malha

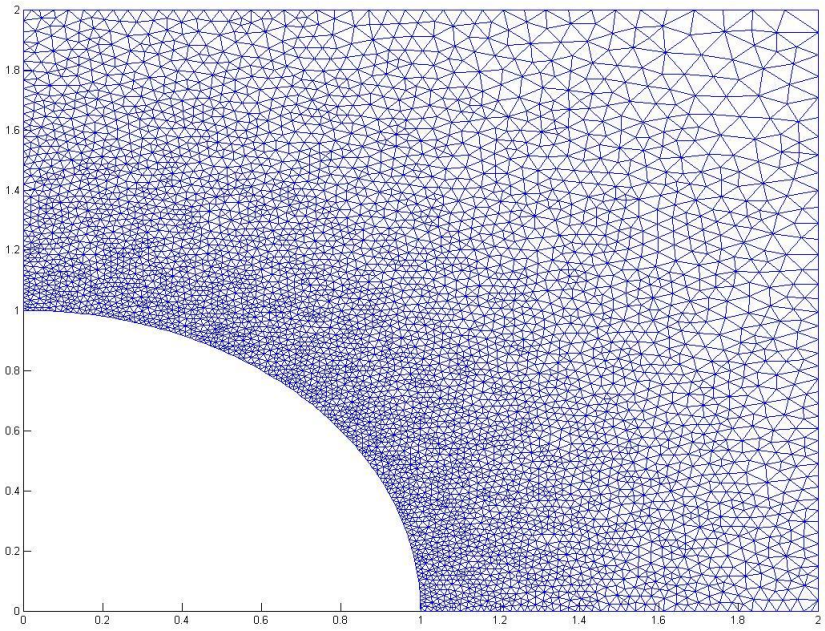


Figura 4.18 Malha regularizada

CAPÍTULO 5

Soluções Numéricas

Neste capítulo serão apresentados os resultados obtidos nas simulações bem como as técnicas utilizadas para validação das implementações. Neste capítulo também será discutido o processo de geração de malha no simulador.

5.1 Modelo Elasto-Viscoplástico com Dano

5.1.1 Resultados da Simulação

A solução numérica para o problema descrito foi implementado no MPhyScas. Para validação desta implementação uma solução para o problema também foi implementada no software MATLAB. Os resultados obtidos serão apresentados nesta seção sempre realizando uma comparação entre os resultados obtidos nos dois programas. Um estudo sobre o efeito do dano também será feito.

A simulação foi realizada aplicando-se uma carregamento cíclico de $P(t) = 5,5 \times 10^5 \text{sen}(2\pi t)$ N na extremidade livre da barra e uma força de corpo $f = 1,0 \times 10^4\, N/m$, durante um período de tempo de $22,64$ segundos e com incremento de tempo de $\Delta t = 1,0 \times 10^{-4}$ segundos. O valor da área da seção transversal foi considerada constante em toda barra com valor de $A = 3,0 \times 10^{-3}\, m^2$, o valor da constante do meio elástico k foi considerado igual a zero e o comprimento da barra foi de $L = 1,0\, m$. O valor dos parâmetros do material utilizados na simulação podem ser vistos na tabela 5.1[10]. Uma malha com vinte elementos foi simulada em cada sistema.

A influência do dano pode ser vista nas curvas da tensão pela deformação mostradas nos gráficos da figura 5.1. Observa-se que na figura 5.1(a) o material ficou praticamente em regime elástico enquanto que na figura 5.1(b) observa-se o efeito da plastificação do material.

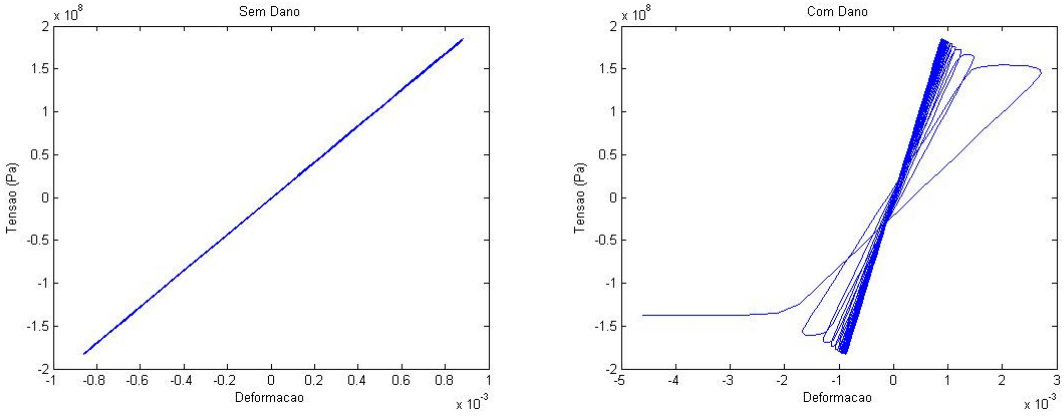
As curvas na figura 5.1 foram obtida utilizando-se o MPhyScaS enquanto que as curvas da figura 5.2 foram obtidas utilizando-se o MATLAB. Observa-se que foram obtidos resultados bem próximos em ambas simulações.

O efeito do dano na deformação plástica bem como na tensão pode ser visto na figura 5.3. Na figura 5.3(a) observa-se uma menor deformação plástica em relação a figura 5.3(b). O efeito da plastificação pode ser observado também nos valores máximos e mínimos alcançados pela tensão, na figura 5.3(a) observa-se que os máximos e mínimos de tensão ficam praticamente constantes, enquanto que na figura 5.3(b) observa-se que a diferença entre os valores máximo e o mínimo da tensão tende a diminuir.

Na figura 5.4 tem-se um resultado próximo ao obtido na figura 5.3, mostrando mais uma vez a concordância entre os dois programas implementados.

Tabela 5.1 Valor dos parâmetros utilizados na simulação

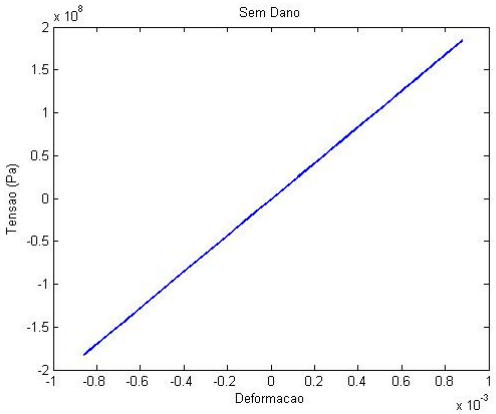
Parâmetro	Valor
$E\ (Pa)$	210.0×10^9
$K\ (Pa)$	151.0×10^6
N	24.0
$\alpha\ (1/K)$	15.4×10^{-6}
$\rho\ (kg/m^3)$	7.8×10^3
$c_p\ (J/kgK)$	4.54×10^2
$b\ (Pa)$	60.0×10^6
d	8.0
$\sigma_p\ (Pa)$	82.0×10^6
$a_1\ (Pa)$	108.3×10^9
$a_2\ (Pa)$	4.5×10^9
φ_1	2800.0
φ_2	25.0
$S_0\ (Pa)$	20.0×10^3



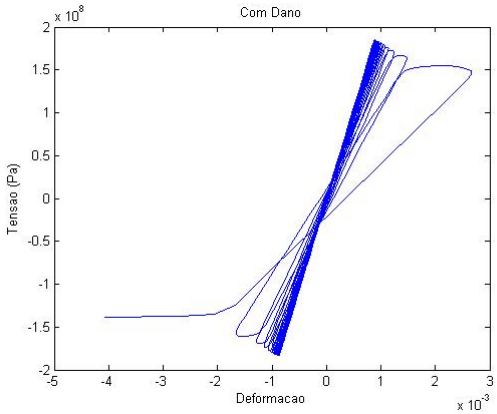
(a) Tensão × Deformação sem dano

(b) Tensão × Deformação com dano

Figura 5.1 Curva da Tensão × Deformação no MPhyScas

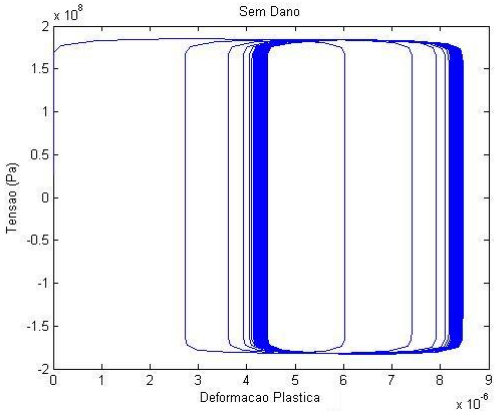


(a) Tensão × Deformação sem dano

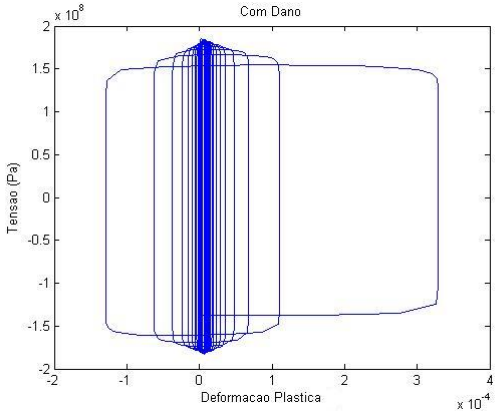


(b) Tensão × Deformação com dano

Figura 5.2 Curva da Tensão × Deformação no MATLAB

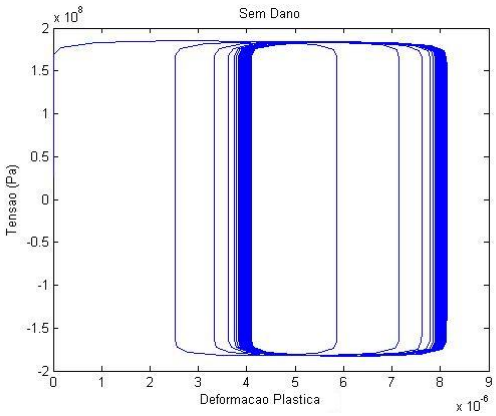


(a) Tensão × Deformação plástica sem dano

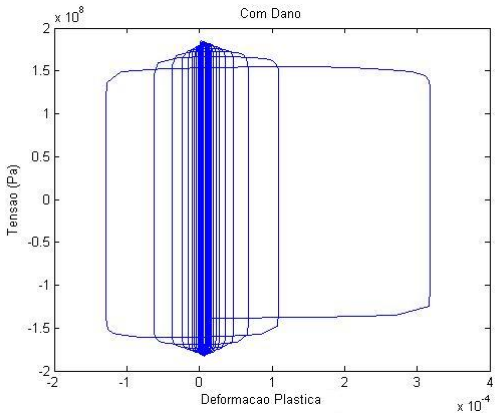


(b) Tensão × Deformação plástica com dano

Figura 5.3 Curva da Tensão × Deformação plástica no MPhyScas



(a) Tensão × Deformação plástica sem dano



(b) Tensão × Deformação plástica com dano

Figura 5.4 Curva da Tensão × Deformação plástica no MATLAB

A evolução do dano no tempo pode ser visto na figura 5.6, onde uma comparação entre os dois programas também pode ser observada. A diferença absoluta entre o valor do dano nas duas simulações foi da ordem de 0.0001% (no final da simulação).

O efeito do dano na evolução da temperatura é mostrado nos gráficos da figura 5.6. Pode-se observar que quando o dano não é levado em conta, a temperatura fica oscilando em torno de uma temperatura média de 293 K. Quando o dano é levado em consideração é observado um aumento na temperatura. As oscilações nesses gráficos ocorrem devido ao efeito termoelástico [8] [10].

O efeito do dano nos endurecimentos isotrópico e cinemático podem ser vistos nas figuras 5.7 e 5.9(a), respectivamente. Na figura 5.7(b), observa-se o amolecimento do material devido o efeito do dano. As figuras 5.8 e 5.9(b) mostram os resultados obtidos para os respectivos endurecimentos no MATLAB.

5.2 Modelo de Difusão Bidimensional

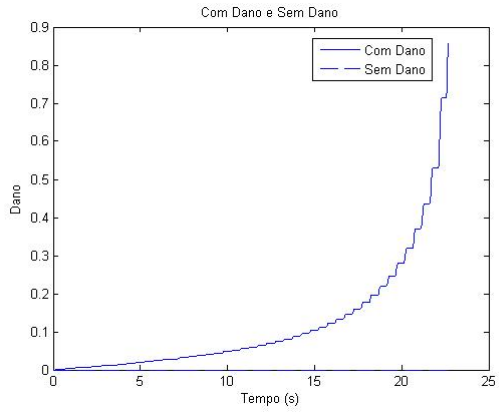
A solução numérica para o problema descrito na seção 4.2 foi implementada no MPhyScas. Para validação dos resultados, foram simulados problemas com simetria axial, os quais uma solução analítica foi desenvolvida. A comparação dos resultados foi realizada calculando-se o erro relativo entre entre as duas soluções. O erro foi calculado utilizando a norma L2. Na Tabela 5.2 [17] pode ser visto o valor dos parâmetros do material utilizados nas simulações.

Tabela 5.2 Valor dos parâmetros utilizados na simulação do problema bidimensional

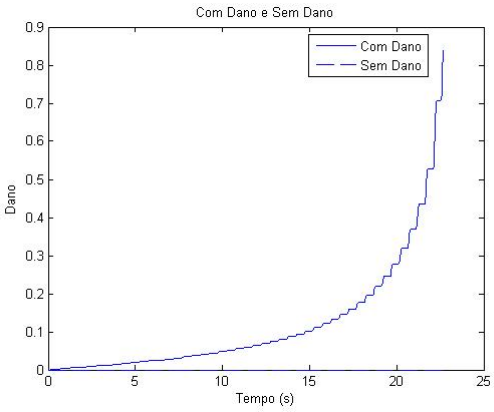
Parâmetro	Valor
$E\ (Pa)$	200.0×10^9
e	5.2
ν	0.3
$\alpha\ (J/m^3)$	1.94×10^{10}
$D\ (m^2/s)$	1.0

A geometria do problema pode ser vista na Figura 5.10(a), onde $R_0 = 1,0$ e $R_1 = 2,0$. Os códigos dos entes geométricos podem ser vistos na Figura 5.10(b).

O processo de validação do problema bidimensional implementado foi dividido em etapas. Na primeira etapa foi desenvolvida uma solução analítica com simetria axial para o problema de elasticidade (cálculo dos deslocamentos nodais, cálculo das deformações e cálculo das tensões). Nesta etapa a influência da difusão no problema de elasticidade não foi considerada. Na segunda etapa, uma solução analítica com simetria axial para o problema de difusão acoplado com elasticidade (nesta caso o problema de elasticidade não depende da difusão) foi desenvolvida. Nesta etapa, a solução do problema de difusão não depende do raio. Por fim, uma solução analítica com simetria axial para o problema de difusão acoplado com elasticidade semelhante ao implementado na segunda etapa foi desenvolvida, no entanto, esta solução apresenta uma dependência em relação ao raio. As quantias que foram ativadas e os resultados obtidos nessas simulações serão mostrados nas próximas seções.

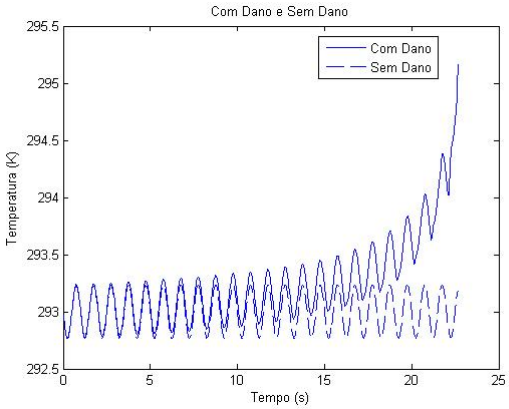


(a) Dano × Tempo no MPhyScaS

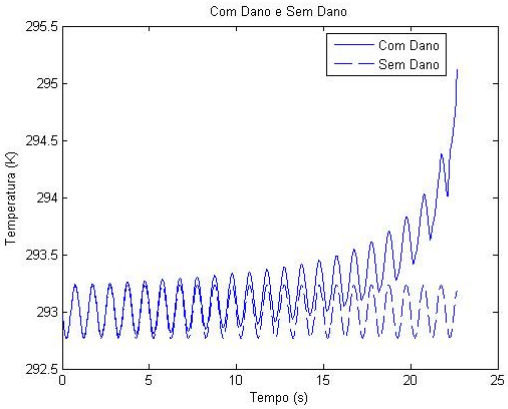


(b) Dano × Tempo no MATLAB

Figura 5.5 Curva do Dano × Tempo no MPhyScaS e no MATLAB

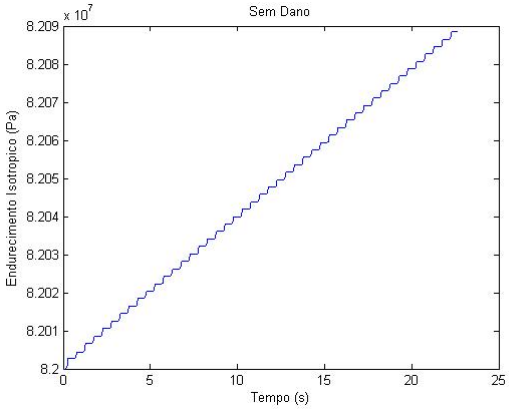


(a) Temperatura × Tempo no MPhyScaS

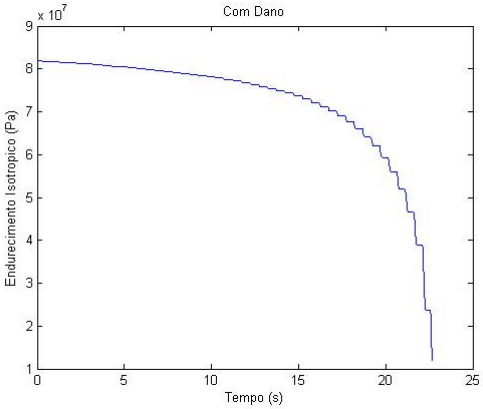


(b) Temperatura × Tempo no MATLAB

Figura 5.6 Curva da Temperatura × Tempo no MPhyScaS e no MATLAB

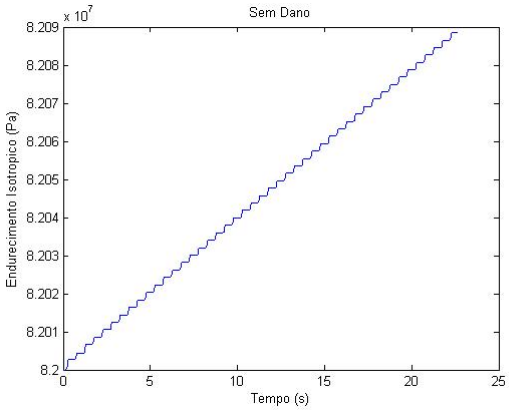


(a) Endurecimento isotrópico× Tempo sem dano

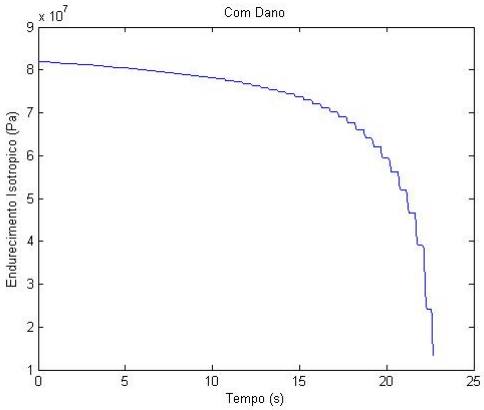


(b) Endurecimento isotrópico× Tempo com dano

Figura 5.7 Curva do Endurecimento isotrópico × Tempo no MPhyScaS

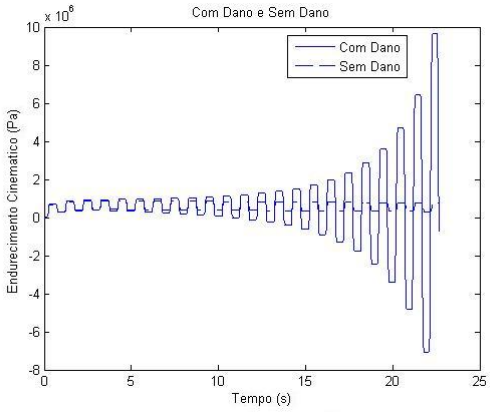


(a) Endurecimento isotrópico× Tempo sem dano

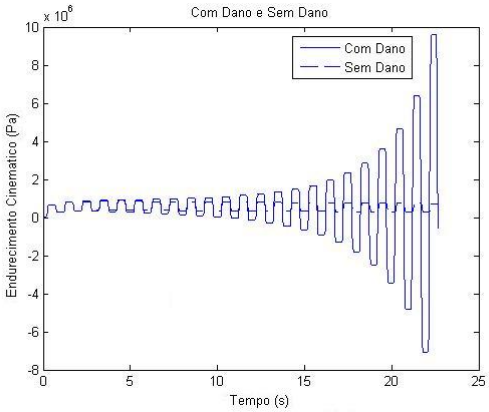


(b) Endurecimento isotrópico× Tempo com dano

Figura 5.8 Curva do Endurecimento isotrópico × Tempo no MATLAB



(a) Endurecimento cinemático × Tempo no MPhyScaS



(b) Endurecimento cinemático × Tempo no MATLAB

Figura 5.9 Curva do Endurecimento cinemático × Tempo no MPhyScaS e no MATLAB

5.2.1 Problema de Elasticidade com Simetria Axial

Nesta seção serão mostradas as quantias que foram ativadas no problema de elasticidade bem como as soluções obtidas na simulação, uma comparação entre a solução analítica e a solução numérica será realizada. Para o Fenômeno Deslocamento as quantias ativadas no domínio (ente geométrico com código 8) podem ser vistas na Tabela 5.3 e as quantias ativadas no contorno podem ser vistas na Tabela 5.4.

Tabela 5.3 Quantias no Domínio - Fenômeno Deslocamento

Quantia	Código da Quantia	Descrição
WFElastStiffness	0	Monta a matriz de rigidez
WFElastLoad	1	Produz a força de corpo
WFElastBoundCond	2	Gerencia as condições de contorno

Tabela 5.4 Quantias no Contorno - Deslocamento

GeomEntity	Código da Quantia	Descrição
4	5	Dirichlet - $u = 0$ na direção normal
5	5	Dirichlet - $u = 0$ na direção normal
6	5	Dirichlet - $u = 0$ na direção normal
7	5	Dirichlet - $u = 1.0 \times 10^{-4}$ na direção normal

No problema de elasticidade, para se ter uma simetria axial, a força de corpo utilizada é dada pela função mostrada na Equação 5.1

$$\frac{f_r}{\sqrt{x^2+y^2}} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.1}$$

onde foi utilizado $f_r = 1.0 \times 10^{10}\, N/m^2$.

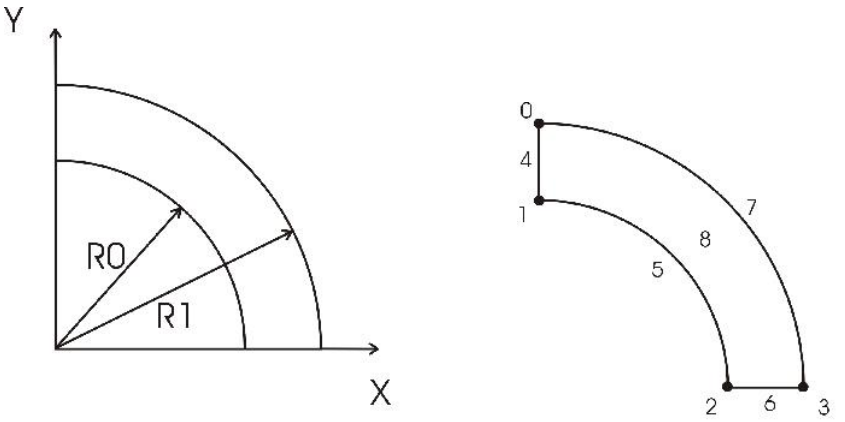
O Fenômeno Deformação só possui uma quantia ativada no domínio, que pode ser vista na Tabela 5.5.

Tabela 5.5 Quantias no Domínio - Deformação

Quantia	Código da Quantia	Descrição
WFDeformation	0	Calcula a deformação

O Fenômeno Matriz de Elasticidade só possui uma quantia ativada no domínio, esta pode ser vista na Tabela 5.6.

E por fim, o Fenômeno Tensão só possui quantias ativadas no domínio, as quais podem ser vistas na Tabela 5.7.



(a) Geometria do problema bidimensional

(b) Geometria com os códigos

Figura 5.10 Geometria do problema bidimensional

Tabela 5.6 Quantias no Domínio - Matriz de Elasticidade

Quantia	Código da Quantia	Descrição
WFElastArrStr	1	Matriz de elasticidade (estado plano de tensões)

Tabela 5.7 Quantias no Domínio - Tensão

Quantia	Código da Quantia	Descrição
WFElemStr	0	Tensão nos elementos
WFStrTrace	1	Traço das tensões nodais
WFENodStr	2	Tensão nodal

A solução analítica desenvolvida para o deslocamento radial é pode ser visto na Equação 5.2 (ver apêndice A).

$$u_r = -kr^2 + \frac{c_1}{2}r + \frac{c_2}{r} \tag{5.2}$$

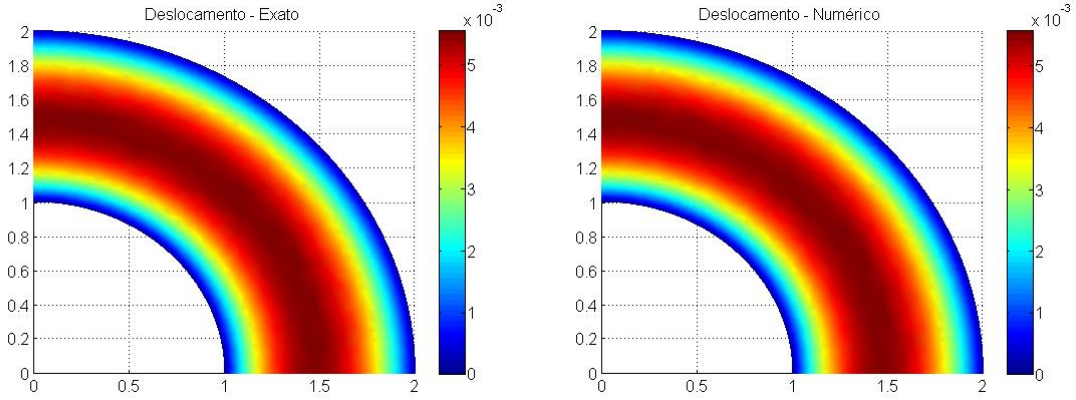
onde

$$k = \frac{f_r(1-\nu^2)}{3E} \tag{5.3}$$

$$c_2 = -\frac{1}{3}(4k + 2\times 10^{-4}) \tag{5.4}$$

$$c_1 = 2k - 2c_2 \tag{5.5}$$

A solução exata e a solução numérica para o deslocamento podem ser vistas nas Figuras 5.11(a) e 5.11(b). O erro relativo entre as duas soluções foi de 0.11%.



(a) Solução exata para o deslocamento

(b) Solução numérica para o deslocamento

Figura 5.11 Soluções para o deslocamento

Na Figura 5.13(a) é mostrado o o gráfico do erro relativo para todos os pontos da geometria. Na Figura 5.13(b), observa-se também o erro relativo para os pontos pertencentes as curvas para $\theta = 0^0$, $\theta = 45^0$ e $\theta = 90^0$ (com r variando de $R_0 = 1$ até $R_1 = 2$, ver Figura 5.12). Na Tabela 5.8 podem ser vistos os erros máximos e minímos para os gráficos da Figura 5.13(b). Pode-se observar que nos gráficos da Figura 5.13(a) os erros obtidos foram menores que os erros no gráfico da Figura 5.13(b). Isto se deve ao fato de que os pontos escolhidos para gerar os gráficos da Figura 5.13(b) não foram pontos dos vértices da malha então, o valor do deslocamento para esses pontos foram obtidos através da interpolação dos valores nodais do deslocamento em um determinado elemento finito, enquanto que na Figura 5.13(a) foram utilizados os valores nodais dos deslocamentos (sem interpolação).

5.2.2 Problema de Difusão Independente do Raio

Nesta seção serão mostradas as quantias que foram ativadas no problema de difusão bem como as soluções obtidas na simulação. Uma comparação entre a solução analítica e a solução

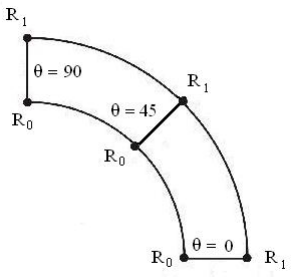
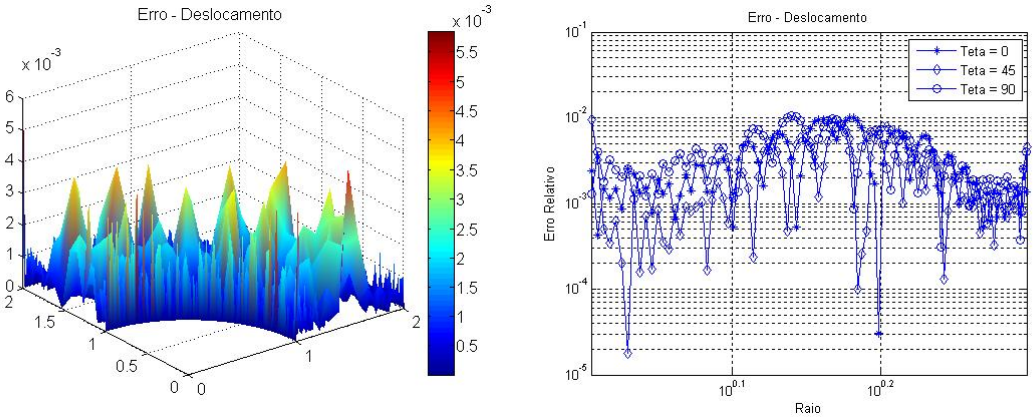


Figura 5.12 Ângulos θ para os quais foram calculados os erros



(a) Erro relativo do deslocamento para todos os pontos da geometria

(b) Erro relativo do deslocamento para os pontos pertencentes as curvas para $\theta = 0^0$, $\theta = 45^0$ e $\theta = 90^0$

Figura 5.13 Gráfico dos erros para o deslocamento

Tabela 5.8 Erros máximos e minímos - deslocamento

θ	Erro máximo	Erro mínimo
0°	$1,0014 \times 10^{-2}$	$3,0757 \times 10^{-5}$
45°	$9,4156 \times 10^{-3}$	$1,795 \times 10^{-5}$
90°	$1,0637 \times 10^{-2}$	$3,0787 \times 10^{-4}$

Tabela 5.9 Quantias no Domínio - Difusão

Quantia	Código da Quantia	Descrição
WFCondMtx	0	Calcula a matriz capacitância
WFCondMtxGrd	2	Calcula a matriz condutância
WFCondMtxTr	3	Calcula o termo acoplado a tr($\mathbf{S}$)
WFDifBoundCond	4	Gerencia as condições de contorno
WFDifInitialize	9	Inicializa o campo vetorial$c$
WFDifLoad	10	Calcula o termo de fonte de soluto

numérica será realizada. As quantias ativadas para o Fenômeno Difusão no domínio e no contorno para esta simulação podem ser vistas nas Tabelas 5.9 e 5.10, respectivamente.

A solução analítica para a concentração de soluto pode ser vista na Equação 5.6 (ver apêndice B)

$$c(x,y,t) = \frac{2}{\lambda D} e^{-\lambda D t} \tag{5.6}$$

onde $\lambda = 0.5$.

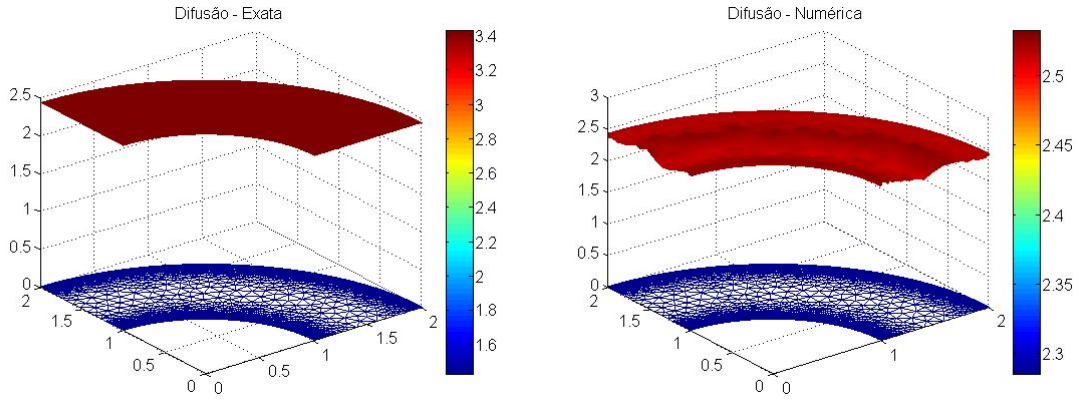
O termo de fonte de soluto utilizado nesta simulação pode ser visto na Equação 5.7

$$f(x,y,t) = \frac{-2}{\sqrt{x^2+y^2}} e^{-\lambda D t} (\sqrt{x^2+y^2} + \frac{k}{\lambda}) \tag{5.7}$$

onde

$$k = \frac{e f_r (1+\nu)}{\alpha} \tag{5.8}$$

A simulação foi realizada em um intervalo de tempo de 1 segundo, com um total de 1000 passos no tempo. A solução exata e a solução numérica para concentração do soluto pode ser vista nas Figuras 5.14(a) e 5.14(b). O erro relativo entre as duas soluções foi de 1.8%.



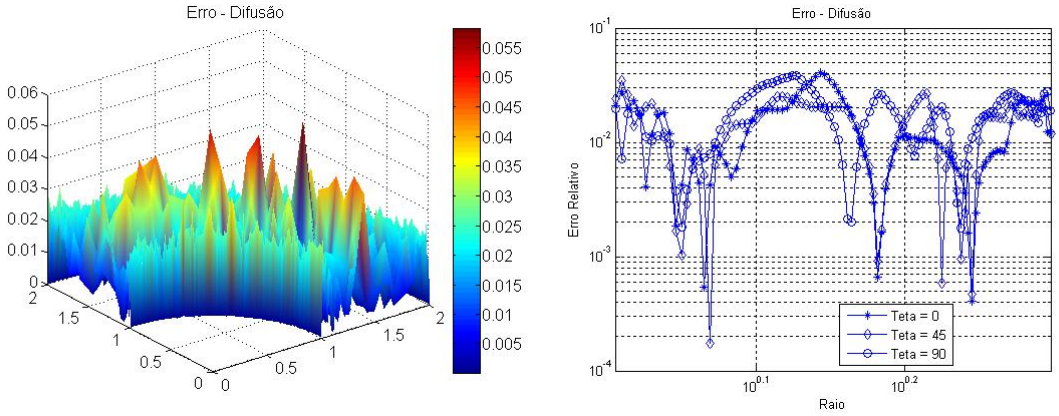
(a) Solução exata da concentração do soluto (b) Solução numérica da concentração do soluto

Figura 5.14 Solução para a concentração do soluto

Na Figura 5.15(a) é mostrado o o gráfico do erro relativo para todos os pontos da geometria. Na Figura 5.15(b), observa-se também o erro relativo para os pontos pertencentes as curvas para $\theta = 0^0$, $\theta = 45^0$ e $\theta = 90^0$ (com r variando de $R_0 = 1$ até $R_1 = 2$, ver Figura 5.12). Na Tabela 5.11 podem ser vistos os erros máximos e minímos para os gráficos da Figura 5.15(b). Pode-se observar que nos gráficos da Figura 5.15(a) os erros obtidos foram menores que os erros no gráfico da Figura 5.15(b). Isto se deve ao fato de que os pontos escolhidos para gerar os gráficos da Figura 5.15(b) não foram pontos dos vértices da malha então, o valor da concentração para esses pontos foram obtidos através da interpolação dos valores nodais da concentração em um determinado elemento finito, enquanto que na Figura 5.15(a) foram utilizados os valores nodais das concentrações (sem interpolação).

Tabela 5.10 Quantias no Contorno - Difusão

GeomEntity	Código da Quantia	Descrição
5	5	Dirichlet - usando a Equação 5.6
7	5	Dirichlet - usando a Equação 5.6



(a) Erro relativo da difusão para todos os pontos da geometria

(b) Erro relativo da difusão para os pontos pertencentes as curvas para $\theta = 0^0$, $\theta = 45^0$ e $\theta = 90^0$

Figura 5.15 Gráfico dos erros para a difusão

Tabela 5.11 Erros máximos e minímos - difusão

θ	Erro máximo	Erro mínimo
$0°$	$4,0616 \times 10^{-2}$	$4,0332 \times 10^{-4}$
$45°$	$3,4485 \times 10^{-2}$	$1,7559 \times 10^{-4}$
$90°$	$3,8279 \times 10^{-2}$	$1,7787 \times 10^{-3}$

5.2.3 Problema de Difusão Dependente do Raio

Nesta seção serão mostradas as quantias que foram ativadas no problema de difusão. Uma comparação entre a solução analítica e a solução numérica será realizada. As quantias ativadas para o Fenômeno no domínio e no contorno são as mesmas das Tabelas 5.9 e 5.10 no entanto, a equação utilizada para o cálculo da condição de Dirichlet é a Equação 5.9.

A solução analítica da concentração de soluto desenvolvida para este problema pode ser vista na Equação 5.9 (ver apêndice B)

$$c(x,y,t) = \frac{2}{\lambda D} e^{-\lambda D t}\left(-\sqrt{x^2+y^2} + \frac{2k}{\lambda}\right) \tag{5.9}$$

onde $\lambda = 0.5$.

O termo de fonte de soluto utilizado nesta simulação pode ser visto na Equação 5.10

$$f(x,y,t) = \frac{2}{\sqrt{x^2+y^2}} e^{-\lambda D t}(x^2+y^2+h) \tag{5.10}$$

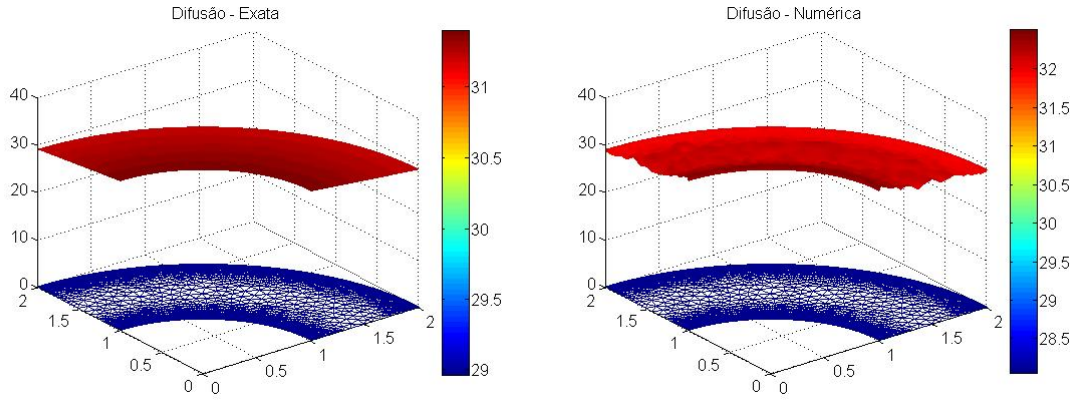
onde

$$k = \frac{e f_r (1+\nu)}{\alpha} \tag{5.11}$$

e

$$h = -\frac{-\lambda + 2k^2}{\lambda^2} \tag{5.12}$$

A simulação foi realizada em um itervalo de tempo de 1 segundo, com um total de 1000 passos no tempo. A solução exata e a solução numérica para concentração do soluto pode ser vista nas Figuras 5.16(a) e 5.16(b). O erro relativo entre as duas soluções foi de 1.75%. Uma malha mais refinada foi utilizada nesta simulação, por esta razão obteve-se um erro relativo menor em comparação com a outra simulação.

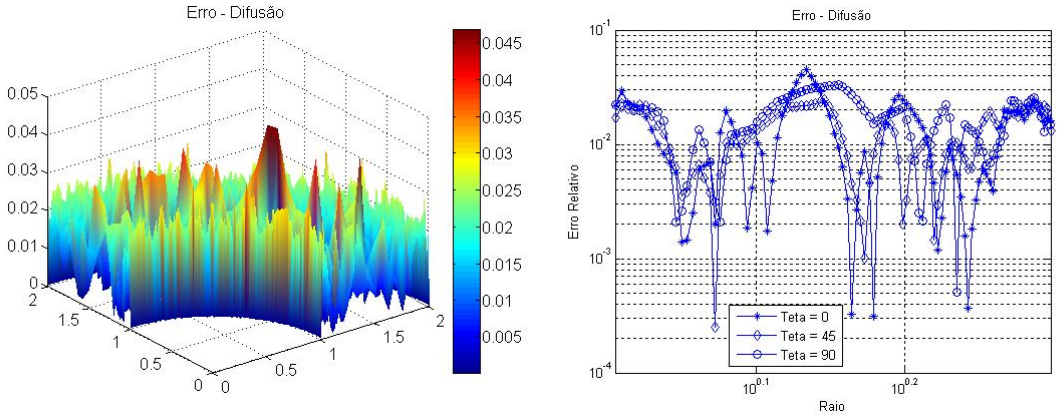


(a) Solução exata da concentração do soluto (b) Solução numérica da concentração do soluto

Figura 5.16 Solução para a concentração do soluto

Na Figura 5.17(a) é mostrado o o gráfico do erro relativo para todos os pontos da geometria. Na Figura 5.17(b), observa-se também o erro relativo para os pontos pertencentes as curvas

para $\theta = 0^0$, $\theta = 45^0$ e $\theta = 90^0$ (com r variando de $R_0 = 1$ até $R_1 = 2$, ver Figura 5.12). Na Tabela 5.12 podem ser vistos os erros máximos e minímos para os gráficos da Figura 5.17(b). Pode-se observar que nos gráficos da Figura 5.17(a) os erros obtidos foram menores que os erros no gráfico da Figura 5.17(b). Isto se deve ao fato de que os pontos escolhidos para gerar os gráficos da Figura 5.17(b) não foram pontos dos vértices da malha então, o valor da concentração para esses pontos foram obtidos através da interpolação dos valores nodais da concentração em um determinado elemento finito, enquanto que na Figura 5.17(a) foram utilizados os valores nodais das concentrações (sem interpolação).



(a) Erro relativo da difusão para todos os pontos da geometria

(b) Erro relativo da difusão para os pontos pertencentes as curvas para $\theta = 0^0$, $\theta = 45^0$ e $\theta = 90^0$

Figura 5.17 Gráfico dos erros para a difusão

Tabela 5.12 Erros máximos e minímos - difusão

θ	Erro máximo	Erro mínimo
$0°$	$4,4756 \times 10^{-2}$	$3,1448 \times 10^{-4}$
$45°$	$2,4645 \times 10^{-2}$	$2,4952 \times 10^{-4}$
$90°$	$3,3288 \times 10^{-2}$	$5,1072 \times 10^{-4}$

CAPÍTULO 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Foi apresentado o desenvolvimento de um simulador para problemas acoplados utilizando o MEF objetivando o aumento da reusabilidade, da manutenabilidade e da adaptabilidade, permitindo um trabalho cooperativo multidisciplinar e um gerenciamento de conhecimento e possibilitando alterações no algoritmo de solução com um mínimo de reprogramação.

Foi observado que os simuladores baseados no método do elemento finito podem ser moldados em uma arquitetura de camadas, são elas: laços iterativos globais, articulação de resolvedores, resolvedores e fenômenos. Esta divisão é importante no sentido da modularização do software, no entanto não fornece uma visão de como pode ser descrita a dependência entre as camadas, o que é muito importante na definição de abstrações para padronizar a forma como processos e dados nas camadas se comportam e interagem. Foi mostrado que problemas multi-físicas acoplados tornam o desenvolvimento de simuladores bastante complexo devido ao fato de que a produção e montagem de matrizes e vetores locais para cada fenômeno podem estar acopladas com outros fenômenos, significando que o cálculo dessas quantiais necessitam de informações de outros fenômenos, onde essas informações são definidas no nível da solução, fazendo com que mudanças do algoritmo de solução possam exigir extensa reprogramação em ambas as camadas.

Com a finalidade de contornar estas dificuldades, foi implementado um ambiente de simulação baseado na representação das camadas através dos padrões: o **Kernel** que fica responsável pela representação dos laços iterativos globais, o padrão **Block** responsável pela representação da articulação entre os resolvedores, o padrão **Group** responsável pela representação dos procedimentos relacionados com a montagem e solução de sistemas algébricos são implementados. No **Group** também são implementados operações com matrizes e vetores (blas 1, 2 e 3). Por fim, o padrão **Phenomenon** fica responsável pelo cálculo de matrizes, vetores e escalares locais, por operações envolvendo matrizes e vetores no nível do elemento finito e a montagem em matrizes e vetores globais fornecidos. O padrão GIG também foi utilizado com a finalidade representar o algoritmo na forma de um grafo acíclico orientado, tornando fácil a sua definição a partir da linguagem natural do algoritmo.

Todo o ambiente de simulação foi implementado neste trabalho, tomando-se como ponto de partida a conceitualização teórica proposta em [2]. A linguagem de programação utilizada foi o C++. Na implementação desse sistema foram utilizados cerca de seiscentos arquivos, totalizando cerca de duzentas classes.

A partir do ambiente de simulação implementado foram realizas a simulação computacional de dois modelos multi-físicas. O primeiro modelo implementado consistiu em um modelo

termomecânico unidimensional com a finalidade de possibilitar uma análise mais simples do acoplamento fortemente não linear entre os diferentes mecanismos dissipativos (plasticidade, endurecimento, dano e temperatura). Na implementação deste modelo foi utilizada uma versão mais simplificada do simulador. O segundo modelo implementado (mais complexo) baseou-se em um modelo bidimensional com a finalidade de analisar a deterioração de sólidos elásticos levando em conta a difusão de soluto. Para a implementação deste modelo, foi utilizada uma versão mais sofisticada do simulador.

A análise dos resultados obtidos foram realizadas de duas formas. A primeira forma de análise, a qual foi empregada no modelo termomecânico, consistiu em comparar os resultados obtidos em dois programas, um implementado no MPhyScas e o outro implementado no MATLAB. A outra forma de análise consistiu na obtenção de soluções analíticas manufaturadas. Esta forma de análise foi empregada no modelo de difusão. No primeiro modelo, foi obtido um erro absoluto da ordem de $0,0001\%$ para o dano entre os dois programas. Esta diferença foi considerada satisfatória, uma vez que as operações realizadas nos dois programas são completamente diferentes. No segundo modelo, foi obtido um erro relativo de $0,11\%$ para o problema de elasticidade e um erro relativo de $1,8\%$ para o primeiro problema de difusão e um erro de $1,75\%$ para o segundo problema de difusão, os quais são considerados satisfatórios para problemas de engenharia.

A implementação desses dois modelos apresentou uma melhoria na quantidade de tempo gasto com o desenvolvimento de simuladores e na reusabilidade dos componentes de software (abrindo espaço para a construção de repositórios). Isto pôde ser observado na implementação do segundo modelo, onde foi utilizada grande parte da estrutura implementada na primeira versão do simulador. Essa reutilização foi possível pois todos as informações para a construção do simulador são fornecidas através de arquivos de dados.

6.2 Trabalhos Futuros

No nível da arquitetura, será implementado uma nova versão para o sistema, menos simplificada, considerando

- Uma arquitetura de componentes de alto desempenho (CCA - Common Component Architecture);
- Desenvolvimento de uma interface gráfica.

No nível das implementações, pretende-se

- Utilização de outras bibliotecas para serviços como geração de malha, visualização, resolvedores lineares, etc;
- Implementar adaptatividade;
- Implementar problemas tridimensionais;
- Considerar problemas mais complexos.

Referências Bibliográficas

[1] COMMITTEE on Theoretical and Applied Mechanics. *A report of the United States Association for Computational Mechanics*, Research Directions in Computational Mechanics, 2000.

[2] LENCASTRE, M. *Conceptualisation of an Environment for the Development of FEM Simulators*. Tese (Doutorado em Ciências da Computação) — Universidade Federal de Pernambuco, Recife, Pernambuco, 2004.

[3] REDE de Pesquisa Cooperativa em Modelagem Computacional. Disponível em: <http://www.demec.ufpe.br/rpcmod/historico.htm>.

[4] ROSS, C. T. F. Finite element programs in structural engineering & continuum mechanics. Albion Engineering Science Series, 1996.

[5] SANTOS, F. C. G.; BRITO, E. R. R. J.; BARBOSA, J. M. A. Dealing with coupled phenomena in the finite element method. *XXVII Latin American Congress on Computational Methods in Engineering*, Belém, Brasil, p. 461, 2006.

[6] SIMO, J. D.; MIECHE, C. Associative coupled thermoplasticity at finite strains: Formulation, numerical analysis and implementation. *Computer Methods in Applied Mechanics and Engineering*, Elsevier Science Ltd, n. 98, p. 41–104, 1992.

[7] CONSTANTINESCU, A. et al. A computational approach to thermomechanical fatigue. *International Journal of Fatigue*, Elsevier Science Ltd, n. 26, p. 805–818, 2004.

[8] PACHECO, P. M. C. L. *Análise do acoplamento termomecânico em materiais elasto-viscoplásticos*. Tese (Doutorado em Engenharia Mecanica) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1994.

[9] CHABOCHE, J. L.; J., L. Mechanics of solids materials. Cambridge University Press, 1990.

[10] BARBOSA, J. M. A. *Estudo da Localização da Deformação em Materiais Elasto-Viscoplásticos Levando-se em Conta Efeitos Térmicos e de Inércia*. Tese (Doutorado em Engenharia Mecanica) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1998.

[11] WOODTLI, J.; KIESELBACH, R. Damage due to hydrogen embrittlement and stress corrosion cracking. *Engineering Failure Analysis*, Elsevier Science Ltd, n. 7, p. 427–450, 2000.

[12] KNOTT, T. A problem well thought through. *Offshore Engineer*, 2001. Disponível em: <http://www.oilonline.com/news/features/oe/20011101.A_proble.7851.asp>.

[13] SOFRONIS, P.; TAHA, A. A micromechanics approach to the study of hydrogen transpor an embrittlement. *Engineering Fracture Mechanics*, Elsevier Science Ltd, n. 68, p. 803–837, 2001.

[14] BOLOTIN, V. V.; SHIPKOV, A. A. Mechanical aspects of corrosion fatigue and stress corrosion cracking. *International Journal of Solids and Structures*, Elsevier Science Ltd, n. 38, p. 7297–7318, 2001.

[15] CHOPIN, C. E.; P., T. J. Modeling of coupled deformation-diffusion in non-porous solids. *International Journal of Engineering Science*, Elsevier Science Ltd, n. 37, p. 1–24, 1999.

[16] ZOHDI, T. I. Modeling and simulation of a class of coupled thermo-chemo-mechanical processes in multiphase solids. *Computer methods in applied mechanics and engineering*, Elsevier Science Ltd, n. 193, p. 679–699, 2004.

[17] DUDA, F. P. et al. On the modeling of deformation-diffusion-damage coupling in elastic solids. *III European Conference on Computational Mechanics Solids, Structures and Coupled Problems in Engineering*, Lisbon, Portugual, 2006.

[18] SCI Institute. Disponível em: <http://www.sci.utah.edu/>.

[19] TAYLOR, L.; EDWARDS, H. C.; STEWART, J. R. Functional requirements for sierra. *Sandia National Laboratories*, 1999.

[20] CTL Manual for Linux/Unix for the Usage with C++. Disponível em: <http://www.wire.tu-bs.de/forschung/projekte/ctl/files/manual.pdf>.

[21] SEED, G. An introduction to object-oriented programming in c++: With application in computer graphics. Springer, 1996.

[22] SANTOS, F.; LENCASTRE, M.; RODRIGUES, I. Fem simulator based on skeletons for coupled phenomena. *Proceedings of the 2nd Latin American Conference on Pattern Languages of Programming*, Brazil, I, 2002.

[23] SANTOS, F.; LENCASTRE, M.; VIEIRA, M. Workflow for simulators based on finite element method. *Proceedings of the International Conference on Computational Science (ICCS)*, Melbourn, Australia and Saint Petersburg, Russia, 2003.

[24] SANTOS, F.; LENCASTRE, M. An approach for fem simulator development. *Journal of Computational ans Applied Mathematics*, Holanda, v. 185, n. 2, p. 326–346, 2006.

[25] SANTOS, F. C. G.; BRITO, E. R. R. J.; BARBOSA, J. M. A. Phenomenon computational pattern: coupling relationship between phenomena on multi-physics simulation. *Industrial Simulation Conference 2006*, Palermo, Italy, I, p. 182–187, 2006.

[26] SANTOS, F. C. G.; BRITO, E. R. R. J.; BARBOSA, J. M. A. Phenomenon computational coupling relationship between phenomena on multi-physics simulation. *20th European Conference on Modelling and and Simulation*, Bonn, Germany, 2006.

[27] SANTOS, F. C. G.; BRITO, E. R. R. J.; BARBOSA, J. M. A. Coping with data dependence and sharing in the simulatin of coupled phenomena. *International Congress on Computational and Applied Mathematics - ICCAM*, Leuven, Belgium, I, 2006.

[28] STROUSTRUP, B. A linguagem de programação c++. Bookman, 2002.

[29] BRITO, E. R. R. J.; BARBOSA, J. M. A. Simulação da influência da geração de calor na degradação de estruturas metálicas. *Mecânica Computacional Vol. XXII*, Bahía Blanca, Argentina, 2003.

[30] SANTOS, F. C. G.; BRITO, E. R. R. J.; BARBOSA, J. M. A. Simulação do problema de evolução do dano em uma barra elasto-viscoplástica com acoplamento termomecânico empregando grafo de interface genérica (gig). 7° *Congresso Iberoamericano de Engenharia Mecânica*, México, 2005.

[31] REDDY, J. N.; GARTLING, D. K. The finite element method in heat transfer and fluid dynamics. CRC Press, 1994.

[32] FREY, P. J.; GEORGE, P. L. Mesh generation application to finite elements. Hermes Science, 2000.

APÊNDICE A

Solução Analítica para o Problema de Elasticidade com Simetria Axial

Neste apêndice será desenvolvida uma solução analítica para o problema de difusão com simetria axial. Considere a Equação 4.78 então, utilizando a Equação 4.70 e desconsiderando o termo do contorno e o termo referente a difusão do soluto tem-se que

$$\int_{\Omega} \varepsilon(\mathbf{w})^T \cdot \mathbf{C}\varepsilon(\mathbf{u}) = \int_{\Omega} \mathbf{b} \cdot \mathbf{w} \, \mathrm{d}\Omega \tag{A.1}$$

Devido a simetria axial plana, a Equação 4.70 será redefinida como

$$\varepsilon(\bullet) = \begin{bmatrix} \varepsilon_{rr}(\bullet) \\ \varepsilon_{\theta\theta}(\bullet) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial r}(\bullet) \\ \frac{(\bullet)}{r} \end{bmatrix} \tag{A.2}$$

e

$$\mathbf{C} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu \\ \nu & 1 \end{bmatrix} \tag{A.3}$$

Escrevendo a integral na Equação A.1 em coordenadas cilíndricas com r variando de R_0 até R_1 e θ variando de θ_0 até θ_1 então

$$\int_{R_0}^{R_1} \int_{\theta_0}^{\theta_1} \varepsilon(\mathbf{w})^T \cdot \mathbf{C}\varepsilon(\mathbf{u}) \, r\mathrm{d}\theta\mathrm{d}r = \int_{R_0}^{R_1} \int_{\theta_0}^{\theta_1} \mathbf{b} \cdot \mathbf{w} \, r\mathrm{d}\theta\mathrm{d}r \tag{A.4}$$

onde

$$\mathbf{b} = f_r \begin{bmatrix} \cos\theta \\ \mathrm{sen}\theta \end{bmatrix} \tag{A.5}$$

Utilizando as Equações A.2 e A.3 na Equação A.4 e considerando a simetria axial, então

$$\frac{E}{1-\nu^2} \int_{R_0}^{R_1} \left[\frac{u_r w_r}{r^2} + \frac{\partial u_r}{\partial r}\frac{\partial w_r}{\partial r} + \nu\frac{1}{r}\frac{\partial}{\partial r}(u_r w_r)\right] r\mathrm{d}r = \int_{R_0}^{R_1} w_r f_r \, r\mathrm{d}r \tag{A.6}$$

de onde tem-se que

$$\frac{E}{1-\nu^2} \int_{R_0}^{R_1} \left[\frac{u_r w_r}{r^2} + \frac{\partial u_r}{\partial r}\frac{\partial w_r}{\partial r}\right] r\mathrm{d}r + \frac{E\nu}{1-\nu^2} \, u_r v_r|_{R_0}^{R_1} = \int_{R_0}^{R_1} w_r f_r \, r\mathrm{d}r \tag{A.7}$$

Como $v_r = 0$ onde for prescrito condição de Dirichlet (será considerado Dirichlet em R_0 e R_1) então a Equação A.7 fica da forma

$$\int_{R_0}^{R_1} \{-\frac{E}{1-\nu^2}[-\frac{u_r}{r}+\frac{\partial}{\partial r}(r\frac{\partial u_r}{\partial r})]-f_r r\}\, w_r \mathrm{d}r = 0 \tag{A.8}$$

A Equação A.8 deve ser satisfeita para todo w_r, então tem-se que

$$-\frac{E}{1-\nu^2}[-\frac{u_r}{r}+\frac{\partial}{\partial r}(r\frac{\partial u_r}{\partial r})] = f_r r \tag{A.9}$$

a qual é a Equação diferencial para os deslocamento com simetria axial. Esta Equação pode ser escrita da seguinte forma

$$-\frac{E}{1-\nu^2}[\frac{\partial}{\partial r}(\frac{u_r}{r}+\frac{\partial u_r}{\partial r})] = f_r \tag{A.10}$$

A solução analítica para u_r é obtida integrando-se duas vezes a Equação A.10 (considerando f_r constante), então

$$u_r = -\frac{f_r(1-\nu^2)}{3E}r^2 + \frac{c_1 r}{2} + \frac{c_2}{r} \tag{A.11}$$

onde c_1 e c_2 são constantes obtidas a partir das condições de contorno.

Como foi considerado que o problema de elasticidade não está acoplado com o problema de difusão, então tem-se que

$$\mathrm{tr}(\mathbf{S}) = \mathrm{tr}(\sigma) \tag{A.12}$$

onde σ é dado por

$$\sigma = \begin{bmatrix} \sigma_{rr} \\ \sigma_{\theta\theta} \end{bmatrix} = \mathbf{C}\begin{bmatrix} \varepsilon_{rr}(u_r) \\ \varepsilon_{\theta\theta}(u_r) \end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix} \frac{\partial u_r}{\partial r}+\nu\frac{u_r}{r} \\ \nu\frac{u_r}{\partial r}+\frac{u_r}{r} \end{bmatrix} \tag{A.13}$$

então,

$$\frac{\partial}{\partial r}\mathrm{tr}(\sigma) = \frac{\partial}{\partial r}(\sigma_{rr}+\sigma_{\theta\theta}) = \frac{E(1+\nu)}{1-\nu^2}\frac{\partial}{\partial r}(\frac{u_r}{r}+\frac{\partial u_r}{\partial r}) \tag{A.14}$$

Utilizando a Equação A.11 na Equação A.14 finalmente chega-se a

$$\frac{\partial}{\partial r}\mathrm{tr}(\sigma) = -(1+\nu)f_r \tag{A.15}$$

APÊNDICE B

Solução Analítica para o Problema de Difusão com Simetria Axial

Neste apêndice será desenvolvida uma solução analítica para o problema de difusão com simetria axial. Considerando a Equação 4.42, mas desconsiderando o dano e introduzindo o termo de geração de soluto f, então tem-se que:

$$\int_\Omega \dot{c}v \, \mathrm{d}\Omega - \int_\Omega \nabla \cdot (D\nabla c)v \, \mathrm{d}\Omega + \int_\Omega \nabla \cdot (\frac{cD}{\alpha}\nabla(e\mathrm{tr}(\mathbf{S})))v \, \mathrm{d}\Omega = \int_\Omega fv \, \mathrm{d}\Omega \tag{B.1}$$

O termo de geração de soluto foi introduzido para auxiliar na obtenção de uma solução analítica. Considerando

$$\nabla c = \begin{bmatrix} \frac{\partial c}{\partial x} \\ \frac{\partial c}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial c}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial c}{\partial \theta}\frac{\partial \theta}{\partial x} \\ \frac{\partial c}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial c}{\partial \theta}\frac{\partial \theta}{\partial y} \end{bmatrix} \tag{B.2}$$

onde

$$\frac{\partial r}{\partial x} = \cos\theta \tag{B.3}$$

$$\frac{\partial r}{\partial y} = \mathrm{sen}\theta \tag{B.4}$$

e como está sendo considerada uma simetria axial tem-se que

$$\frac{\partial c}{\partial \theta} = 0 \tag{B.5}$$

Então a Equação B.2 fica da seguinte forma

$$\nabla c = \begin{bmatrix} \cos\theta \\ \mathrm{sen}\theta \end{bmatrix} \frac{\partial c}{\partial r} \tag{B.6}$$

O termo $\nabla \cdot (D\nabla c)$ é dado por

$$\nabla \cdot (D\nabla c) = \frac{\partial}{\partial x}(D\frac{\partial c}{\partial x}) + \frac{\partial}{\partial y}(D\frac{\partial c}{\partial y}) \tag{B.7}$$

Desenvolvendo a Equação B.7 tem-se que

$$\nabla \cdot (D\nabla c) = \frac{\partial}{\partial r}(D\frac{\partial c}{\partial x})\frac{\partial r}{\partial x} + \frac{\partial}{\partial \theta}(D\frac{\partial c}{\partial x})\frac{\partial \theta}{\partial x} + \frac{\partial}{\partial r}(D\frac{\partial c}{\partial y})\frac{\partial r}{\partial y} + \frac{\partial}{\partial \theta}(D\frac{\partial c}{\partial y})\frac{\partial \theta}{\partial y} \tag{B.8}$$

onde

$$\frac{\partial \theta}{\partial x} = -\frac{\text{sen}\theta}{r} \tag{B.9}$$

$$\frac{\partial \theta}{\partial y} = \frac{\cos\theta}{r} \tag{B.10}$$

Utilizando as Equações B.3, B.4, B.6, B.9 e B.10 na Equação B.8, então

$$\nabla \cdot (D\nabla c) = \frac{1}{r}\frac{\partial}{\partial r}(rD\frac{\partial c}{\partial r}) \tag{B.11}$$

O termo $\nabla \cdot (cD\nabla \text{tr}(\mathbf{S}))$ é dado por

$$\nabla \cdot (cD\nabla \text{tr}(\mathbf{S})) = \nabla \cdot (cD \begin{bmatrix} \cos\theta \\ \text{sen}\theta \end{bmatrix} \frac{\partial}{\partial r}\text{tr}(\mathbf{S})) \tag{B.12}$$

Desenvolvendo a Equação B.12 então

$$\nabla \cdot (cD\nabla \text{tr}(\mathbf{S})) = \frac{1}{r}\frac{\partial}{\partial r}(rcD\frac{\partial}{\partial r}\text{tr}(\mathbf{S})) \tag{B.13}$$

Usando as Equações B.6, B.11 e B.13 na Equação B.1 então

$$\int_\Omega \dot{c}v \, \mathrm{d}\Omega - \int_\Omega \frac{1}{r}\frac{\partial}{\partial r}[rD(\frac{\partial c}{\partial r} - c\frac{e}{\alpha}\frac{\partial}{\partial r}\text{tr}(\mathbf{S}))]v \, \mathrm{d}\Omega - \int_\Omega fv \, \mathrm{d}\Omega = 0 \tag{B.14}$$

Escrevendo a integral na Equação B.14 em coordenadas cilíndricas com r variando de R_0 até R_1 e θ variando de θ_0 até θ_1 então

$$\int_{R_0}^{R_1}\int_{\theta_0}^{\theta_1} \dot{c}v \, r\mathrm{d}\theta\mathrm{d}r - \int_{R_0}^{R_1}\int_{\theta_0}^{\theta_1} \frac{1}{r}\frac{\partial}{\partial r}[rD(\frac{\partial c}{\partial r} - c\frac{e}{\alpha}\frac{\partial}{\partial r}\text{tr}(\mathbf{S}))]v \, r\mathrm{d}\theta\mathrm{d}r - \\ - \int_{R_0}^{R_1}\int_{\theta_0}^{\theta_1} fv \, r\mathrm{d}\theta\mathrm{d}r = 0 \tag{B.15}$$

Como está sendo considerado uma simetria axial então a Equação B.15 resulta em

$$\int_{R_0}^{R_1} \{\dot{c}r - \frac{\partial}{\partial r}[rD(\frac{\partial c}{\partial r} - c\frac{e}{\alpha}\frac{\partial}{\partial r}\text{tr}(\mathbf{S}))] - fr\}v \, \mathrm{d}r = 0 \tag{B.16}$$

A Equação B.16 deve ser satisfeita para qualquer v, então tem-se que

$$\dot{c}r - \frac{\partial}{\partial r}[rD(\frac{\partial c}{\partial r} - c\frac{e}{\alpha}\frac{\partial}{\partial r}\text{tr}(\mathbf{S}))] = fr \tag{B.17}$$

a qual é a Equação diferencial para a difusão do soluto com simetria axial. Utilizando as Equações A.12 e A.15 na Equação B.17 então

$$\dot{c}r - \frac{\partial}{\partial r}[rD(\frac{\partial c}{\partial r} - c\frac{e}{\alpha}(1+\nu)f_r)] = fr \tag{B.18}$$

Uma solução para a Equação B.18 é dada por

$$c(r,t) = \frac{w_0 g}{\lambda D} e^{-\lambda D t} [Ar^2 + Br + C] \tag{B.19}$$

e o termo de geração de soluto que deve ser utilizado em conjunto com essa solução é dado por

$$f(r,t) = \frac{w_0 g}{r} e^{-\lambda D t} (r^n + h) \tag{B.20}$$

onde w_0, g e λ são cosntantes a serem escolhidas e A, B,C e h são constantes que devem ser calculadas e n deve ser escolhido entre $n = 0,1,2,3$. O cálculo de A, B,C e h deve ser realizado substituindo-se as Equações B.19 e B.20 na Equação B.18 e escolhendo um valor para n. Utilizando $n = 1$ e escolhendo $\lambda = 0,5$, $g = -2$ e $w_0 = 1$ obtém-se

$$A = 0 \tag{B.21}$$

$$B = 0 \tag{B.22}$$

$$C = -1 \tag{B.23}$$

$$h = \frac{e f_r (1+\nu)}{\lambda \alpha} \tag{B.24}$$

a qual é a solução utilizada na seção 5.2.2. Utilizando $n = 2$ e escolhendo $\lambda = 0,5$, $g = -2$ e $w_0 = 1$ obtém-se

$$A = 0 \tag{B.25}$$

$$B = -1 \tag{B.26}$$

$$C = \frac{2 e f_r}{\alpha \lambda} (1+\nu) \tag{B.27}$$

$$h = \frac{-[-\lambda + 2(\frac{e f_r (1+\nu)}{\alpha})^2]}{\lambda^2} \tag{B.28}$$

a qual é a solução utilizada na seção 5.2.3.