UNIVERSIDADE DE SÃO PAULO
Escola de Engenharia de São Carlos

Proposta de plataforma orientada a objetos para a resolução de problemas mecânicos por meio do MEF Posicional

ALEXANDRE TEN CATE MATTÉ

Dissertação de Mestrado do Programa de Pós-Graduação em Engenharia Civil (Engenharia de Estruturas) da Escola de Engenharia de São Carlos, Universidade de São Paulo

EESC · USP

ALEXANDRE TEN CATE MATTÉ

Proposta de plataforma orientada a objetos para a resolução de problemas mecânicos por meio do MEF Posicional

VERSÃO CORRIGIDA

(A versão original encontra-se na Escola de Engenharia de São Carlos)

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Civil (Engenharia de Estruturas) da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos necessários para obtenção do título de Mestre em Ciências.

Área de concentração: Estruturas

Orientador: Prof. Dr. Rogério Carrazedo

São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues
Fontes da EESC/USP

M435p

Matté, Alexandre Ten Cate
Proposta de plataforma orientada a objetos para a resolução de problemas mecânicos por meio do MEF posicional / Alexandre Ten Cate Matté; orientador Rogério Carrazedo. -- São Carlos, 2024.

Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Civil (Engenharia de Estruturas) e Área de Concentração em Estruturas -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2024.

1. Método dos elementos finitos posicional. 2. Arquitetura de código. 3. Orientação a objetos. 4. Escalabilidade. 5. Programação. 6. C++. I. Título.

Elaborado por Elena Luzia Palloni Gonçalves – CRB 8/4464

FOLHA DE JULGAMENTO

Candidato: Engenheiro **ALEXANDRE TEN CATE MATTÉ**.

Título da dissertação: "Proposta de plataforma orientada a objetos para a resolução de problemas mecânicos por meio do MEF Posicional".

Data da defesa: 10/12/2024.

Comissão Julgadora	**Resultado**
Prof. Associado Rogério Carrazedo (Orientador) (Escola de Engenharia de São Carlos/EESC-USP)	Aprovado
Prof. Dr. Ricardo Afonso Angélico (Escola de Engenharia de São Carlos/EESC-USP)	Aprovado
Prof. Dr. Daniel Nelson Maciel (Universidade Federal do Rio Grande do Norte/UFRN)	Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Civil (Engenharia de Estruturas):
Prof Associado **Ricardo Carrazedo**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Carlos De Marqui Junior**

Este trabalho é dedicado à minha avó, Lilia Maria (in memoriam), por toda a sua dedicação e cuidado, fortalecendo sempre os valores de uma família unida, repleta de amor e carinho.

AGRADECIMENTOS

Aos meus pais, Narciso e Mara, que formaram grande parte do meu caráter e contribuíram muito na minha formação, sempre me incentivando a alcançar meus objetivos e dando todo o suporte possível.

À minha avó Lilia Maria (in memoriam), por ter sido minha primeira professora, antes mesmo de ir para a escola, despertando em mim, desde cedo, o gosto pelo estudo.

Ao meu irmão, Marcelo, pela parceria de sempre, por servir de exemplo e me orientar, sempre que possível, nos mais diversos assuntos.

À minha noiva, Priscilla, por todo amor, carinho e dedicação, que me motiva a dar o meu melhor todos os dias.

Aos meus afilhados, Gabriel e Lucas, pela alegria que proporcionam a toda família e pela compreensão das minhas ausências em momentos importantes.

Ao meu sócio e amigo Diego, por ter sido meu mentor, contribuindo muito para minha formação profissional, pela grande amizade e por estar sempre disposto a ajudar.

Ao meu orientador, Rogério, pela paciência, compreensão, e por todo o tempo reservado para partilhar do seu conhecimento e experiência.

À toda equipe da MDC Projetos e da Authenty, por estarem sempre dispostos a ajudar e por tornarem meus dias de trabalho cada vez melhores.

Aos amigos e colegas do SET, especialmente aos colegas de sala, Felipe Macedo, Felipe Sasso, Juan Ibiapina e Matheus Alves, pela parceria e pelo ambiente de trabalho leve e descontraído.

A todos os meus professores, desde a educação infantil até os dias atuais, por tantos ensinamentos, que foram essenciais nas diferentes etapas da minha formação.

"a puzzling limitation of our mind: our excessive confidence in what we believe we know and our apparent inability to acknowledge the full extent of our ignorance and the uncertainty of the world we live in."

Daniel Kahneman

RESUMO

MATTE, A.t.C **Proposta de plataforma orientada a objetos para a resolução de problemas mecânicos por meio do MEF posicional**. 2024. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2024.

Este trabalho trata de uma proposta de arquitetura de código, orientada a objetos, para a resolução de problemas mecânicos, por meio do Método dos Elementos Finitos (MEF) Posicional. A implementação englobou problemas quase estáticos, dinâmicos, e com a imersão de elementos na malha. Foram empregados elementos bidimensionais, triangulares, de aproximações linear, quadrática e cúbica, e elementos de barra simples, com aproximação linear. O código foi desenvolvido na linguagem de programação C++, visando o aproveitamento de características importantes que a orientação a objetos proporciona, como manutenibilidade e escalabilidade. Por tratar, geralmente, de problemas com grandes dimensões, a arquitetura proposta apresenta ambientes bem definidos de pré processamento, processamento e pós processamento, evitando ao máximo a movimentação desnecessária de dados. A documentação foi inserida diretamente no código, e gerada automaticamente por meio do *software* Doxygen®. A plataforma foi disponibilizada por meio de um repositório no GitHub® e publicada por meio da plataforma Zenodo®, totalmente *open source*. Foram apresentados diversos exemplos de aplicação, validando a implementação para todos os tipos de problemas propostos.

Palavras-chave: Método dos elementos finitos posicional. Arquitetura de código. Orientação a objetos. Escalabilidade. Programação. C++.

ABSTRACT

MATTE, A.t.C **Object oriented platform proposal for mechanic problems resolution through posicional FEM**. 2024. Dissertation (Master) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2024.

This work presents an object-oriented code architecture proposal for solving mechanical problems using the Positional Finite Element Method (FEM). The implementation covers quasi-static, dynamic problems, and problems with immersed elements in the mesh. Two-dimensional triangular elements with linear, quadratic, and cubic approximations, as well as simple bar elements with linear approximation, were employed. The code was developed in C++ to leverage the important features that object-oriented programming provides, such as maintainability and scalability. Since it generally deals with large-scale problems, the proposed architecture features well-defined environments for pre-processing, processing, and post-processing, minimizing unnecessary data movement. Documentation was directly embedded in the code and automatically generated using Doxygen® software. The platform was made available through a GitHub® repository and published via Zenodo®, completely open-source. Several application examples were presented, validating the implementation for all types of proposed problems.

Keywords: Positional finite element method. Code architecture. Object orientation. Scalability. Programming. C++.

LISTA DE FIGURAS

SUMÁRIO

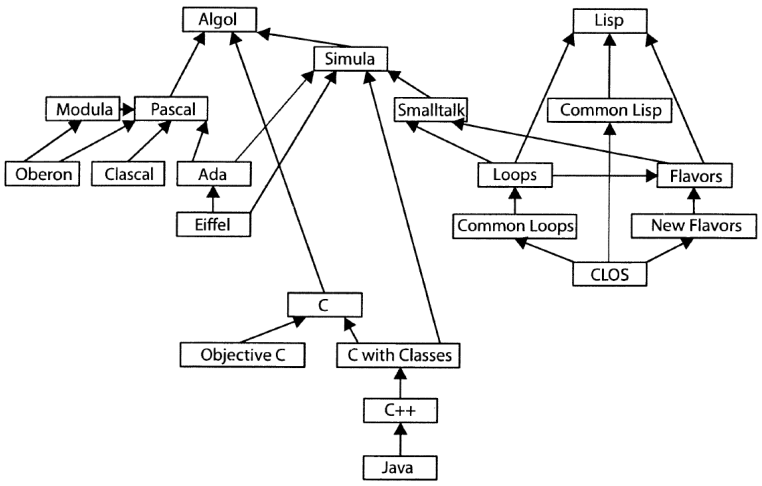
1 INTRODUÇÃO

Os códigos que envolvem a resolução de problemas com o Método dos Elementos Finitos (MEF) costumam apresentar alto nível de complexidade. Muitos deles são construídos sem a devida preocupação com futuros desdobramentos ou extensões, o que acaba por dificultar, ou até mesmo inviabilizar, uma futura adaptação ou atualização. Devido a sua grande extensão acabam por limitar os profissionais habilitados à realizar modificações e ajustes, exigindo o conhecimento de grande parte, ou até a totalidade da teoria envolvida na sua concepção (MCKENNA, 1997).

Para que um código seja facilmente adaptável, é de extrema importância que, na fase de projeto, sejam elencadas não somente as suas funcionalidades atuais, como também uma série de funcionalidades que poderão complementá-lo futuramente. Também é imprescindível que sejam seguidos alguns princípios que tornem as futuras alterações menos trabalhosas, como políticas de modularização do sistema, facilitando a localização de elementos específicos (ANICHE; YODER; KON, ; KULESZA, 2000).

Os avanços na engenharia de software proporcionaram uma série de ferramentas primordiais para a reconstrução da arquitetura geral desses códigos. No início da década de 80, com o surgimento das primeiras linguagens orientadas a objetos – primeiramente a SIMULA 67 (LAMPRECHT, 1983), e depois uma série de outras linguagens, com ênfase para C++, que é amplamente utilizada ainda na atualidade – abrem-se novas oportunidades de pesquisa nessa área. Na Figura 1.1 pode-se observar a evolução de diversas linguagens e a conexão entre elas.

Figura 1.1 – Evolução de várias linguagens de programação orientadas a objetos



Fonte: Seed (1996).

Na simbologia utilizada por Seed (1996), similar à simbologia de herança entre classes, as setas conectam as linguagens que tiveram influência, ou até serviram de base para a criação de novas linguagens, apontando sempre para as predecessoras. A linguagem C++, por exemplo, foi amplamente influenciada pela linguagem Simula (SEED, 1996).

No início da década de 90 começam a surgir os primeiros trabalhos referentes ao MEF utilizando o paradigma de orientação a objetos (OHTSUBO; KAWAMURA; KUBOTA, 1993; SCHOLZ, 1992). Todavia, os métodos numéricos também não ficaram estagnados no tempo. Foram criados métodos completamente distintos, e os métodos existentes foram sendo continuamente reformulados, adaptados e aprimorados. Isso ocorre devido a uma série de fatores, dentre os quais podem ser citados a resolução de novos problemas de engenharia, a avaliação de diferentes tipos de materiais, as melhorias relacionadas ao desempenho, precisão, convergência, assim como a melhoria dos equipamentos utilizados para análise, dentre outros fatores.

Nessa linha que surge o Método dos Elementos Finitos Posicional (MEFP), que vem sendo desenvolvido desde Coda (2003), no intuito de simplificar a implementação dos códigos para problemas com consideração de não linearidade geométrica. Uma série de trabalhos vem sendo desenvolvidos para comprovar a eficiência do método em diferentes problemas de engenharia. Esses trabalhos tratam de estruturas de pórtico, treliçadas bidimensionais (CODA; GRECO, 2004) e tridimensionais (GRECO *et al.*, 2006), cascas (CODA; PACCOLA, 2007; 2008), etc., envolvendo modelos quase-estáticos, dinâmicos (GRECO; CODA, 2006), termomecânicos (CARRAZEDO, 2009; CARRAZEDO; CODA, 2010) e de interação fluido-estrutura (SANCHES; CODA, 2013; 2014). Também há estudos sobre a consideração de materiais heterogêneos (CARRAZEDO; PACCOLA; CODA, 2018; NOGUEIRA; PACCOLA; CODA, 2016; PACCOLA; CODA, 2016), por meio de elementos imersos na malha, a consideração do comportamento viscoelástico dos materiais (RABELO *et al.*, 2018), de danos causados por corrosão (RAMOS; CARRAZEDO, 2020), dentre outros estudos.

Muitos desses trabalhos envolvendo novas formulações, adaptações e melhorias dos métodos foram sendo desenvolvidos com foco na resolução do problema de engenharia, sem grandes preocupações com a estrutura do código. Dentro desse contexto, a estratégia de plataforma se mostra uma alternativa muito interessante em diversas frentes. O termo plataforma é utilizado para descrever um conjunto de subsistemas que torna possível o desenvolvimento de uma determinada gama de produtos. O conceito trata de reutilizar grande parte do código, deixando apenas a customização dos trechos necessários para o desenvolvimento de produtos específicos (GHANAM; MAURER; ABRAHAMSSON, 2012). Nesse sentido, a plataforma promove maior integração entre os trabalhos da mesma área, otimizando o tempo dedicado de cada profissional, evitando retrabalhos com repetição de códigos, e proporcionando uma padronização de desenvolvimento, tornando os trabalhos

mais acessíveis a novos pesquisadores.

Todavia, a construção de uma plataforma envolve uma série de desafios. Conforme Sanner e Nielsen (2019), a capacidade de inovação de uma plataforma de *software* depende muito de sua arquitetura. A sua estrutura deve ser organizada e pensada para comportar futuros desenvolvimentos. Além disso, a comunicação é um fator imprescindível, devendo a documentação ser clara e suficiente, para que novos desenvolvedores não tenham dificuldades de implementação.

Em vista disso, este trabalho trata do desenvolvimento de uma plataforma voltada à resolução de problemas mecânicos, por meio do MEF Posicional, no paradigma orientado a objeto, visando uma estrutura facilmente escalável, adaptável e bem documentada, tornando possível a sua utilização em futuras aplicações do método, assim como a continuidade de atualizações.

1.1 Metodologia

Com relação à formulação matemática para a elaboração dos códigos, foi utilizado como base o material de Coda (2018). A plataforma foi projetada para ser facilmente adaptável, podendo abranger uma série de problemas distintos (envolvendo alterações de materiais, elementos, condições de contorno, etc.). Todavia, o trabalho se restringiu à resolução de problemas mecânicos estáticos e dinâmicos. Também foi prevista a resolução com elementos compósitos, nos quais são imersos elementos de mesma dimensionalidade da malha base, ou de dimensionalidade inferior. Foram utilizados elementos de chapa triangulares, com aproximação linear, quadrática e cúbica. A integração numérica foi feita na forma de quadratura de Hammer (HAMMER; MARLOWE; STROUD, 1956), onde serão empregados modelos com 7 e 12 pontos de integração.

O código foi escrito na linguagem C++, e sua estrutura foi desenvolvida com base no paradigma de orientação a objetos. A documentação do código foi elaborada através do software Doxygen®[1]. A disponibilização do repositório O2P2: Pre-alpha release (CARRAZEDO *et al.*, 2023) está sendo feita por meio do GitHub®[2].

1.2 Estrutura do trabalho

No capítulo 2 é apresentado um breve histórico de pesquisas e o atual estado da arte referente aos assuntos abordados. Os principais temas tratados foram o MEF Posicional e a programação orientada a objetos. Também são abordadas as estruturas de código utilizadas atualmente para a resolução desses problemas, com ênfase naqueles desenvolvidos segundo o paradigma orientado a objetos. No capítulo 3 é apresentada a

[1] Disponível em: <https://doxygen.nl/index.html/>. Versão: 1.9.8. Acesso em: Outubro de 2023.

[2] Disponível em: <https://github.com/>. Acesso em: Outubro de 2023.

teoria do método dos elementos finitos posicional, que foi utilizada como base para a elaboração do código. O trabalho trata de problemas mecânicos estáticos, dinâmicos e também com a inserção de elementos imersos na malha.

O capítulo 4 traz os principais conceitos referentes à teoria de programação orientada a objetos, que serviram de base para a lógica de desenvolvimento do sistema. Nesse mesmo capítulo, a estrutura geral do código é apresentada, elencando as principais classes e as interações entre elas. Todas as classes que compõem o código foram descritas, trazendo suas principais características e funcionalidades dentro do sistema. Também é exposto o processo de documentação e disponibilização, onde são apresentadas as ferramentas utilizadas, tanto para a geração dos textos da documentação quanto para a disponibilização do código. Por fim, o capítulo ainda trata do desenvolvimento da estrutura padrão utilizada para a documentação, de forma a simplificar não só o entendimento geral do sistema, mas também de cada uma das funções componentes do código.

No Capítulo 5 são apresentados problemas já discutidos em outros trabalhos, com resultados conhecidos, com a finalidade de validar o código desenvolvido. Por fim, o Capítulo 6 traz as conclusões a partir dos resultados obtidos e discutidos, em conformidade com os objetivos geral e específicos do trabalho, citados no Capítulo 1.

1.3 Delimitação do tema

Desenvolvimento de plataforma genérica orientada a objetos na linguagem C++ aplicada ao MEF Posicional (não linear geométrico), sendo implementada, neste trabalho, para problemas mecânicos estáticos e dinâmicos com a consideração de elementos imersos, utilizando o modelo constitutivo de Saint-Venant-Kirchhoff.

1.4 Objetivos

O objetivo deste trabalho é desenvolver uma plataforma no paradigma orientado a objetos por meio do MEF posicional para análise mecânica estática e dinâmica de estruturas, com a consideração de elementos imersos na matriz.

1.4.1 Objetivos específicos

a) Organizar a arquitetura do código com uma visão generalista, de forma a possibilitar a resolução de diferentes problemas;

b) Implementar classes de objetos que sejam independentes do problema a ser resolvido, de forma a facilitar a inclusão de novas classes a posteriori.

c) Desenvolver códigos modulares, facilmente adaptáveis;

d) Documentar todas as funções, de forma clara e concisa, para que a plataforma possa ser utilizada de forma trivial como base para o desenvolvimento de novas ferramentas;

e) Disponibilizar o código para a comunidade científica através do repositório GitHub®.

1.5 Justificativa

No desenvolvimento de códigos computacionais voltados à resolução de problemas específicos de engenharia é comum que o pesquisador construa, inicialmente, um código base, tornando possível a evolução de sua pesquisa. A existência de uma plataforma com características genéricas, com estrutura organizada e documentada se torna uma ferramenta primordial para a elaboração de validações dessas pesquisas, evitando retrabalhos e gastos excessivos de tempo de programação. Desta forma, o pesquisador pode dedicar mais tempo na parte teórica e na implementação de um trecho específico de código. Além disso, caso seja seguida a mesma linha de raciocínio nesses trabalhos, os códigos serão facilmente integrados à plataforma, e poderão ser utilizados por futuros pesquisadores. À vista disso, este trabalho pretende desenvolver uma plataforma genérica, aplicada ao MEF Posicional, segundo o paradigma de orientação a objetos, de maneira que sua estrutura seja de fácil escalabilidade, adaptação e de fácil entendimento, possibilitando o seu aprimoramento contínuo.

2 ESTADO DA ARTE

Este capítulo tem como objetivo apresentar o atual estado da arte relacionado ao tema em questão. Em um primeiro momento são discutidos aspectos referentes ao MEF, com foco no MEF Posicional. Mais à frente é abordado o emprego de programação orientada a objetos na implementação desse tipo de código.

2.1 O Método dos Elementos Finitos Posicional

O método dos elementos finitos (MEF) foi desenvolvido na década de 50 e no início dos anos 60, quando passou a ser visto como uma técnica numérica viável, sendo utilizado principalmente para estruturas aeronáuticas. A partir daí foi ganhando espaço em diferentes indústrias, como a automobilística, construção civil, biomecânica, geotécnica, entre outras. O tipo de análise também foi sendo diversificado com o passar dos anos, destacando-se as análises mecânicas (estáticas e dinâmicas), térmicas, de campos elétricos e magnéticos, de escoamento de fluidos, etc. (BOWER, 2010; FISH; BELYTSCHKO, 2007; OÑATE *et al.*, 2004; WRIGGERS, 2008; ZIENKIEWICZ; HUANG; LIU, 1990).

Observando as limitações do MEF, foram surgindo novos métodos, sendo alguns deles com poucas adaptações na sua formulação, e outros com uma concepção totalmente distinta, com o objetivo de atender a necessidades específicas de cada aplicação. No geral há uma série de vantagens e desvantagens quando comparamos essas metodologias, que acabam nos direcionando conforme o tipo de problema a ser analisado. Citando alguns exemplos, de forma resumida, o Método dos Elementos de Contorno (MEC) apresenta uma concepção totalmente distinta, mais complexa, resultando em uma matriz densa e assimétrica. Todavia, o método proporciona uma diminuição do tempo de modelagem do problema, uma vez que a sua dimensão é reduzida em uma ordem (problemas tridimensionais são descritos por meio de uma superfície de contorno, enquanto problemas bidimensionais são descritos por meio de uma linha de contorno). Essas características são favoráveis em análise de fratura, por exemplo, devido à necessidade contínua de reconstrução da malha (BECKER, 1992). Outro exemplo mais recente é o Método dos Elementos Finitos Generalizados (MEFG), que foi desenvolvido na década de 90 (DUARTE; BABUŠKA; ODEN, 2000). O método é baseado no MEF tradicional, porém suas funções de enriquecimento possibilitam a obtenção de resultados mais precisos em regiões particulares do domínio, como fissuras e regiões de contato, dispensando assim um refinamento excessivo da malha (PIEDADE NETO, 2013).

Com os avanços da engenharia estrutural, em busca de otimização de recursos, e também de estruturas diferenciadas, tornam-se mais comuns os projetos envolvendo estruturas com menor rigidez. Desse modo surge a necessidade da utilização de métodos

de cálculo adequados para estruturas que apresentam grandes deslocamentos. Com base nisso surge o Método dos Elementos Finitos Posicional (BONET *et al.*, 2000/ CODA, 2003; 2018), apresentando, em sua implementação, uma metodologia mais simples e direta para a consideração da não linearidade geométrica dentro do MEF. Baseado no princípio da estacionariedade da energia, o método se diferencia principalmente por utilizar a posição como variável incógnita. A formulação é não-linear geométrica de forma natural, apresentando vantagens nas análises com grandes deslocamentos, rotações e deformações, por considerar a cinemática exata do corpo sólido (CARVALHO, 2019; SIQUEIRA, 2019). A partir de sua formalização, realizada por Coda (2003), o método já teve várias aplicações em diferentes tipos de problemas. Pode-se citar alguns dos trabalhos pioneiros, como o desenvolvido por Coda e Greco (2004), em estruturas bidimensionais compostas por elementos de pórtico, e o trabalho de Greco *et al.* (2006), simulando treliças tridimensionais, onde o método se mostrou eficiente quanto à sua convergência e precisão, enfatizando ainda a capacidade de avaliação das estruturas após a ocorrência de instabilidades geométricas. Na pesquisa desenvolvida por Greco e Coda (2006), voltada para multicorpos flexíveis, foi demonstrado que a formulação é adequada para problemas dinâmicos com grandes deslocamentos e grandes rotações, considerando não linearidade geométrica.

A formulação seguiu sendo aplicada com êxito em diferentes tipos de problemas. Pode-se citar os trabalhos de Coda e Paccola (2007, 2008) envolvendo a análise de cascas, os de Carrazedo (2009) e Carrazedo e Coda (2010) com a consideração de efeitos termomecânicos, e os de Sanches e Coda (2013, 2014) e Fernandes, Coda e Sanches (2019), demonstrando a aplicabilidade da formulação em problemas de interação fluido-estrutura, e de Avancini e Sanches (2020) em problemas de escoamentos de fluidos em superfície livre. Como exemplo de aplicação em problemas de impacto, menciona-se o trabalho de Cavalcante, Maciel e Greco (2018), que avaliou a resposta dinâmica de treliças bi e tridimensionais. Com relação a formulações voltadas a materiais compósitos, pode-se citar os trabalhos de Paccola e Coda (2016), apresentando uma alternativa para a consideração de compósitos particulados, de Nogueira, Paccola e Coda (2016), propondo uma formulação para o tratamento de estruturas laminadas e de Carrazedo, Paccola e Coda (2018), com a proposição de uma estratégia numérica para consideração da rigidez de painéis sanduíche[1]. Também pode-se mencionar alguns trabalhos envolvendo a não linearidade física, como o de Reis e Coda (2014), com a implementação de conexões elastoplásticas semi-rígidas, os trabalhos de Rabelo *et al.* (2018), avaliando o comportamento mecânico viscoelástico em treliças tridimensionais, o de Pascon (2022), tratando de um modelo de dano com grandes deformações, e o de Ramos e Carrazedo (2020), propondo uma estratégia alternativa de expansão e dano de seções transversais de concreto causados pela corrosão não uniforme associados ao ingresso de cloretos. O trabalho de Carvalho, Coda e Sanches (2020) trata

[1] Constituídos normalmente por duas chapas de pequena espessura e maior rigidez, separadas por uma camada espessa de um material leve, de menor rigidez.

de problemas envolvendo contato, avaliando o processso de conformação a frio. Ainda podem ser citados os trabalhos de Soares, Paccola e Coda (2019, 2021), sobre análise de instabilidade em estruturas de parede fina e o de Paulino e Leonel (2021), sobre otimização topológica, entre outras abordagens, que corroboram a utilidade do método para inúmeros problemas de engenharia.

2.2 Programação orientada a objetos no MEF

Desde o período inicial das aplicações do MEF surge a problemática relacionada à complexidade de seus códigos computacionais. Na maioria dos problemas há a necessidade de utilização de uma quantidade elevada de variáveis, que devem ser relacionadas de diferentes formas, em trechos distintos do código. Dessa forma, caso não seja dada a devida atenção à sua organização, os códigos se tornam repetitivos, e apresentam dificuldades quanto a sua alteração e escalabilidade. Nas primeiras versões do software NASTRAN já ficava clara a preocupação de Wills e Roe (1972) com questões relacionadas a escalabilidade e facilidade de adaptação dos códigos, assim como com a sua documentação. Outros trabalhos realizados na mesma época demonstram preocupações similares, como é o caso de Dodds e Lopez (1978), criando uma discussão a respeito de análises não lineares, e de Iwaki, Maeda e Ishii (1979), com foco no alto desempenho, no desenvolvimento do software Mitsui Structural Analysis System (MISA).

Com o avanço das linguagens de programação e o surgimento da orientação a objetos na década de 80, inicialmente através da linguagem SIMULA 67 (LAMPRECHT, 1983), e posteriormente de outras linguagens, com ênfase para C++, que foi – e ainda é – amplamente utilizada, iniciaram-se os desenvolvimentos de códigos de MEF orientados a objetos no início da década de 90. Pode-se citar alguns trabalhos pioneiros, como o trabalho de Forde, Foschi e Stieme (1990), que foi o primeiro a aplicar a programação orientada a objetos em problemas relacionados ao MEF, e o trabalho de Scholz (1992), desenvolvido na linguagem C++, que apesar da simplicidade relacionado ao problema de engenharia, traz aspectos importantes com relação ao paradigma de orientação a objetos. Mackie (1992) implementou seu código empregando Turbo Pascal (BORLAND INTERNATIONAL INC., 1988), ilustrando algumas vantagens da utilização da metodologia. Zimmermann, Dubois-Pèlerin e Bomme (1992) abordaram os princípios de orientação a objetos voltados ao MEF utilizando a linguagem Smalltalk/V286 (1998) e, simultaneamente desenvolveram um protótipo utilizando a mesma linguagem, tratando da implementação dos códigos orientados a objetos (DUBOIS-PÈLERIN; ZIMMERMANN; BOMME, 1992). No ano seguinte, Dubois-Pèlerin e Zimmermann (1993) buscaram a implementação de um programa orientado a objetos preocupando-se também com a eficiência numérica, desta vez utilizando a linguagem C++. Nesse mesmo ano, Ohtsubo, Kawamura e Kubota (1993) desenvolveram o MODeling tool for Integrated Finite element analYsis (MODIFY), trazendo preocupações

relativas ao pré e pós processamento, também desenvolvido na linguagem C++.

Uma série de pesquisadores se dedicaram a esses problemas nos anos subsequentes, sendo desenvolvidos códigos orientados a objetos aplicados aos mais diversos problemas de engenharia utilizando o MEF. Mackerle (2004) realizou uma revisão bibliográfica visando trabalhos voltados a programação orientada a objetos aplicados ao MEF e ao MEC, considerando o intervalo de 1990 a 2003. Nessa revisão o autor cita mais de 100 trabalhos somente aplicados ao MEF. Li e Zhou (2013) se propuseram a montar um estado da arte relacionado a programação orientada a objetos aplicada em códigos de elementos finitos. O Quadro 2.1 tem por objetivo listar os principais trabalhos desenvolvidos segundo o paradigma de orientação a objeto aplicados ao MEF. Foi feita uma relação entre os autores, o modelo de estudo, a linguagem de programação e a abordagem utilizada, onde se nota que é recente a consideração de problemas não lineares em códigos orientados a objetos. As linguagens de programação utilizadas na implementação são adicionadas no quadro através de siglas, sendo *Object Pascal*(OP), *Smalltalk*(ST), *Matlab*(ML) e *C++*(CPP). Já as siglas NLF, NLG, NLFG e L, que estão entre parênteses, indicam, respectivamente, a consideração de não linearidade física, não linearidade geométrica, não linearidade física e geométrica (simultaneamente) e linearidade física e geométrica.

Quadro 2.1 – Principais trabalhos envolvendo Orientação a Objetos aplicado ao MEF

Autor	**Modelo**		
	Estático	Dinâmico	Térmico
Forde, Foschi e Stieme (1990)	OP(L)		
Zimmermann, Dubois-Pèlerin e Bomme (1992)	ST(L)		
Dubois-Pèlerin, Zimmermann e Bomme (1992)	ST(L)		
Mackie (1992)	OP(L)	OP(L)	
Dubois-Pèlerin e Zimmermann (1993)	CPP(L)	CPP(L)	
Ohtsubo, Kawamura e Kubota (1993)	CPP		
Zeglinski, Han e Aitchison (1994)	CPP		
Bose e Carey (1999)	CPP		
Patzák e Bittnar (2001)	CPP(NLG)	CPP(NLG)	
Jayesh, Jeyakarthikeyan e Yogeshwaran (2018)			ML
Gong *et al.* (2020)	CPP(NLFG)	CPP(NLFG)	
Ding, Yu e Bui (2020)	ML(NLF)		
Ding *et al.* (2020)	ML(NLF)	ML(NLF)	
Tan *et al.* (2021)	CPP(NLFG)	CPP(NLFG)	

Fonte: Próprio autor.

Pode-se observar, por meio do quadro acima que, com o passar do tempo, foram sendo desenvolvidos trabalhos com a consideração de não linearidade física e geométrica aplicados a diferentes modelos. Também se nota a preferência pela linguagem C++, devido principalmente à sua eficiência numérica. Conforme Dubois-Pèlerin e Zimmermann (1993),

a linguagem foi desenvolvida tendo como proposta, a combinação entre as vantagens da orientação a objetos e a eficiência numérica da linguagem C.

Alguns pesquisadores disponibilizaram seus códigos para a comunidade científica, facilitando ainda mais os avanços na área. Commend e Zimmermann (2001), por exemplo, empregaram a metodologia voltada a problemas não lineares e disponibilizaram os códigos para a comunidade através do seu próprio website. Atualmente há uma série de softwares cujos códigos são abertos, ou seja, são disponibilizados à comunidade científica por meio de repositórios públicos, sendo possível a sua utilização como bibliotecas, facilitando o desenvolvimento de novas ferramentas. Um exemplo é a Modular Finite Element Methods (MFEM), desenvolvida por Kolev e Dobrev (2010) no Lawrence Livermore National Lab. (LLNL), na linguagem C++. Outro exemplo é a biblioteca code-aster (EDF, 1989), criada pela estatal francesa Életricité de France (EDF), também na linguagem C++, porém, com alguns comandos desenvolvidos na linguagem Python, com a intenção de facilitar a sua utilização. No Brasil, pode-se citar a Interactive Structural Analysis Environment (INSANE, 2007), desenvolvida no Departamento de Engenharia de Estruturas (DEES) da Universidade Federal de Minas Gerais (UFMG).

Com base no que foi exposto neste capítulo, conclui-se que o MEF Posicional, apesar de ser mais recente, já foi amplamente empregado nos mais diversos tipos de problemas. O método se mostrou eficiente, principalmente no que diz respeito à sua utilização em problemas que necessitam da consideração da não linearidade geométrica. Levando em consideração a abordagem de orientação a objetos aplicada ao MEF, também percebe-se a grande quantidade de trabalhos publicados, com diferentes abordagens e com a utilização de diferentes linguagens de programação. Apresentado o atual estado atual da arte referente a esses temas, os capítulos a seguir apresentam a teoria utilizada como base na implementação do código, iniciando pelo MEF Posicional e, posteriormente, abrangendo programação orientada a objetos.

3 MÉTODO DOS ELEMENTOS FINITOS POSICIONAL

Este capítulo tem a finalidade de expor a formulação matemática empregada na concepção dos códigos. A abordagem do MEF baseado em posições teve início nos trabalhos de Bonet *et al.* (2000) e Coda (2003). Coda (2018) formalizou essa abordagem descrevendo o método para vários tipos de elementos, trazendo exemplos diversos, sempre de forma clara e didática, com o objetivo de servir aos alunos de pós-graduação em engenharia de estruturas. Em vista disso, esse material foi utilizado como base para a elaboração desse capítulo. Por mais que grande parte da teoria aqui descrita se aplique de forma genérica a diferentes elementos e problemas, o texto tem foco na resolução de problemas estáticos e dinâmicos, com a utilização de elementos sólidos bidimensionais triangulares. O modelo constitutivo adotado é o de Saint Venant Kirchhoff. Também é tratada, ao final desse capítulo, a teoria de imersão de elementos na malha.

3.1 Funções aproximadoras

O método dos elementos finitos é assim chamado por representar o contínuo através de um conjunto finito de subdomínios. Nesse caso, ao invés de um número infinito de incógnitas, essas são aproximadas em um conjunto finito de parâmetros nodais representativos.

As aproximações que ocorrem nos elementos para a representação do contínuo são obtidas por meio de polinômios aproximadores. Nesse processo são estabelecidos um intervalo válido e alguns pontos conhecidos. Posteriormente, calcula-se o polinômio que coincida com os valores dos pontos pré-estabelecidos. Esse processo é de simples dedução quando se trata de polinômios de grau 1 ou grau 2. Todavia, quando são utilizados polinômios de graus maiores, ou inclusive para calcular essa aproximação de forma genérica (pela facilidade de programação), são utilizadas outras técnicas, dentre elas, os polinômios de Lagrange, que foi a técnica utilizada neste trabalho.

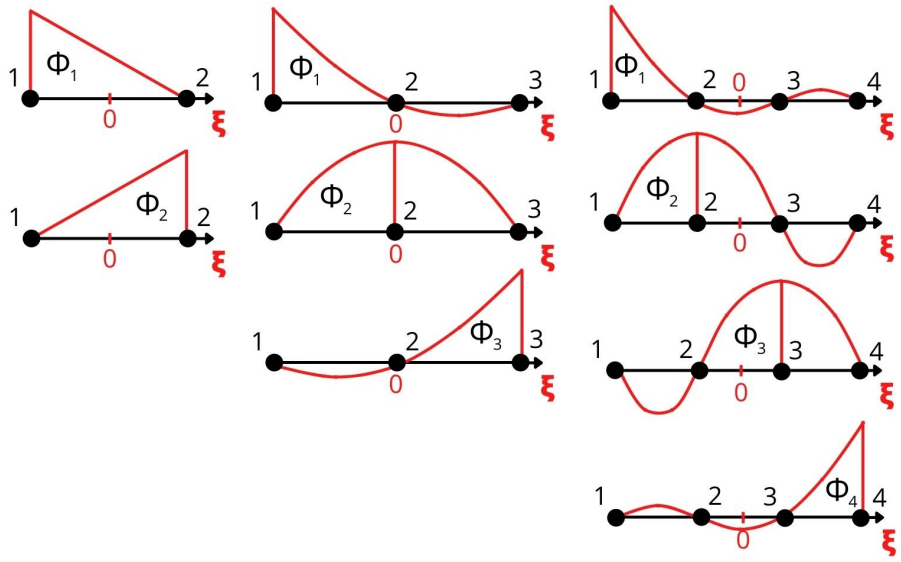
3.1.1 Polinômios de Lagrange

Para uma aproximação unidimensional, os polinômios de Lagrange, empregados como funções de forma, são definidos por meio de coordenadas adimensionais ξ, definidas no intervalo de $[-1, 1]$ (BOWER, 2010). O primeiro nó do elemento equivale à $\xi = -1$ e o último equivale à $\xi = 1$, sendo os nós intermediários distribuídos de maneira equidistante, conforme ilustra a Figura 3.1. A figura representa as funções de forma para elementos com aproximação linear, quadrática e cúbica. A função de forma Φ_α (correspondente ao nó α), vale 1 no nó α e 0 nos demais nós.

Tendo em vista a metodologia descrita, a obtenção das funções de forma se dá por

meio da equação (3.1), que representa a fórmula geral dos polinômios de Lagrange.

Figura 3.1 – Funções de forma linear, quadrática e cúbica



Fonte: Próprio autor.

$$\Phi_\alpha = \frac{\xi - \xi_1}{\xi_\alpha - \xi_1} \cdots \frac{\xi - \xi_{\alpha-1}}{\xi_\alpha - \xi_{\alpha-1}} \cdot \frac{\xi - \xi_{\alpha+1}}{\xi_\alpha - \xi_{\alpha+1}} \cdots \frac{\xi - \xi_n}{\xi_\alpha - \xi_n} \tag{3.1}$$

Desta forma, as funções de forma referentes ao elemento com aproximação linear obtidas por meio dos polinômios de Lagrange, representado na Figura 3.1, por exemplo, são dadas por:

$$\Phi_1 = \frac{1 - \xi}{2} \tag{3.2}$$

$$\Phi_2 = \frac{1 + \xi}{2} \tag{3.3}$$

A utilização dos polinômios de Lagrange como função aproximadora se dá devido à propriedade de partição da unidade. Ou seja, somando-se todos os valores de todos os polinômios, obtém-se 1 como resultado em qualquer ponto do domínio. Portanto, sendo conhecidos os valores da função em cada um dos nós do elemento (F_α), a função aproximadora pode ser escrita conforme o seguinte somatório:

$$f(\xi) = \Phi_\alpha(\xi) F_\alpha \tag{3.4}$$

No caso do MEF isoparamétrico, essa metodologia é utilizada inclusive para a aproximação da geometria dos elementos. Assim, sendo X_i e Y_i as posições, respectivamente, inicial e atual de um ponto do elemento, onde i equivale à direção da coordenada e α ao

nó do elemento, escreve-se, de forma análoga à equação (3.4):

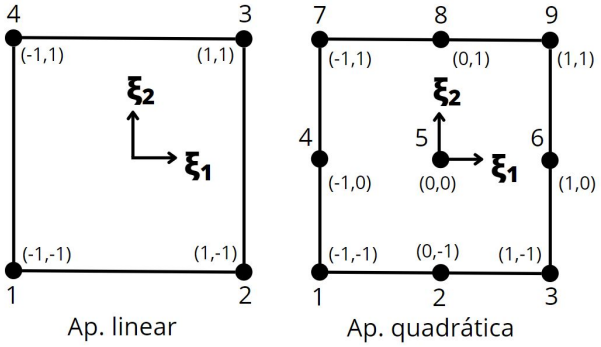
$$f_i^0(\xi) = \Phi_\alpha(\xi) X_i^\alpha \tag{3.5}$$

$$f_i^1(\xi) = \Phi_\alpha(\xi) Y_i^\alpha \tag{3.6}$$

As funções aproximadoras f_i^0 e f_i^1 representam, respectivamente, o mapeamento das configurações inicial e atual do elemento. Essa mesma metodologia pode ser aplicada para a obtenção de funções referentes a outras variáveis como, por exemplo, forças distribuídas aplicadas sobre os elementos.

Para aproximação bidimensional em elementos quadrangulares, utilizando os polinômios de Lagrange, segue-se a mesma lógica empregada na aproximação unidimensional. Ou seja, a função de forma do nó α deve valer 1 no ponto correspondente ao nó α e 0 nos demais nós do elemento. A representação de um elemento quadrangular com aproximações linear e quadrática no espaço adimensional é ilustrada por meio da Figura 3.2.

Figura 3.2 – Elementos quadrangulares de ordens 1 e 2 no espaço adimensional



Fonte: Próprio autor.

A obtenção das funções de forma desses elementos se dá por meio da seguinte equação:

$$\Phi_\alpha(\xi_1, \xi_2) = \Phi_\beta(\xi_1)\Phi_\gamma(\xi_2) \tag{3.7}$$

Em elementos tridimensionais pode-se obter as funções de forma seguindo a mesma metodologia, sendo elas descritas por meio da equação:

$$\Phi_\alpha(\xi_1, \xi_2, \xi_3) = \Phi_\beta(\xi_1)\Phi_\gamma(\xi_2)\Phi_\theta(\xi_3) \tag{3.8}$$

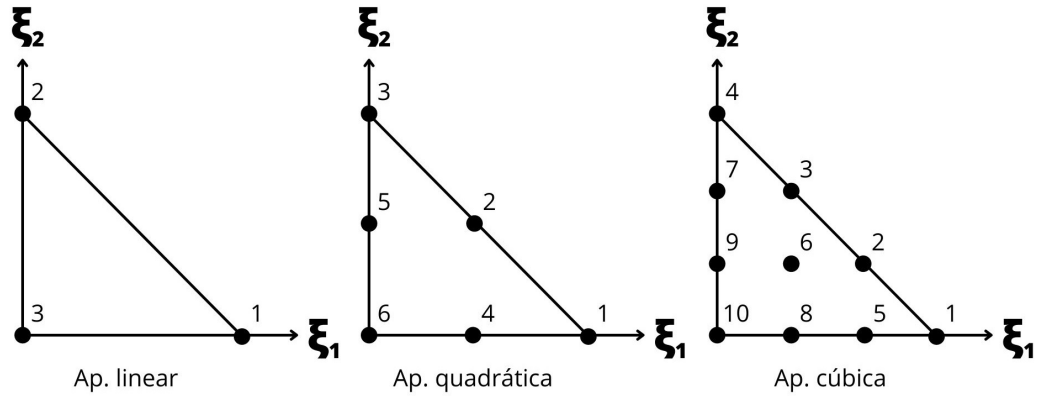
sendo β, γ e θ relacionados aos nós do elemento.

3.1.2 Funções aproximadoras em elementos com base triangular

Quando comparados aos elementos quadrangulares, os elementos triangulares possuem algumas vantagens e desvantagens. Dentre as vantagens pode-se destacar a

facilidade de acomodação à geometria das estruturas e a geração de polinômios completos. Nos elementos quadrangulares, por exemplo, são gerados alguns termos superabundantes, provenientes das funções de forma unidimensionais (FISH; BELYTSCHKO, 2007). A Figura 3.3 apresenta, no espaço adimensional, os elementos triangulares com aproximações linear, quadrática e cúbica. Nota-se que tanto ξ_1 quanto ξ_2 variam somente no intervalo de $0 \leq \xi_i \leq 1$.

Figura 3.3 – Elementos triangulares de ordens 1, 2 e 3 no espaço adimensional



Fonte: Próprio autor.

Para uma aproximação quadrática, ou seja, para um elemento triangular de 6 nós, são geradas as funções de forma conforme a regra de Pascal, por meio da equação (3.9). O índice α é relativo ao nó do elemento e, consequentemente, à sua função de forma. Assim, para a obtenção das funções de forma, é necessário calcular os 6 coeficientes referentes a cada um dos nós.

$$\Phi_\alpha(\xi_1, \xi_2) = a_{k1} + a_{k2}\xi_1 + a_{k3}\xi_2 + a_{k4}\xi_1^2 + a_{k5}\xi_1\xi_2 + a_{k6}\xi_2^2 \tag{3.9}$$

As coordenadas dos nós do elemento triangular com aproximação quadrática são dadas por $P_1(1,0)$, $P_2(\frac{1}{2},\frac{1}{2})$, $P_3(0,1)$, $P_4(\frac{1}{2},0)$, $P_5(0,\frac{1}{2})$ e $P_6(0,0)$. Como a função de forma correspondente ao nó α tem valor unitário no próprio nó, pode-se escrever 6 equações substituindo as coordenadas do nó α na função de forma α e igualando a 1. A equação (3.10) apresenta essas funções no formato matricial.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1/2 & 1/2 & 1/4 & 1/4 & 1/4 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1/2 & 0 & 1/4 & 0 & 0 \\ 1 & 0 & 1/2 & 0 & 0 & 1/4 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t = I \tag{3.10}$$

Dessa forma, como a multiplicação entre as matrizes resulta na matriz identidade, sendo M a matriz dos coeficientes, conclui-se que $M = P^{-1}$. Assim, através do cálculo dos coeficientes são obtidas as funções de forma do elemento.

3.1.3 Modelo Constitutivo de Saint-Venant-Kirchhoff

O gradiente dos mapeamentos das configurações inicial e atual (vide equações (3.5) e (3.6)) são necessários ao cálculo das deformações associadas ao método dos elementos finitos posicional. Estas são dadas pelas equações (3.11) e (3.12), a seguir:

$$A_{ij}^0 = f_{i,j}^0 = \Phi_{\alpha,j} X_i^\alpha \tag{3.11}$$

$$A_{ij}^1 = f_{i,j}^1 = \Phi_{\alpha,j} Y_i^\alpha \tag{3.12}$$

em que i e j representam a direção da coordenada, α é relativo ao nó do elemento, A_{ij}^0 é o gradiente do mapeamento da configuração inicial e A_{ij}^1 é o gradiente do mapeamento da configuração atual.

O cálculo do gradiente da função mudança de configuração (A_{ij}), utilizado para calcular a deformação de Green e, portanto, a energia de deformação, é dado por:

$$A_{ij} = A_{ij}^1 (A_{ij}^0)^{-1} \tag{3.13}$$

A medida de deformação utilizada neste trabalho é a de Green-Lagrange (E_{ij}), sendo essa uma medida adequada para grandes deslocamentos e rotações, e qualquer valor de deformação. A deformação de Green-Lagrange é calculada através do gradiente da função mudança de configuração (apresentado anteriormente através da equação (3.13)), conforme a equação abaixo:

$$E_{ij} = \frac{1}{2}(A_{ji} A_{ij} - \delta_{ij}) \tag{3.14}$$

Como conjugado energético da deformação de Green-Lagrange, tem-se o tensor de Piola-Kirchhoff de segunda espécie (S_{ij}), que pode ser escrito como sendo a variação da energia específica de deformação, em relação à deformação de Green-Lagrange:

$$S_{ij} = \frac{\partial u_e}{\partial E_{ij}} \tag{3.15}$$

A segunda variação da energia específica de deformação em função de E_{ij} resulta no tensor constitutivo elástico tangente (C_{ijkl}), conforme a equação abaixo:

$$C_{ijkl} = \frac{\partial^2 u_e}{\partial E_{ij} \partial E_{kl}} \tag{3.16}$$

Adotando o modelo constitutivo de Saint-Venant-Kirchhoff, que permite a consideração de problemas com grandes deslocamentos, porém, com deformações moderadas, a

energia específica de deformação e, consequentemente, a relação entre tensão e deformação são apresentadas, respectivamente, nas equações (3.17) e (3.18).

$$u_e^{SVK}(E) = \frac{1}{2} E_{ij} C_{ijkl} E_{kl} \tag{3.17}$$

$$S_{ij} = C_{ijkl} E_{kl} \tag{3.18}$$

Tratando-se especificamente de elementos sólidos bidimensionais, compostos de materiais isotrópicos, o tensor de Piola-Kirchhoff de segunda espécie, para o estado plano de deformações e de tensões é dado, respectivamente, pelas equações (3.19) e (3.20).

$$S = \begin{bmatrix} \frac{2G}{(1-2\nu)}((1-\nu)E_{11} + \nu E_{22}) & 2GE_{12} \\ 2GE_{21} & \frac{2G}{(1-2\nu)}((1-\nu)E_{22} + \nu E_{11}) \end{bmatrix} \tag{3.19}$$

$$S = \begin{bmatrix} \frac{2G}{(1-\nu)}(E_{11} + \nu E_{22}) & 2GE_{12} \\ 2GE_{21} & \frac{2G}{(1-\nu)}(E_{22} + \nu E_{11}) \end{bmatrix} \tag{3.20}$$

sendo G o módulo de elasticidade transversal do material, ν o coeficiente de poisson e E_{ij} o tensor de deformações de Green-Lagrange, em que $E_{12} = E_{21}$.

3.1.4 Integração numérica

No procedimento de resolução do MEF Posicional, deve-se realizar o cálculo de algumas integrais de domínio e superfície. Conforme descrito anteriormente nesse capítulo, a geometria inicial da estrutura é aproximada por meio dos elementos finitos, que são definidos por meio de um espaço adimensional (por meio de coordenadas ξ_i). Dessa forma, pode-se dizer que uma integral sobre o volume inicial de uma função qualquer $F(x)$, no espaço bidimensional é dada por:

$$\int_{V_0} F(\overrightarrow{x}) dV_0 = t \int_{\xi_1} \int_{\xi_2} F(\overrightarrow{x}(\xi_1, \xi_2)) J_0(\xi_1, \xi_2) d\xi_1 d\xi_2 \tag{3.21}$$

em que J_0 é o jacobiano do mapeamento inicial, dado pelo determinante do gradiente do mapeamento da configuração inicial ($det(A_0)$). A variável t representa a espessura, por se tratar de um elemento sólido bidimensional. Para casos de elementos tridimensionais, resolve-se uma integral tripla, adicionando-se a dimensão ξ_3.

A resolução dessas integrais apresenta certa complexidade, tornando mais adequado o uso de integração numérica. Nesse caso, para elementos unidimensionais, quadrangulares ou hexaédricos pode-se utilizar a quadratura de Gauss. Já para elementos triangulares ou tetraédricos pode ser empregada a quadratura de Hammer. Essas metodologias consistem em um somatório de n termos, sendo n o número de pontos no elemento. Para uma determinada configuração de pontos, são dadas as coordenadas adimensionais e o peso (w) referentes a cada ponto. Os termos são calculados por meio da multiplicação do valor da

função a ser integrada no ponto específico, multiplicado pelo jacobiano da integração e pelo peso correspondente, conforme apresentado abaixo:

$$\int_{V_0} F(\overrightarrow{x})dV_0 = \sum_{i=1}^{n} F(\overrightarrow{x}(\xi_i))J_0(\xi_i)w_i \tag{3.22}$$

Tal equação pode ser empregada nas duas quadraturas. No presente trabalho foram utilizadas as configurações de 7 e 12 pontos da quadratura de Hammer.

3.2 Princípio da estacionariedade

O MEF posicional é fundamentado no princípio da estacionariedade. Tal princípio estabelece a posição de equilíbrio de um sólido como sendo aquela onde a energia mecânica tem sua variação nula. Neste método, o equilíbrio é sempre calculado em relação à posição atual, ou seja, pode-se dizer que o método considera de forma natural a não linearidade geométrica. Conforme Greco e Coda (2006), a energia mecânica pode ser dividida em quatro termos: $\mathbb{P}$ - Energia potencial das forças externas; $\mathbb{U}$ - energia de deformação; $\mathbb{K}$ - energia cinética; $\mathbb{Q}$ - energia de dissipação. Desta forma, o cálculo da energia mecânica é dado por:

$$\Pi = \mathbb{P} + \mathbb{U} + \mathbb{K} + \mathbb{Q} \tag{3.23}$$

Seguindo o princípio da estacionariedade, calcula-se a variação da energia mecânica e iguala-se o resultado à 0, conforme a equação abaixo:

$$\delta\Pi = \delta\mathbb{P} + \delta\mathbb{U} + \delta\mathbb{K} + \delta\mathbb{Q} = 0 \tag{3.24}$$

3.2.1 Potencial das forças externas

Neste trabalho serão consideradas apenas forças conservativas, ou seja, que permanecem com intensidade, direção e sentido constantes, que não se sujeitam a mudanças com a variação de forma da estrutura. O potencial das forças externas pode conter parcelas decorrentes de forças concentradas, distribuídas em partes da superfície do sólido, ou ainda em partes do seu domínio. De forma geral, o potencial pode ser escrito por meio da seguinte equação:

$$\mathbb{P} = -F_i^{\alpha}Y_i^{\alpha} - \int_{\Gamma_0} q_i y_i d\Gamma_0 - \int_{\Omega_0} b_i y_i d\Omega_0 \tag{3.25}$$

em que o índice i é relativo a direção da coordenada, α ao nó do elemento, a variável Y se refere à posição atual da força concentrada, e y é utilizado para representar a posição atual das forças distribuídas, com a intenção de diferenciar os pontos isolados de pontos da superfície ou da linha. Já q e b representam, respectivamente, força de superfície e força de volume e, por fim, Γ e Ω representam superfície e domínio, respectivamente, sendo o índice 0 utilizado para caracterizá-los com base em sua posição inicial, já que se tratam de forças conservativas.

3.2.2 Energia de deformação

A descrição da energia de deformação em materiais hiperelásticos é dada pela integral da energia específica de deformação em todo o volume do sólido (sendo esse o volume inicial, considerando problemas Lagrangianos). Tomando como base o modelo constitutivo de Saint-Venant-Kirchhoff, por meio das equações (3.17) e (3.18), a energia de deformação pode ser representada pela equação abaixo:

$$\mathbb{U} = \int_{\Omega_0} u_e d\Omega_0 = \int_{\Omega_0} \frac{1}{2} E_{mn} S_{mn} d\Omega_0 \tag{3.26}$$

A variação da energia de deformação é descrita por:

$$\delta\mathbb{U} = \int_{\Omega_0} S_{mn} \delta E_{mn} d\Omega_0 \tag{3.27}$$

em que os índices m e n representam as direções da deformação e da tensão, S o tensor de Piola-Kirchhoff de segunda espécie e E a deformação de Green-Lagrange.

3.2.3 Energia cinética

A parcela referente à energia cinética pode ser descrita por meio da Equação 3.28. Esse termo é dependente da massa do elemento, assim como da variação da sua posição com relação ao tempo (velocidade), sendo que os termos ρ_0 e $\dot{y}_i$ representam, respectivamente, a densidade do material e a variação da posição em função do tempo.

$$\mathbb{K} = \frac{1}{2} \int_{\Omega_0} \rho_0 \dot{y}_i \dot{y}_i d\Omega_0 \tag{3.28}$$

Desenvolvendo a sua variação, tomando o tempo como parâmetro, obtém-se a equação descrita abaixo, sendo $\ddot{y}_i$ a segunda variação da posição em função do tempo (aceleração):

$$\delta\mathbb{K} = \frac{d\mathbb{K}}{dt}\delta t = \int_{\Omega_0} \rho_0 \ddot{y}_i \delta y_i d\Omega_0 \tag{3.29}$$

3.2.4 Energia de dissipação

Em um sistema no qual há a conservação de energia, a entrada e a saída de energia devem ser iguais. Caso ocorra a dissipação de parte dessa energia, há uma alteração na energia total do sistema ao longo do tempo. Por esse motivo, o termo de energia de dissipação é acrescido na descrição da energia mecânica total (Equação 3.23). Sua variação em função da posição, considerando apenas a dissipação de energia de amortecimento, proporcional à sua massa, pode ser descrita da seguinte forma:

$$\frac{\partial \mathbb{Q}}{\partial Y_i^\alpha} = \int_{\Omega_0} \frac{\partial q}{\partial Y_i^\alpha} d\Omega_0 = \int_{\Omega_0} \rho_0 c_m \dot{y}_i d\Omega_0 \tag{3.30}$$

sendo q o funcional de energia específica de dissipação e c_m a constante de amortecimento.

3.3 Método de Newton-Raphson

A descrição da energia em função das posições atuais dos nós da estrutura é um dos pontos cruciais para a resolução do problema. Dessa forma, recorrendo ao exposto na equação (3.4), escrevem-se as derivadas (em função da posição atual dos nós) do potencial das forças externas, da energia de deformação, da energia cinética, e da energia de dissipação, respectivamente, por meio das equações (3.31), (3.32), (3.33) e (3.34).

$$\frac{\partial \mathbb{P}}{\partial Y_i^\alpha} = -F_i^\alpha - \int_{\Gamma_0} \Phi_\alpha \Phi_\beta d\Gamma_0 Q_i^\beta - \int_{\Omega_0} \Phi_\alpha \Phi_\beta d\Omega_0 B_i^\beta \tag{3.31}$$

$$\frac{\partial \mathbb{U}}{\partial Y_i^\alpha} = \int_{\Omega_0} S_{mn} \frac{\partial E_{mn}}{\partial Y_i^\alpha} d\Omega_0 \tag{3.32}$$

$$\frac{\partial \mathbb{K}}{\partial Y_i^\alpha} = \int_{\Omega_0} \rho_0 \Phi_\alpha \Phi_\beta \ddot{Y}_i^\alpha d\Omega_0 \tag{3.33}$$

$$\frac{\partial \mathbb{Q}}{\partial Y_i^\alpha} = \int_{\Omega_0} \rho_0 c_m \Phi_\alpha \Phi_\beta \dot{Y}_i^\alpha d\Omega_0 \tag{3.34}$$

sendo i relacionado a direção da coordenada, Φ às funções de forma, α e β aos nós do elemento e Q e B as forças de superfície e de volume, respectivamente.

Tratando de problemas estáticos, considera-se $\mathbb{K} = 0$ e $\mathbb{Q} = 0$. Portanto, tem-se que a variação da energia mecânica em função da posição atual (Y_i^α) é dada pela equação (3.35) que, segundo o princípio da energia mecânica estacionária, é nula. Desta forma, pode-se dizer que ($\partial \mathbb{U}/\partial Y_i^\alpha$) é o vetor de forças internas, que somado ao vetor de forças externas ($\partial \mathbb{P}/\partial Y_i^\alpha$) resulta em zero.

$$\frac{\partial \Pi}{\partial Y_i^\alpha} \delta Y_i^\alpha = \left(\frac{\partial \mathbb{P}}{\partial Y_i^\alpha} + \frac{\partial \mathbb{U}}{\partial Y_i^\alpha} \right) \delta Y_i^\alpha = 0 \tag{3.35a}$$

$$-F_i^\alpha - \int_{\Gamma_0} \Phi_\alpha \Phi_\beta d\Gamma_0 Q_i^\beta - \int_{\Omega_0} \Phi_\alpha \Phi_\beta d\Omega_0 B_i^\beta + \int_{\Omega_0} S_{mn} \frac{\partial E_{mn}}{\partial Y_i^\alpha} d\Omega_0 = 0 \tag{3.35b}$$

O método de solução do problema não linear utilizado foi a estratégia incremental-iterativa tangente, também conhecido como método de Newton-Raphson. O procedimento se inicia adotando uma posição tentativa (Y^0) para calcular a variação da energia de deformação. Enquanto a posição tentativa for diferente da solução do problema, a equação (3.35) não é satisfeita, gerando um vetor resíduo ($g(Y^0)$), também chamado de vetor de desbalanceamento mecânico. Posteriormente é realizada uma expansão em série de Taylor em torno dessa posição tentativa, representada por meio da equação (3.36), sendo $g_i^\alpha = \partial \Pi / \partial Y_i^\alpha$.

$$g_i^\alpha = g_i^\alpha(Y^k) + \left. \frac{\partial g_i^\alpha}{\partial Y_j^\beta} \right|_{(Y^k)} \left(\Delta Y_j^\beta \right)^k + (O_i^\alpha)^2 = 0 \tag{3.36}$$

A variável $(O_j^\alpha)^2$ representa os termos de ordens superiores. Desprezando-se esses

termos, pode-se reescrever a equação (3.36) da seguinte forma:

$$\left(\Delta Y_j^\beta\right)^k = -\left(\left.\frac{\partial g_i^\alpha}{\partial Y_j^\beta}\right|_{(Y^k)}\right)^{-1} g_i^\alpha(Y^k) \tag{3.37}$$

Como a segunda variação do potencial de forças externas não é função da posição atual dos nós, escreve-se:

$$\left.\frac{\partial g_i^\alpha}{\partial Y_j^\beta}\right|_{(Y^k)} = \left.\frac{\partial^2 \mathbb{U}}{\partial Y_i^\alpha Y_j^\beta}\right|_{(Y^k)} = H_{ij\alpha\beta}^k \tag{3.38}$$

em que $H_{ij\alpha\beta}$ é a matriz de rigidez tangente, ou matriz Hessiana para essa posição tentativa. Com base na equação (3.32), escreve-se:

$$H_{ij\alpha\beta}^k = \int_{\Omega_0} \frac{\partial}{\partial Y_j^\beta}\left(\left. S_{mn}\frac{\partial E_{mn}}{\partial Y_i^\alpha}\right)\right|_{Y^k} d\Omega_0 = 0 \tag{3.39}$$

Assim, reescreve-se abaixo a equação (3.37), apresentando de forma resumida o cálculo do vetor $\left(\Delta Y_j^\beta\right)^k$, sendo esse o ponto chave da estratégia incremental-iterativa tangente, ou método de Newton-Raphson.

$$\left(\Delta Y_j^\beta\right)^k = -\left(H_{ij\alpha\beta}^k\right)^{-1} g_i^\alpha(Y^k) \tag{3.40}$$

Em seguida, atualizam-se as posições:

$$\left(Y_j^\beta\right)^{k+1} = \left(Y_j^\beta\right)^k + \left(\Delta Y_j^\beta\right)^k \tag{3.41}$$

Desta forma, o processo de solução tem início adotando-se para a posição tentativa a própria posição inicial. A partir daí é realizado um incremento de carga (ou de posição prescrita) e, por meio da equação (3.40), calcula-se o vetor $\left(\Delta Y_j^\beta\right)^k$. Atualiza-se a posição, somando esse vetor ao vetor de posição tentativa. Caso o valor de $\left|\left(\Delta Y_j^\beta\right)^k / X_j^\beta\right|$ (cálculo do erro, sendo X_j^β o vetor de posições iniciais) for maior que a tolerância adotada, repete-se o mesmo procedimento para a nova posição tentativa. Caso o valor seja menor que essa tolerância, é realizado um novo incremento de carga e, depois disso, repete-se o processo, até que se chegue à carga total (ou posição prescrita total), com um valor de erro menor que a tolerância adotada.

Com a finalidade de facilitar o entendimento da resolução do problema estático por meio do MEF Posicional, foi elaborado um algoritmo de solução simplificado, que pode ser visualizado abaixo:

Algoritmo 1: Resolução de problema estático

Atribuição das coordenadas iniciais como atuais para definição da primeira posição tentativa ($Y_i^\alpha = X_i^\alpha$);
para cada *passo de carga* (dF_i^α) *ou posição* (dY_i^α) **faça**
    Atualização do vetor de forças externas ($F_i^\alpha = F_i^\alpha + dF_i^\alpha$) ou de posição ($Y_i^\alpha = Y_i^\alpha + dY_i^\alpha$);
    enquanto *resíduo > tolerância* **faça**
        para cada *elemento* **faça**
            para cada *ponto de integração* **faça**
                Cálculo das funções de forma e suas derivadas;
                Cálculo do gradiente do mapeamento da configuração inicial (A_{ij}^0 - Equação 3.11) e final (A_{ij}^1 - Equação 3.12);
                Cálculo da deformação de Green-Lagrange (E_{ij} - Equação 3.14);
                Cálculo do tensor de Piola-Kirchhoff de segunda espécie (S_{ij} - Equação 3.15);
                para cada *nó* α *na direção* i **faça**
                    Cálculo do vetor global de força interna ($\frac{\partial \mathbb{U}}{\partial Y_i^\alpha}$ - Equação 3.32);
                    para cada *nó* β *na direção* j **faça**
                        Cálculo da hessiana global ($H_{ij\alpha\beta}^k$ - Equação 3.38);
                    fim
                fim
            fim
            Imposição das condições de contorno;
            Contribuição do elemento na força interna e hessiana globais;
        fim
        Resolução do sistema linear (Obtenção do $(\Delta Y_i^\alpha)^k$ - Equação 3.40);
        Atualização da posição ($(Y_i^\alpha)^{k+1}$ - Equação 3.41);
        Avaliação do resíduo $\left(\left|\left(\Delta Y_j^\beta\right)^k / X_j^\beta\right|\right)$;
    fim
fim

3.4 Problemas dinâmicos (Newmark + Newton-Raphson)

Para o cálculo de problemas dinâmicos, passa-se a considerar $\mathbb{K} \neq 0$ e $\mathbb{Q} \neq 0$. Portanto, a variação da energia mecânica em função da posição atual (Y_i^α), descrita anteriormente por meio da equação (3.35), pode ser obtida por meio da equação (3.42).

$$\frac{\partial \Pi}{\partial Y_i^\alpha} = \frac{\partial \mathbb{P}}{\partial Y_i^\alpha} + \frac{\partial \mathbb{U}}{\partial Y_i^\alpha} + \frac{\partial \mathbb{K}}{\partial Y_i^\alpha} + \frac{\partial \mathbb{Q}}{\partial Y_i^\alpha} = 0 \tag{3.42a}$$

$$\frac{\partial \Pi}{\partial Y_i^\alpha} = -F_i^\alpha - \int_{\Gamma_0} \Phi_\alpha \Phi_\beta d\Gamma_0 Q_i^\beta - \int_{\Omega_0} \Phi_\alpha \Phi_\beta d\Omega_0 B_i^\beta + \int_{\Omega_0} S_{mn} \frac{\partial E_{mn}}{\partial Y_i^\alpha} d\Omega_0 + \int_{\Omega_0} \rho_0 c_m \Phi_\alpha \Phi_\beta \dot{Y_i^\beta} d\Omega_0 + \int_{\Omega_0} \rho_0 \Phi_\alpha \Phi_\beta \ddot{Y_i^\beta} d\Omega_0 = 0 \tag{3.42b}$$

Uma vez que a matriz de massa de um elemento finito é descrita por:

$$M = \int_{\Omega_0} \rho \Phi_\alpha \Phi_\beta d\Omega_0 \tag{3.43}$$

e ainda, considerando a matriz de amortecimento $C = Mc_m$, pode-se reescrever a equação (3.42) da seguinte forma:

$$-F_i^\alpha - \int_{\Gamma_0} \Phi_\alpha \Phi_\beta d\Gamma_0 Q_i^\beta - \int_{\Omega_0} \Phi_\alpha \Phi_\beta d\Omega_0 B_i^\beta + \int_{\Omega_0} S_{mn} \frac{\partial E_{mn}}{\partial Y_i^\alpha} d\Omega_0 + C_{\alpha\beta} \dot{Y_i^\beta} + M_{\alpha\beta} \ddot{Y_i^\beta} = 0 \tag{3.44}$$

Para iniciar o procedimento de cálculo, deve-se admitir um valor inicial para a aceleração. Partindo da Equação 3.44, escreve-se:

$$\ddot{Y_{(t_0)}}_{\alpha\beta} = M^{-1}_{\alpha\beta}\left[F_i^\alpha + \int_{\Gamma_0}\Phi_\alpha\Phi_\beta d\Gamma_0 Q_i^\beta + \int_{\Omega_0}\Phi_\alpha\Phi_\beta d\Omega_0 B_i^\beta - \int_{\Omega_0} S_{mn}\frac{\partial E_{mn}}{\partial Y_i^\alpha}d\Omega_0 - C_{\alpha\beta}\dot{Y_{(t_0)}}\right] \tag{3.45}$$

O tempo é uma variável contínua, entretanto, para a resolução do problema numérico, ele é tomado como uma variável discreta. Desta forma, são utilizados integradores temporais para calcular, de forma aproximada, passos de tempo, nos quais são realizadas a análise. Neste trabalho foram utilizados os integradores temporais de Newmark. Cada passo é escrito em função do passo anterior, ou seja, sendo $t_{(s)}$, $t_{(s+1)}$ e Δt, respectivamente, o passo de tempo atual, o passo de tempo anterior, e um intervalo de tempo, tem-se que:

$$t_{(s+1)} = t_{(s)} + \Delta t \tag{3.46}$$

Assim sendo, considerando $Y_{t_{(s+1)}}$, $\dot{Y}_{t_{(s+1)}}$ e $\ddot{Y}_{t_{(s+1)}}$, respectivamente, a posição, velocidade e aceleração atuais, aproximam-se os valores atuais de posição e velocidade, por meio das expressões de Newmark:

$$Y_{t_{(s+1)}} = Y_{t_{(s)}} + \Delta t\dot{Y}_{t_{(s)}} + \Delta t^2\left[\left(\frac{1}{2}-\beta\right)\ddot{Y}_{t_{(s)}} + \beta\ddot{Y}_{t_{(s+1)}}\right] \tag{3.47}$$

$$\dot{Y}_{t_{(s+1)}} = \dot{Y}_{t_{(s)}} + \Delta t(1-\gamma)\ddot{Y}_{t_{(s)}} + \gamma\Delta t\ddot{Y}_{t_{(s+1)}} \tag{3.48}$$

sendo β e γ parâmetros livres do método. Para considerar a aceleração constante em um passo de tempo, por exemplo, são utilizados os parâmetros $\beta = 1/4$ e $\gamma = 1/2$.

Isolando-se a velocidade e a aceleração atuais nas equações (3.47) e (3.48), descrevem-se abaixo essas variáveis em função das posições atuais e valores referentes ao passo de tempo anterior (já conhecidos).

$$\ddot{Y}_{t_{(s+1)}} = \frac{Y_{t_{(s+1)}}}{\beta\Delta t^2} - \left(\frac{Y_{t_{(s)}}}{\beta\Delta t^2} + \frac{\dot{Y}_{t_{(s)}}}{\beta\Delta t^2} + \left(\frac{1}{2\beta}-1\right)\ddot{Y}_{t_{(s)}}\right) \tag{3.49}$$

$$\dot{Y}_{t_{(s+1)}} = \frac{\gamma}{\beta\Delta t}Y_{t_{(s+1)}} + \left[\dot{Y}_{t_{(s)}} + \Delta t(1-\gamma)\ddot{Y}_{t_{(s)}}\right] - \left(\frac{Y_{t_{(s)}}}{\beta\Delta t^2} + \frac{\dot{Y}_{t_{(s)}}}{\beta\Delta t^2} + \left(\frac{1}{2\beta}-1\right)\ddot{Y}_{t_{(s)}}\right)\gamma\Delta t \tag{3.50}$$

De modo a facilitar os cálculos, separam-se as parcelas relacionadas ao passo de tempo anterior, nomeadas como Q_s e R_s, conforme as equações abaixo:

$$Q_s = \frac{Y_{t_{(s)}}}{\beta\Delta t^2} + \frac{\dot{Y}_{t_{(s)}}}{\beta\Delta t} + \left(\frac{1}{2\beta}-1\right)\ddot{Y}_{t_{(s)}} \tag{3.51}$$

$$R_s = \dot{Y}_{t_{(s)}} + \Delta t(1-\gamma)\ddot{Y}_{t_{(s)}} \tag{3.52}$$

substituindo os valores de Q_s e R_s nas equações (3.49) e (3.50), são obtidas as seguintes equações:

$$\ddot{Y}_{t_{(s+1)}} = \frac{Y_{t_{(s+1)}}}{\beta \Delta t^2} - Q_s \tag{3.53}$$

$$\dot{Y}_{t_{(s+1)}} = \frac{\gamma}{\beta \Delta t} Y_{t_{(s+1)}} + R_s - Q_s \gamma \Delta t \tag{3.54}$$

De posse desses valores, pode-se reescrever a variação da energia mecânica em função da posição no instante atual ($Y|_{t_{(s+1)}}$), substituindo os valores das equações (3.53) e (3.54) na equação (3.42), conforme demonstrado abaixo (índices suprimidos para facilitar a leitura):

$$\left.\frac{\partial \Pi}{\partial Y}\right|_{t_{(s+1)}} = \left.\frac{\partial \mathbb{P}}{\partial Y}\right|_{t_{(s+1)}} + \left.\frac{\partial \mathbb{U}}{\partial Y}\right|_{t_{(s+1)}} + C\left(\frac{\gamma}{\beta \Delta t} Y_{t_{(s+1)}} + R_s - Q_s \gamma \Delta t\right)$$
$$+M\left(\frac{Y_{t_{(s+1)}}}{\beta \Delta t^2} - Q_s\right) = 0 \tag{3.55}$$

Para a montagem da resolução de problemas dinâmicos, realiza-se o mesmo procedimento utilizado para problemas estáticos (equações (3.36) a (3.40)), porém, considerando as parcelas referentes à energia cinética e de dissipação. Desta forma, é admitida uma posição tentativa, adotando-se, para o início dos cálculos, $Y^k = Y_{t_{(s+1)}} = Y_{t_{(s)}}$. Em seguida, é realizada uma expansão em série de Taylor truncada em primeira ordem, ou seja, desprezando os termos de ordens superiores, o que pode ser visto na equação abaixo:

$$g_i^\alpha\left(Y_{t_{(s+1)}}\right) = g_i^\alpha\left(Y^k_{t_{(s+1)}}\right) + \left.\frac{\partial g_i^\alpha}{\partial Y_j^\beta}\right|_{\left(Y^k_{t_{(s+1)}}\right)} \left(\Delta Y_j^\beta\right)^k = 0 \tag{3.56}$$

$$\left(\Delta Y_j^\beta\right)^k = -\left(\left.\frac{\partial g_i^\alpha}{\partial Y_j^\beta}\right|_{\left(Y^k_{t_{(s+1)}}\right)}\right)^{-1} g_i^\alpha\left(Y^k_{t_{(s+1)}}\right) \tag{3.57}$$

Desta forma, a segunda variação da energia mecânica em função da posição, agora considerando as parcelas relativas à energia cinética e energia de dissipação, é escrita da seguinte forma:

$$\left.\frac{\partial g_i^\alpha}{\partial Y_j^\beta}\right|_{\left(Y^k_{t_{(s+1)}}\right)} = \left.\frac{\partial^2 \mathbb{U}}{\partial Y_i^\alpha Y_j^\beta}\right|_{(Y^k_{t_{(s+1)}})} + \frac{M_{\alpha\beta}}{\beta \Delta t^2} + \frac{C_{\alpha\beta}\gamma}{\beta \Delta t} = H^{(din)}_{ij\alpha\beta} \tag{3.58}$$

sendo $H^{(din)}_{ij\alpha\beta}$ a matriz Hessiana para a posição tentativa, no instante atual ($t_{(s+1)}$). Da mesma forma que foi feito para problemas estáticos, reescreve-se o cálculo do vetor de variação da posição $\left(\Delta Y_j^\beta\right)^k$, que é de suma importância no processo de resolução.

$$\left(\Delta Y_j^\beta\right)^k = -\left(H^{k(din)}_{ij\alpha\beta}\right)^{-1} g_i^\alpha\left(Y^k_{t_{(s+1)}}\right) \tag{3.59}$$

O processo de solução para problemas dinâmicos é muito similar ao utilizado para problemas estáticos. Basicamente, resolve-se um problema estático, seguindo-se o mesmo procedimento, para cada passo de tempo diferente. Por meio do algoritmo 2 são descritos os passos de solução, vinculando cada um deles com as equações expostas neste capítulo.

Algoritmo 2: Resolução de problema dinâmico

```
1  Atribuição da posição, velocidade e aceleração do passo anterior como primeira tentativa:
     (Y_{t(s+1)} = Y^k_{t(s+1)} = Y_{t(s)}) / (Ẏ_{t(s+1)} = Ẏ^k_{t(s+1)} = Ẏ_{t(s)}) / (Ÿ_{t(s+1)} = Ÿ^k_{t(s+1)} = Ÿ_{t(s)});
2  Cálculo da matriz de massa (M) e de amortecimento (C) (Equação 3.43);
3  Cálculo da aceleração inicial (Equação 3.45)
4  para cada passo de carga (dF_i^α), posição (dY_i^α) ou tempo (Δt) faça
5      Atualização do vetor de forças externas (F_i^α = F_i^α + dF_i^α) ou de posição (Y_i^α = Y_i^α + dY_i^α);
6      Cálculo das parcelas relativas ao passo de tempo anterior - Q_s (Equação 3.51) e R_s (Equação 3.52);
7      enquanto resíduo > tolerância faça
8          para cada elemento faça
9              para cada ponto de integração faça
10                 Cálculo das funções de forma e suas derivadas;
11                 Cálculo do gradiente do mapeamento da configuração inicial (A^0_ij - Equação 3.11) e final
                     (A^1_ij - Equação 3.12);
12                 Cálculo da deformação de Green-Lagrange (E_ij - Equação 3.14);
13                 Cálculo do tensor de Piola-Kirchhoff de segunda espécie (S_ij - Equação 3.15);
14                 para cada nó α na direção i faça
15                     Cálculo da contribuição no vetor global de força interna (∂𝕌/∂Y_i^α + ∂𝕂/∂Y_i^α + ∂ℚ/∂Y_i^α -
                         Soma das equações (3.32), (3.33) e (3.34) );
16                     para cada nó β na direção j faça
17                         Cálculo da contribuição na hessiana global dinâmica (H^{k(din)}_{ijαβ} - Equação 3.58);
18                     fim
19                 fim
20             fim
21             Imposição das condições de contorno;
22             Contribuição do elemento na força interna e hessiana globais;
23         fim
24         Resolução do sistema linear (Obtenção do (ΔY_j^β)^k - Equação 3.59);
25         Atualização da posição ((Y_j^β)^{k+1} = (Y_j^β)^k + (ΔY_j^β)^k);
26         Atualização da aceleração e da velocidade atuais (Equações (3.53) e (3.54));
27         Avaliação do resíduo (|(ΔY_j^β)^k / X_j^β|);
28     fim
29     Atualização do passo de tempo (t = t + Δt);
30 fim
```

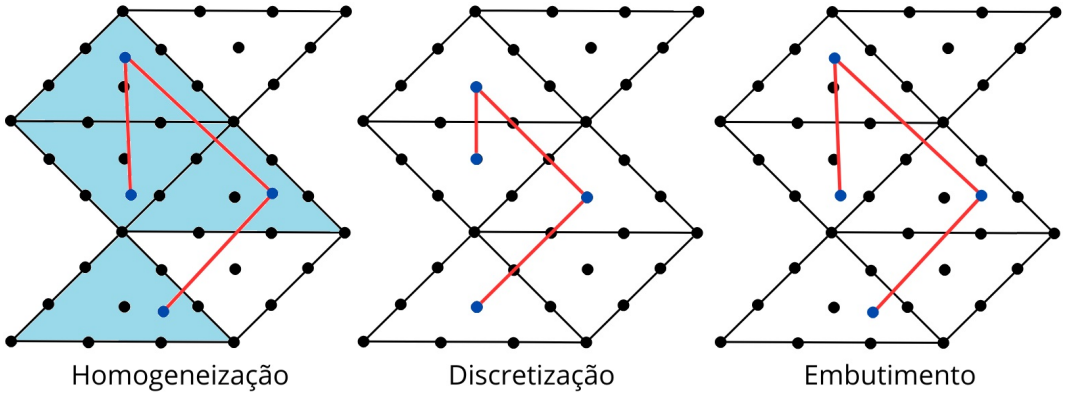
3.5 Elementos imersos

As técnicas de inserção de elementos imersos na malha são utilizadas para a simulação de materiais heterogêneos, compostos por uma matriz reforçada com fibras, ou partículas. Existem diferentes abordagens para a modelagem desse tipo de material, que influenciam de diferentes formas no modelo de geração de malha, nos graus de liberdade, e na formulação utilizada. Conforme descrito por Ramos (2020), pode-se dividir essas técnicas em três diferentes formas de modelagem: Homogeneização, Discretização e Embutimento.

A Figura 3.4 apresenta exemplos de cada uma dessas técnicas.

A primeira delas, baseia-se na homogeneização das propriedades dos elementos nos quais estão contidos os nós dos reforços. Já na estratégia de discretização, os nós dos elementos de reforço devem coincidir com os nós da malha. Ou seja, os elementos de reforço atuam diretamente sobre os graus de liberdade do problema. Nesse caso, para compatibilizar a malha, tem-se duas opções: reconstruir a malha, passando obrigatoriamente sobre os pontos dos elementos de reforço, ou adequar os elementos de reforço para que coincidam com nós já existentes da malha. Por último, a técnica de embutimento, que foi implementada neste trabalho, em que é feita a contribuição dos nós do elemento de reforço, em cada um dos nós do elemento da matriz no qual está contido, reescrevendo os parâmetros nodais do elemento de reforço em função dos parâmetros nodais do elemento da matriz. Essa é uma estratégia muito interessante, uma vez que não é necessária nenhuma alteração na malha, levando a uma grande liberdade no posicionamento dos reforços e não aumenta os graus de liberdade do problema.

Figura 3.4 – Técnicas para inserção de elementos na malha



Fonte: Próprio autor.

A técnica de embutimento aqui tratada pode ser aplicada para a inserção de fibras e/ou partículas na matriz. Neste trabalho foram implementados apenas elementos finitos de barra simples, ou seja, com dois graus de liberdade por nó.

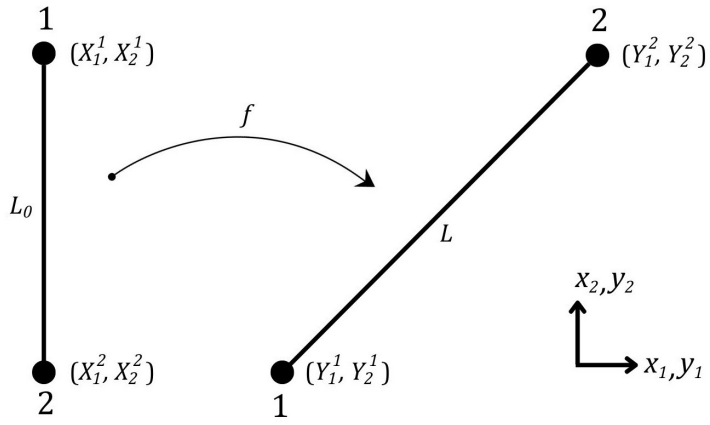
3.5.1 Elemento finito de barra simples

Nesta seção são demonstrados os cálculos referentes à força interna e à hessiana desse tipo de elemento, se tratando, respectivamente, da primeira e da segunda variação da energia de deformação com relação à posição.

Conforme tratado anteriormente, o cálculo da energia de deformação é deduzido por meio das configurações inicial e atual do elemento. Na Figura 3.5 observam-se essas configurações, nas quais estão devidamente indicados as posições dos nós (X_i^α e Y_i^α, sendo

i referente à direção do grau de liberdade e α referente ao nó), assim como os comprimentos inicial (L_0) e atual (L) do elemento.

Figura 3.5 – Configurações inicial e final de um elemento de barra simples



Fonte: Próprio autor.

Desta forma, como o elemento está submetido apenas a esforços normais, a deformação de Green é obtida por meio da seguinte equação:

$$E = \frac{1}{2}\left(\frac{L^2 - L_0^2}{L_0^2}\right) \tag{3.60}$$

sendo L_0^2 e L^2 dados por:

$$L_0^2 = \left(X_1^2 - X_1^1\right)^2 + \left(X_2^2 - X_2^1\right)^2 \tag{3.61}$$

$$L^2 = \left(Y_1^2 - Y_1^1\right)^2 + \left(Y_2^2 - Y_2^1\right)^2 \tag{3.62}$$

Considerando-se a resolução de problemas com cinemática Lagrangiana, sendo Ω_0 o volume inicial, Γ_0 a área inicial e L_0 o comprimento inicial, pode-se escrever a energia de deformação na configuração inicial, para um elemento, da seguinte forma:

$$U^{el} = \int_{\Omega_0} u_e d\Omega_0 = u_e \Omega_0 = u_e \Gamma_0 L_0 \tag{3.63}$$

Seguindo com a utilização da lei constitutiva de Saint-Venant-Kirchhoff, obtém-se a equação da força interna, que pode ser vista abaixo:

$$\frac{\partial U^{el}}{\partial Y_i^\alpha} = \Gamma_0 L_0 \frac{\partial u_e}{\partial E}\frac{\partial E}{\partial Y_i^\alpha} \tag{3.64}$$

aplicando o conceito de conjugado energético, em que aparece a tensão de Piola Kirchhoff de segunda espécie ($S = \partial u_e / \partial E$), e sendo:

$$\frac{\partial E}{\partial Y_i^\alpha} = \frac{(-1)^\alpha}{(L_0)^2}(Y_i^2 - Y_i^1) \tag{3.65}$$

tem-se que:

$$\frac{\partial U^{el}}{\partial Y_i^\alpha} = \Gamma_0 S \frac{(-1)^\alpha}{L_0}(Y_i^2 - Y_i^1) \tag{3.66}$$

Mantendo as mesmas considerações, porém, calculando-se a segunda derivada da energia de deformação em função da posição atual, obtém-se a hessiana, ou matriz tangente, que pode ser escrita conforme a equação abaixo:

$$\begin{aligned}\frac{\partial^2 U^{el}}{\partial Y_i^\alpha Y_j^\beta} &= \frac{\partial}{\partial Y_i^\alpha}\left(\frac{\partial U^{el}}{\partial Y_j^\beta}\right) = \Gamma_0 L_0 \frac{\partial}{\partial Y_i^\alpha}\left(\frac{\partial u_e}{\partial E}\frac{\partial E}{\partial Y_j^\beta}\right) \\ &= \frac{(-1)^\alpha(-1)^\beta\Gamma_0}{L_0}\left(K_t^{SVK}\frac{(Y_i^2 - Y_i^1)}{L_0}\frac{(Y_j^2 - Y_j^1)}{L_0} + S\delta_{ij}\right)\end{aligned} \tag{3.67}$$

3.5.2 Estratégia de acoplamento entre matriz e reforço

Neste trabalho foi utilizada a técnica de embutimento, aplicada ao método dos elementos finitos posicional, com uma formulação proposta inicialmente por Vanalli (2004), e empregada em uma série de trabalhos no decorrer dos últimos anos. Podem ser aqui citados os trabalhos de Fernandes (2016) e Moura (2015), que propõem análises elásticas e elastoplásticas de materiais compósitos, de Felix (2018) e Ramos (2020), que trataram da simulação do processo de corrosão em estruturas de concreto armado, e de Tavares (2020), que utilizou a técnica para modelar uma estrutura de concreto protendido. Também os trabalhos de Salomão (2021), envolvendo análises termomecânicas em compósitos, de Felix (2022) propondo um modelo de dano acumulado em concreto, e de Marques (2023), que também trabalhou com um modelo de dano, porém, para analisar o comportamento de estruturas de concreto armado. Todos esses trabalhos utilizaram o método dos elementos finitos posicional, em conjunto com a estratégia de embutimento, comprovando o seu potencial em diversas aplicações.

A estratégia consiste em tratar os parâmetros nodais do reforço, em função dos parâmetros nodais da matriz, fazendo o uso das funções de forma desses elementos. Cada um dos nós dos elementos de reforço será atribuído a um elemento da matriz, para que essa mudança de referência seja realizada. Sendo ($\hat{\bullet}$) a simbologia utilizada para descrever os termos relacionados à matriz, e ($\overline{\bullet}$) ao reforço, pode-se escrever os mapeamentos das posições inicial e atual dos nós do reforço, conforme a Equação 3.68 e a Equação 3.69, respectivamente. Assim, os elementos imersos podem ser inseridos em qualquer localização, sem precisar coincidir com nenhum nó existente da malha, e sem adicionar graus de liberdade ao problema.

$$\overline{X_i^\beta} = \hat{\Phi}_\alpha(\xi^\beta)\hat{X_i^\alpha} \tag{3.68}$$

$$\overline{Y_i^\beta} = \hat{\Phi}_\alpha(\xi^\beta)\hat{Y_i^\alpha} \tag{3.69}$$

Para que haja a contribuição do reforço no cálculo da força interna e da hessiana do problema, escreve-se a energia de deformação como a soma dos termos referentes a

matriz e ao reforço, conforme descrito na Equação 3.70.

$$\mathbb{U} = \overline{\mathbb{U}} + \hat{\mathbb{U}} \tag{3.70}$$

Seguindo essa linha de raciocínio, escreve-se a variação da energia de deformação em função da posição, para a obtenção da força interna.

$$\frac{\partial \mathbb{U}}{\partial \hat{Y_i^\alpha}} = \frac{\partial \overline{\mathbb{U}} + \partial \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha}} = \frac{\partial \overline{\mathbb{U}}}{\partial \overline{Y_j^\beta}} \frac{\partial \overline{Y_j^\beta}}{\partial \hat{Y_i^\alpha}} + \frac{\partial \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha}} \tag{3.71}$$

sendo $(\partial \overline{Y_j^\beta} / \partial \hat{Y_i^\alpha})$ a variação da posição do nó do reforço, em função da posição do nó da matriz, dada também por:

$$\frac{\partial \overline{Y_j^\beta}}{\partial \hat{Y_i^\alpha}} = \frac{\partial \hat{\Phi}_\gamma(\xi^\beta) \hat{Y_j^\gamma}}{\partial \hat{Y_i^\alpha}} = \hat{\Phi}_\gamma(\xi^\beta) \delta_{ij} \delta_{\gamma\alpha} \tag{3.72}$$

reescrevendo-se, desta forma, a Equação 3.71:

$$\frac{\partial \mathbb{U}}{\partial \hat{Y_i^\alpha}} = \frac{\partial \overline{\mathbb{U}}}{\partial \overline{Y_i^\beta}} \hat{\Phi}_\alpha(\xi^\beta) + \frac{\partial \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha}} \tag{3.73}$$

em que $\partial \overline{\mathbb{U}} / \partial \overline{Y_i^\beta}$ descreve a componente i da força interna relacionada ao nó β de um elemento finito de reforço, $\hat{\Phi}_\alpha(\xi^\beta)$ representa a função de forma referente ao nó α de um elemento finito da matriz, sendo esta avaliada segundo as coordenadas adimensionais do nó β do elemento de reforço, e $\partial \hat{\mathbb{U}} / \partial \overline{Y_i^\alpha}$, por sua vez, representando a componente i da força interna relacionada ao nó α de um elemento finito da matriz.

Seguindo essa mesma lógica, realiza-se o cálculo da hessiana com a consideração de elementos de reforço imersos na malha da matriz. A matriz é obtida por meio da segunda variação da energia específica de deformação, em relação à posição, conforme a Equação 3.74.

$$\begin{aligned} \frac{\partial^2 \mathbb{U}}{\partial \hat{Y_i^\alpha} \hat{Y_j^\beta}} &= \frac{\partial^2 \overline{\mathbb{U}} + \partial^2 \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha} \hat{Y_j^\beta}} = \frac{\partial}{\partial \overline{Y_i^\alpha}} \left(\frac{\partial \overline{\mathbb{U}}}{\partial \overline{Y_k^\gamma}} \frac{\partial \overline{Y_k^\gamma}}{\partial \hat{Y_j^\beta}} \right) + \frac{\partial^2 \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha} \hat{Y_j^\beta}} \\ &= \frac{\partial^2 \overline{\mathbb{U}}}{\partial \overline{Y_l^\zeta} \partial \overline{Y_k^\gamma}} \frac{\partial \overline{Y_l^\zeta}}{\partial \hat{Y_i^\alpha}} \frac{\partial \overline{Y_k^\gamma}}{\partial \hat{Y_j^\beta}} + \frac{\partial^2 \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha} \hat{Y_j^\beta}} \end{aligned} \tag{3.74}$$

Da mesma forma que foi demonstrado na Equação 3.72, tem-se que:

$$\frac{\partial \overline{Y_l^\zeta}}{\partial \hat{Y_i^\alpha}} \frac{\partial \overline{Y_k^\gamma}}{\partial \hat{Y_j^\beta}} = \frac{\partial \hat{\Phi}_\eta(\xi^\zeta) \hat{Y_l^\eta}}{\partial \hat{Y_i^\alpha}} \frac{\partial \hat{\Phi}_\theta(\xi^\gamma) \hat{Y_k^\theta}}{\partial \hat{Y_j^\beta}} = \hat{\Phi}_\eta(\xi^\zeta) \delta_{il} \delta_{\eta\alpha} \hat{\Phi}_\theta(\xi^\gamma) \delta_{jk} \delta_{\theta\beta} \tag{3.75}$$

portanto, pode-se reescrever a equação da matriz hessiana, já com a contribuição do reforço, substituindo a Equação 3.75 na Equação 3.74, conforme a Equação 3.76, apresentada abaixo.

$$\frac{\partial^2 \mathbb{U}}{\partial \hat{Y_i^\alpha} \hat{Y_j^\beta}} = \frac{\partial^2 \overline{\mathbb{U}}}{\partial \overline{Y_i^\zeta} \partial \overline{Y_j^\gamma}} \hat{\Phi}_\alpha(\xi^\zeta) \hat{\Phi}_\beta(\xi^\gamma) + \frac{\partial^2 \hat{\mathbb{U}}}{\partial \hat{Y_i^\alpha} \hat{Y_j^\beta}} \tag{3.76}$$

A metodologia de cálculo para a obtenção da força interna e da hessiana, com a inclusão desses elementos de reforço, partiu da hipótese que os elementos da matriz que recebem os nós dos elementos de reforço são conhecidos. No entanto, é necessário realizar esta localização de alguma forma. Neste trabalho, o algoritmo utilizado para identificar esses elementos foi dividido em duas etapas.

Primeiramente, é feita uma pré seleção dos elementos, para diminuir o custo computacional. Nessa primeira etapa, é calculada a medida entre o nó do reforço e o centro geométrico de todos os elementos. Caso a dimensão for maior do que o raio de um círculo que circunscreve o elemento, majorado em 5% (percentual aproximado, para evitar problemas de precisão numérica, em casos onde o nó do reforço se encontra muito próximo da fronteira do elemento da matriz), o elemento entra na lista de verificação.

Na segunda etapa, o nó de reforço é escrito em função das coordenadas adimensionais de cada um dos elementos selecionados na primeira etapa, conforme a Equação 3.68. Nota-se que os valores conhecidos do problema são as coordenadas iniciais, tanto dos nós dos elementos de reforço, quanto dos nós dos elementos da matriz. Portanto, as incógnitas do problema são, justamente, as coordenadas adimensionais do nó do reforço, que satisfazem a igualdade representada na Equação 3.68. Escrevendo esse conjunto de igualdades, recai-se em um sistema de equações, que, caso o elemento da matriz tenha aproximação quadrática ou superior, é não linear. Desta forma, pode-se recorrer ao método de Newton-Raphson para a resolução desse sistema.

Na Equação 3.77 é demonstrada a expansão em série de Taylor, truncada em primeira ordem, ou seja, os termos de ordem quadrática ou superior são desconsiderados, gerando um resíduo.

$$\overline{X_i^{\beta}} = \hat{\Phi}_{\alpha}(\xi_k^m)\hat{X}_i^{\alpha} + \left(\left. \frac{\partial \hat{\Phi}_{\alpha}(\xi_k)}{\partial \xi_j} \right|_{(\xi_k^m)} \hat{X}_i^{\alpha} \right) \Delta\xi_j^m \tag{3.77}$$

sendo $\overline{X_i^{\beta}}$ as coordenadas iniciais do nó β do elemento de reforço, ξ_k^m as coordenadas adimensionais tentativas, e $\Delta\xi_j^m$ a correção dessas tentativas, cujo valor é obtido por meio da Equação 3.78.

$$\left(\left. \frac{\partial \hat{\Phi}_{\alpha}(\xi_k)}{\partial \xi_j} \right|_{(\xi_k^m)} \hat{X}_i^{\alpha} \right) \Delta\xi_j^m = \overline{X_i^{\beta}} - \hat{\Phi}_{\alpha}(\xi_k^m)\hat{X}_i^{\alpha} \tag{3.78}$$

Desta forma, o valor das coordenadas adimensionais são atualizados, conforme a Equação 3.79. O procedimento é repetido até que o valor de $\Delta\xi_j^m$ seja menor que a tolerância admitida.

$$\xi_k^{m+1} = \xi_k^m + \Delta\xi_j^m \tag{3.79}$$

Para identificar se o nó do reforço está contido no elemento da matriz, as coordenadas adimensionais obtidas no processo de solução, devem estar contidas no domínio desse

elemento. Ou seja, no caso de um elemento triangular, devem ser satisfeitas as seguintes condições: $0 \leq \xi_1^\beta \leq 1$, $0 \leq \xi_2^\beta \leq 1$ e $0 \leq 1 - \xi_1^\beta - \xi_2^\beta \leq 1$.

Desta forma, para esclarecer o procedimento de cálculo, foi elaborado um algoritmo para descrever todos os passos, vinculando cada um deles com as equações descritas neste capítulo.

Algoritmo 3: Resolução de problema com elementos imersos

para cada *elemento da matriz* **faça**
  Cálculo do centro geométrico (X_i^{CG});
  Cálculo do raio do círculo circunscrito ao elemento acrescido de 5% ($R_{0.05}$).
fim
para cada *nó de reforço* **faça**
  para cada *elemento da matriz* **faça**
    se *distância entre* $\overline{X_i}$ *e* X_i^{CG} *for menor que* $R_{0.05}$ **então**
      Adoção de uma coordenada adimensional tentativa ($\xi_k^m = \overline{X_i}/X_i^{CG}$);
      enquanto $\Delta\xi_j >$ *tolerância* **faça**
        Cálculo do $\Delta\xi_j^m$, utilizando ξ_k^m, conforme Equação 3.78;
        Atualização do ξ_k^{m+1}, conforme Equação 3.79;
        Cálculo do resíduo (norma de $\Delta\xi_j^m$).
      fim
      se $0 \leq \xi_1^m \leq 1$, $0 \leq \xi_2^m \leq 1$ *e* $0 \leq 1 - \xi_1^m - \xi_2^m \leq 1$. **então**
        Registro de que o nó de reforço incide sobre o elemento da matriz;
        Registro das coordenadas adimensionais atuais (ξ_k^m);
        Interrupção de busca para o nó de reforço atual.
      fim
    fim
  fim
fim
para cada *passo de carga* (dF_i^α) *ou posição* (dY_i^α) **faça**
  Atualização do vetor de forças externas ($F_i^\alpha = F_i^\alpha + dF_i^\alpha$) ou de posição ($Y_i^\alpha = Y_i^\alpha + dY_i^\alpha$);
  enquanto *resíduo* $>$ *tolerância* **faça**
    para cada *elemento da matriz* **faça**
      para cada *ponto de integração* **faça**
        Cálculo das funções de forma e suas derivadas;
        Cálculo do gradiente do mapeamento da configuração inicial (A_{ij}^0 - Equação 3.11) e final (A_{ij}^1 - Equação 3.12);
        Cálculo da deformação de Green-Lagrange (E_{ij} - Equação 3.14);
        Cálculo do tensor de Piola-Kirchhoff de segunda espécie (S_{ij} - Equação 3.15);
        para cada *nó* α *na direção* i **faça**
          Cálculo do vetor global de força interna ($\frac{\partial \mathbb{U}}{\partial Y_i^\alpha}$ - Equação 3.32);
          para cada *nó* β *na direção* j **faça**
            Cálculo da hessiana global ($H_{ij\alpha\beta}^k$ - Equação 3.38);
          fim
        fim
      fim
      Imposição das condições de contorno;
      Contribuição do elemento na força interna e hessiana globais;
    fim
    para cada *elemento de reforço* **faça**
      Cálculo da força interna desenvolvida no reforço, conforme Equação 3.73;
      Cálculo da hessiana desenvolvida no reforço, conforme Equação 3.76;
      Contribuição do reforço na força interna e hessiana globais.
    fim
    Resolução do sistema linear (Obtenção do ΔY_i^α - Equação 3.40);
    Atualização da posição ($(Y_i^\alpha)^{k+1}$ - Equação 3.41);
    Avaliação do resíduo $\left(\left|\left(\Delta Y_j^\beta\right)^k / X_j^\beta\right|\right)$;
  fim
fim

4 PROPOSTA DE PLATAFORMA PARA O MEF POSICIONAL

No Capítulo 3 foi demonstrada a teoria do MEF Posicional para a resolução de problemas estáticos, dinâmicos, e com a inserção de elementos na malha, desde a sua formulação, até a técnica de solução. Também foram apresentados os elementos triangulares para aproximações linear, quadrática e cúbica. Este capítulo, por sua vez, tem como objetivo expor a estrutura do código desenvolvido, de forma a conectar todas as etapas da formulação e da técnica de solução. Para o desenvolvimento desse código, foi empregada a programação orientada a objetos. Portanto, inicialmente, é apresentada uma breve descrição do método, expondo suas principais características, funcionalidades, aplicabilidades e padrão de representação.

4.1 Programação orientada a objetos

A metodologia de orientação a objetos proporciona uma série de vantagens no desenvolvimento de *softwares* se utilizada corretamente e de maneira organizada. Desta forma, é de extrema importância a adoção de padrões. Eriksson *et al.* (2004) tratam de uma linguagem unificada, denominada *Unified Modeling Language (UML)*, que proporciona aos sistemas desenvolvidos melhor visualização, especificação, construção e documentação. Segundo os autores, o termo *modeling* pode gerar certa confusão, por estar aplicado a diversos níveis. O "modelo" remete, de forma geral, ao planejamento completo do sistema, desde a descrição do problema, passando pela análise, proposição de solução, projeto e implementação. Abaixo estão citados alguns dos principais conceitos relacionados ao desenvolvimentos de modelos orientados a objetos:

- A metodologia de orientação a objetos produz modelos que refletem o domínio em questão, de forma natural, usando terminologias próprias daquele mesmo domínio;
- Conceitos subjacentes: objetos, mensagens, classes, herança e polimorfismo. Os objetos têm um certo comportamento e uma certa identidade, definidos através de nome, atributos e operações. Todo objeto acaba por ser uma instância de uma classe, sendo ela uma espécie de modelo que define as características de um objeto;
- Modelos orientados a objetos, quando construídos da forma correta, são de fácil comunicação, alteração, expansão, validação e verificação, demonstrando uma arquitetura bem definida;
- Os modelos são convenientemente implementados utilizando linguagens de programação orientadas a objeto;

- Orientação a objeto é uma tecnologia testada e utilizada em muitos projetos, construindo diferentes tipos de sistemas.

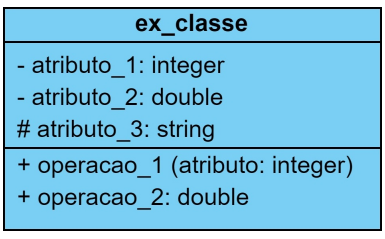
As linguagens orientadas a objetos apresentam uma determinada estrutura, e uma série de características que possibilitam a prática dos conceitos citados. Neste capítulo serão abordadas as definições de classes e objetos, assim como as interações entre eles, tirando proveito das peculiaridades da orientação a objetos como, por exemplo, herança, encapsulamento e polimorfismo.

4.1.1 Objetos e classes

Pode-se descrever os objetos como sendo entidades com estrutura e comportamento bem definidos, através de sua classe geradora (SEED, 1996). Nesse contexto, afirma-se que um objeto é uma instância de uma classe. A ideia geral da programação orientada a objetos é baseada na organização dos dados por meio dessas instâncias, que são responsáveis por enviar e receber mensagens de outros objetos. Ou seja, o código consiste em um grupo de objetos que interagem entre si (SENGUPTA; KOROBKIN, 1994). Geralmente, os objetos do código remetem a objetos reais como, por exemplo, em uma estrutura, os elementos estruturais, os vínculos externos, os carregamentos, etc. Todavia, algumas vezes são utilizadas instâncias com funções específicas, que servem apenas como ferramenta de interação entre esses objetos. É o exemplo de objetos que realizam o pré e o pós processamento do cálculo da estrutura.

Neste trabalho os dados armazenados dentro dos objetos foram denominados como atributos, e as funções responsáveis pela interação entre eles como operações. Para representar as classes de forma eficiente foram utilizados diagramas de classe UML (*Unified Modeling Language*)(ERIKSSON *et al.*, 2004). A Figura 4.1 ilustra um exemplo geral da utilização do diagrama, em que o campo superior é reservado para o nome da classe, o central para os atributos e o inferior para as operações. Os símbolos "+", "−" e "#" indicam os especificadores de acesso *public*, *private* e *protected*, respectivamente. Ainda nos atributos são identificados os tipos de dados, e nas operações os parâmetros de entrada e de saída.

Figura 4.1 – Diagrama UML de uma classe qualquer



Fonte: Próprio autor.

A forma como esses atributos e operações foram escritos e organizados será demonstrada adiante, buscando a utilização de características primordiais em um código orientado a objetos: encapsulamento, herança e polimorfismo.

4.1.2 Encapsulamento

Algumas características dos objetos não tem a necessidade de serem acessadas pela maioria dos usuários. Desta forma, o acesso a essas informações é exclusivamente interno. Portanto, essas características ficam encapsuladas dentro do objeto. Um exemplo muito interessante, trazido por Seed (1996), é o de um carro, que possui milhares de componentes, porém o motorista recebe apenas as informações necessárias para que ele possa dirigi-lo.

Dentro de uma classe há diferentes formas de especificar o acesso aos atributos do objeto, e isso é feito por meio dos especificadores de acesso citados anteriormente. O termo *public* torna os dados (atributos e operações) públicos, ou seja, podem ser acessados dentro e fora da classe por qualquer objeto. Já o especificador *private* é utilizado para restringir o acesso à classe onde os dados foram definidos. Por fim, o termo *protected* permite que subclasses derivadas da classe onde foram definidos os dados tenham acesso, porém, outros objetos não tem essa permissão. A ideia da utilização de subclasses já ingressa em outra característica primordial do código orientado a objetos, que é a herança entre classes.

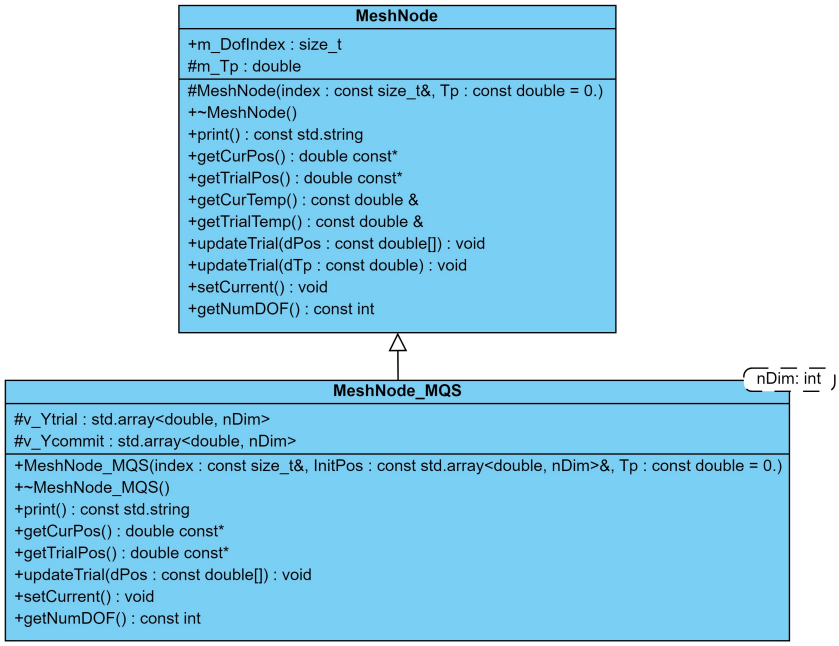
4.1.3 Herança

A escrita de um código enxuto é essencial, e a ideia de herança contribui para que isso seja possível. Basicamente, uma classe herda alguns aspectos de outra classe, para que não haja duplicidade de informações. Isso normalmente ocorre quando se tem vários "tipos" de um determinado elemento, tendo estruturas muito similares, que são herdadas. Todavia, cada um possui suas particularidades, que são adicionadas posteriormente em cada uma delas (KOENIG; MOO, 2000; YAMASSAKI, 2014).

A classe *MeshNode*, por exemplo, não chega a instanciar nenhum objeto dentro do código, sendo tratada como uma classe abstrata, ou seja, ela serve apenas de base para a formação de outras classes. Para diferentes tipos de problemas os nós podem necessitar diferentes dados, e por consequência, diferentes tipos de acesso e processamento. Desta forma, para a resolução de um problema estático cria-se a subclasse *MeshNode _MQS* a partir da classe *MeshNode*. Assim, todo o conteúdo da classe *MeshNode* é herdado pela subclasse *MeshNode _MQS*, evitando a escrita de códigos repetitivos. A Figura 4.2 apresenta o diagrama UML dessas classes.

Nesse processo, há a necessidade de acrescentar atributos e/ou operações, ou ainda de alterar algumas dessas operações. Quando uma operação da classe é alterada em mais de uma das suas subclasses é utilizada a ideia de polimorfismo.

Figura 4.2 – Diagrama UML da classe *MeshNode* e a derivada *MeshNode _MQS*



Fonte: Próprio autor.

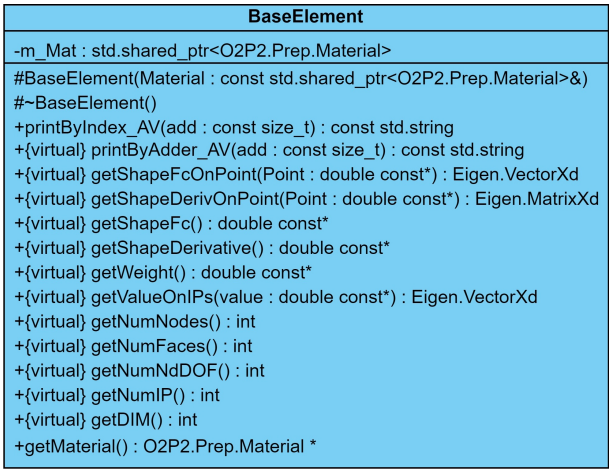
4.1.4 Polimorfismo

Polimorfismo é caracterizado quando duas ou mais classes, que são derivadas de uma mesma classe, possuem operações que podem ser chamadas através do mesmo nome, porém, poderão ter características específicas em cada uma das classes (SENGUPTA; KOROBKIN, 1994). A utilização dessa técnica proporciona uma padronização dos atributos entre as classes de uma mesma "família". Se tivermos uma classe "pai", por exemplo, representando formas geométricas, e como derivadas, as classes retângulo, triângulo, círculo e trapézio. Para cada uma delas, tem-se uma operação que retorna, por exemplo, a área do elemento. O emprego do polimorfismo faz com que a chamada dessa operação seja sempre igual, para qualquer classe derivada.

Essa propriedade foi aplicada diversas vezes no código. Na Figura 4.3 pode-se observar a classe *BaseElement*, da qual derivam os diferentes tipos de elementos. Praticamente todas as operações são implementadas como *virtual*, permitindo a utilização da mesma nomenclatura nas classes derivadas, sobrescrevendo a operação em cada uma delas. Desta forma, citando um exemplo, toda vez que for solicitado, em qualquer elemento, o número de graus de liberdade dos seus nós, será chamada a operação *getNumNdDOF*.

Da mesma forma que é utilizada a ideia de herança como alternativa para evitar a concepção de códigos repetitivos, o uso de *templates* auxilia significativamente nesse sentido.

Figura 4.3 – Diagrama UML da classe *BaseElement*



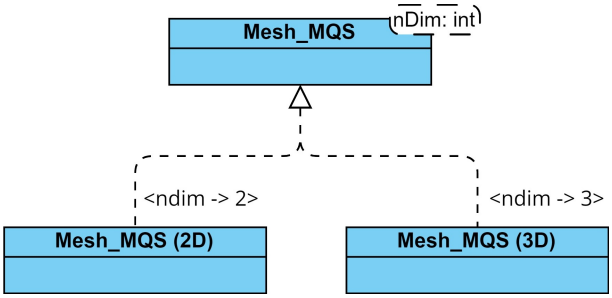
Fonte: Próprio autor.

4.1.5 Templates

As classes, como visto anteriormente, são protótipos para a criação de objetos. Pode-se visualizar a ideia de *template* como um protótipo de classe ou função com parâmetros de entrada variáveis. Conforme Booch *et al.* (2007), *template* é uma família de classes cuja estrutura e comportamento estão definidos, independente dos seus parâmetros. Um aspecto interessante dessa ferramenta é que o seu funcionamento se dá em tempo de compilação, e as classes derivadas desse "protótipo" só são geradas quando são chamadas no código. Ou seja, caso não haja nenhuma instanciação, indicando os parâmetros de entrada, o conteúdo dessa classe ou função sequer é compilado.

Dentro do exemplo da classe *Mesh* foi utilizada essa ferramenta para evitar a reescrita da classe para problemas com domínio de diferentes dimensões (2D e 3D). A representação do *template* da classe *Mesh _MQS* pode ser visualizada na Figura 4.4. O único parâmetro de entrada é o *ndim* (restrito a dados do tipo *integer*), ao qual foram atribuídos os valores 2 e 3, representando domínios bidimensionais e tridimensionais.

Figura 4.4 – Diagrama UML do *template* da classe *Mesh _MQS*



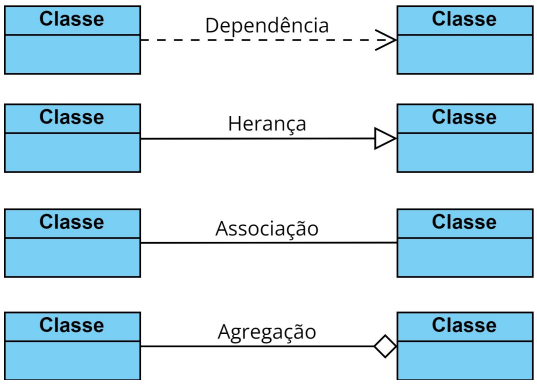
Fonte: Próprio autor.

Ao longo deste capítulo foram tratadas as principais características de um código orientado a objetos, assim como a forma de representar a construção desses códigos por meio de diagramas, segundo a linguagem unificada (UML). Para a representação de diagramas completos, é indispensável uma nomenclatura padrão para o relacionamento entre seus elementos.

4.1.6 Relacionamentos

A ideia da representação de códigos fazendo o uso de diagramas UML, proporciona não só o entendimento de uma classe em si, - por meio da descrição dos atributos e operações - mas também da interação entre as diferentes classes. Para que não haja problemas de ambiguidade, são utilizadas diferentes simbologias para especificar o relacionamento entre os elementos contidos em um diagrama. A Figura 4.5 mostra os tipos de relacionamentos utilizados neste trabalho.

Figura 4.5 – Relacionamentos entre elementos



Fonte: Próprio autor.

A relação de dependência indica que, de alguma forma, um elemento depende do outro. Já a simbologia de Herança, previamente utilizada na Figura 4.2, indica que a classe é derivada de uma outra, ou seja, herda todas as características da classe "pai", somando às suas características adicionais. O vínculo de associação conecta elementos e associa instâncias. A relação de agregação não deixa de ser um tipo de associação, porém, onde um elemento contém outro elemento. No caso de conter mais de um, a simbologia permanece a mesma, porém, o losango que consta na extremidade do contentor passa a ser preenchido.

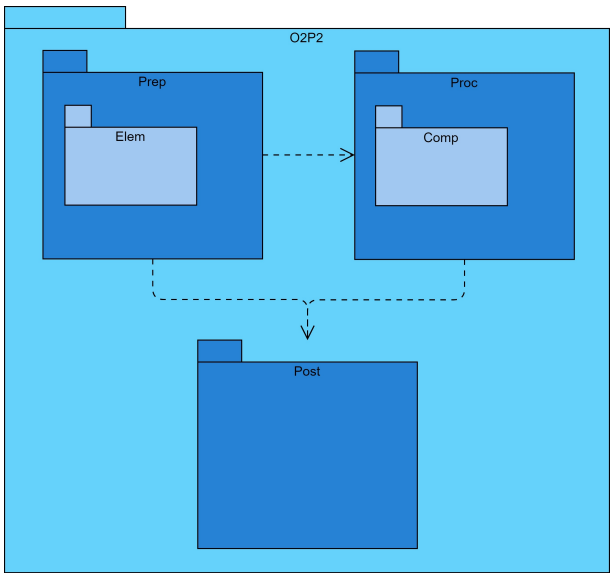
4.2 Estrutura do código

Analisando o Algoritmo 1 apresentado no Capítulo 3, pode-se notar que se trata de um procedimento iterativo e incremental de cálculo, devido à consideração da não linearidade geométrica e a aplicação do carregamento (ou deslocamento prescrito) por

meio de um determinado número de incrementos. Levando em conta esse procedimento de cálculo, buscou-se a concepção de uma arquitetura otimizada, de modo a aproveitar ao máximo as características positivas que a orientação a objetos proporciona, como a fácil manutenibilidade e escalabilidade. A grande dimensão dos problemas que serão resolvidos por meio da plataforma também gera a necessidade de controlar ao máximo a movimentação de dados entre os objetos e namespaces. Todavia, por mais que o tempo de execução sempre foi levado em consideração durante a concepção do código, a pesquisa não tem o objetivo de avaliar o seu tempo de processamento.

A estrutura do código é apresentada abaixo, através de linguagem UML, com diferentes níveis de detalhes, de forma a compreender desde uma visão mais generalizada, até o funcionamento de cada uma de suas classes. Por meio da Figura 4.6 são observados os principais *namespaces* utilizados, organizando as classes dentro de três principais ambientes: *Prep* (Pré processamento), *Proc* (Processamento) e *Post* (Pós processamento). O ambiente *Elem* foi criado com o objetivo de organizar todos os dados relacionados aos diferentes tipos de elementos disponibilizados. Já o ambiente *Comp* detém todos os dados utilizados na análise.

Figura 4.6 – Visão geral da arquitetura - namespaces



Fonte: Próprio autor.

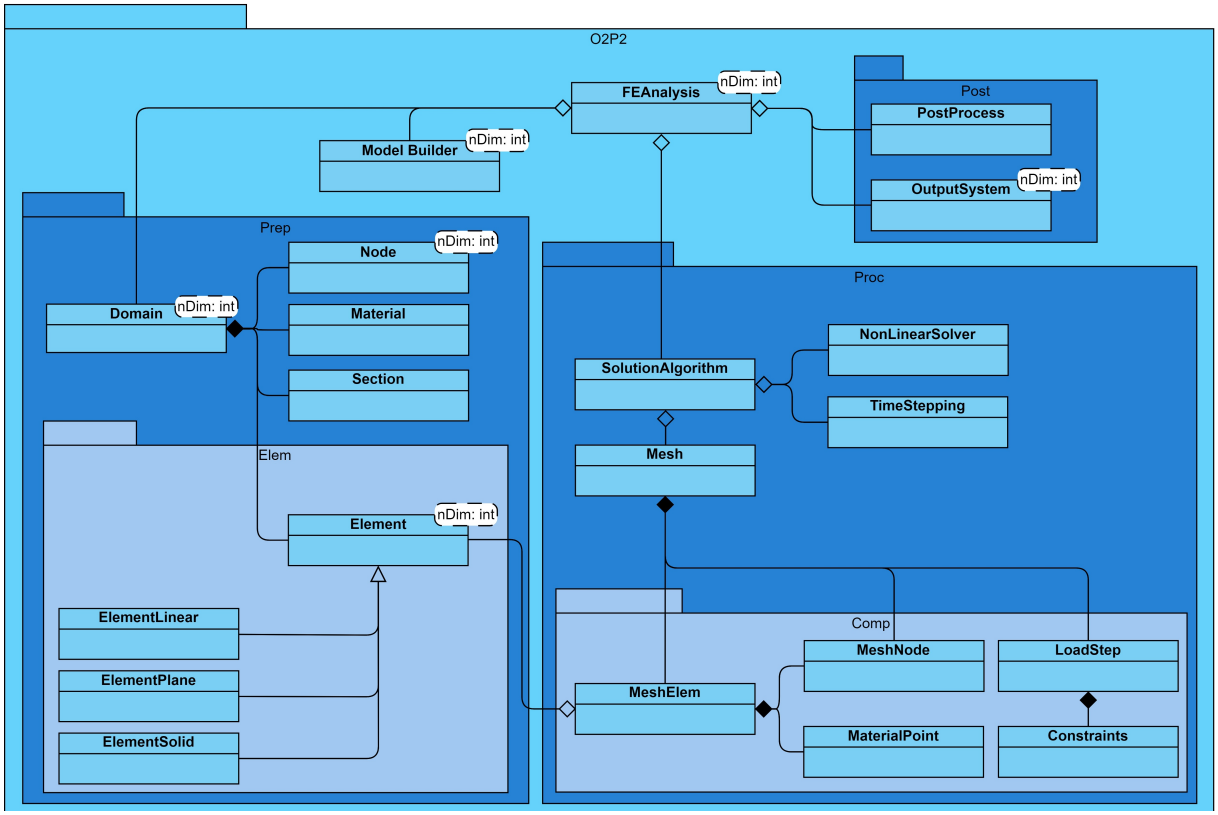
Com o objetivo de ter um melhor entendimento do funcionamento geral da plataforma, foi criado um diagrama de classes geral (Figura 4.7), suprimindo seus atributos e operações, possibilitando a apresentação em um único diagrama.

A classe *FEAnalysis* comanda todo o processo. Inicialmente, por meio do acionamento da *ModelBuilder*, que realiza a leitura dos arquivos de entrada, é criado o domínio do problema, armazenado em uma única instância da classe *Domain*. Esse objeto contém

instâncias das classes *Node*, *Material*, *Section* e *Element*, responsáveis por armazenar dados referentes aos nós, materiais, seções e elementos do problema, respectivamente.

Uma vez que o domínio do problema está construído, a classe *FEAnalysis* aciona a única instância da classe *SolutionAlgorithm*, com o objetivo de criar toda a estrutura de dados necessários para a realização da análise nas classes de processamento. Dessa forma, os dados do domínio do problema, que compõem a etapa de pré processamento, permanecem estáticos, e a resolução do problema se resume ao ambiente de processamento. Os dados acessados durante o procedimento de cálculo estão contidos em instâncias das classes *MeshElem*, *MeshNode*, *MaterialPoint*, *LoadStep*, *Constraints* e em suas classes derivadas.

Figura 4.7 – Visão geral da arquitetura - classes



Fonte: Próprio autor.

Com todos os dados a serem utilizados na análise já organizados, a classe *SolutionAlgorithm* inicia o processo não linear iterativo de cálculo, por meio de instâncias das classes *TimeStepping* e *NonLinearSolver*. O resultado do cálculo nada mais é do que as posições atuais dos nós da estrutura, ou seja, a sua deformada. De posse desses dados, a *FEAnalysis* prossegue acionando o ambiente de pós processamento, por meio das instâncias das classes *PostProcess* e *OutputSystem*. A primeira tem por objetivo o tratamento dos resultados para a geração dos dados que serão apresentados. Já a segunda visa a criação dos

arquivos de apresentação dos resultados, para visualização em softwares como o Paraview[1] e o Acadview[2].

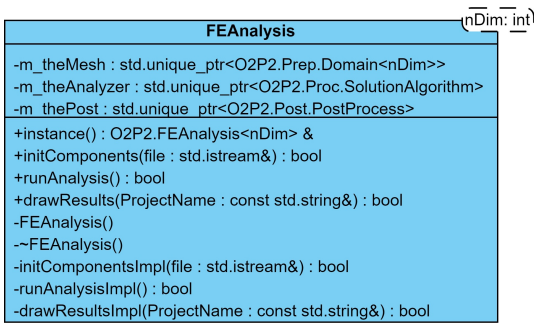
A Figura 4.7 possibilita o entendimento geral da plataforma. Todavia, sem a descrição dos atributos e operações não é possível entender o funcionamento do código. Desta forma foram elaborados diagramas UML completos das principais classes.

4.2.1 Diagramas de classes

Nesta seção são apresentados os diagramas das classes mais relevantes que compõem a plataforma, para que, por meio da descrição de seus atributos e operações, seja possível entender o seu funcionamento e sua função dentro do código. Essas classes de maior relevância são: *FEAnalysis*, *ModelBuilder*, *Domain*, *SolutionAlgorithm*, *Mesh*, *OutputSystem* e *PostProcess*.

A *FEAnalysis* é responsável por comandar todo o processo, desde a leitura dos arquivos de entrada, passando pelo armazenamento e processamento dos dados, até a geração dos arquivos de saída. Pode-se observar, por meio da Figura 4.8, que a classe armazena as instâncias únicas das classes *Domain*, *SolutionAlgorithm* e *PostProcess*, nomeadas, respectivamente, como *theMesh*, *theAnalyzer* e *thePost*. Desta forma, ela detém todos os dados relacionados com o pré processamento, com o processamento e com o pós processamento. Também é possível observar por meio da Figura 4.8 é o parâmetro de template *nDim*. Conforme demonstrado no Capítulo 4, a classe é definida em função desse parâmetro, e reescrita, em tempo de compilação, para cada um dos valores definidos. Neste caso foram definidos apenas os valores 2 e 3 para o parâmetro nDim, adaptando o código para problemas bidimensionais e tridimensionais, respectivamente.

Figura 4.8 – Diagrama de classes: *FEAnalysis*



Fonte: Próprio autor.

A operação *initComponents* é responsável pela leitura dos arquivos de entrada, assim

[1] Disponível em: <https://www.paraview.org/>. Versão: 5. Acesso em: Outubro de 2023.

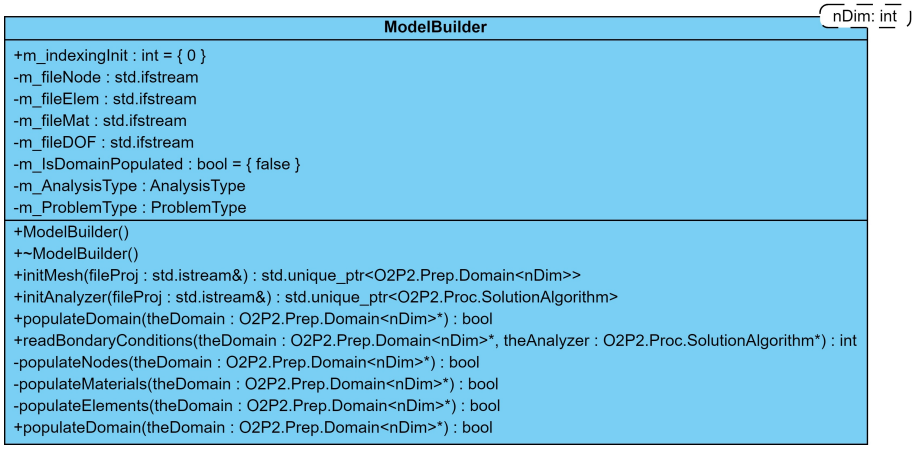
[2] Disponível em: <https://set.eesc.usp.br/software/acadview/>. Versão: 3. Acesso em: Outubro de 2023.

como o armazenamento dos dados do problema. Já a *runAnalysis* realiza o processamento dos dados. Por fim, a operação *drawResults* cria e preenche os arquivos de saída.

A classe *ModelBuilder* desempenha um papel essencial na construção do domínio do problema. Armazena os arquivos de entrada *fileNode*, *fileElem*, *fileMat*, *fileDOF*, que detém os dados referentes aos nós, elementos, materiais e graus de liberdade, respectivamente.

Acionada por meio da classe *FEAnalysis*, realiza a leitura desses arquivos, e cria os objetos que compõem o domínio do problema, armazenando-os em uma instância da classe *Domain*. Como pode ser visto na Figura 4.9, além dos arquivos de entrada citados, a classe detém atributos relacionados ao tipo de análise (*AnalysisType*) e ao tipo de problema (*ProblemType*). Uma vez que a construção do domínio é finalizada, a classe já cumpriu o seu papel, e pode ser descartada sem causar nenhum prejuízo.

Figura 4.9 – Diagrama de classes: *ModelBuilder*



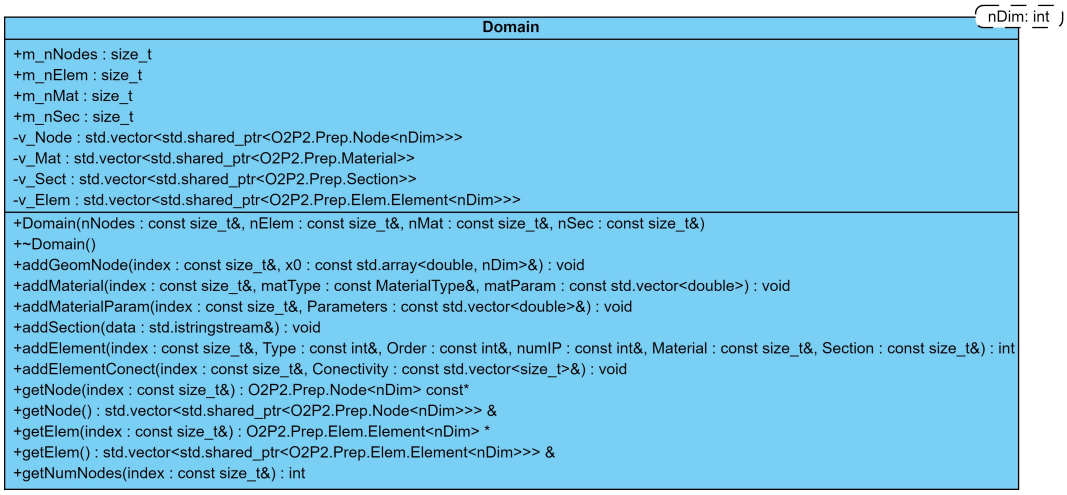
Fonte: Próprio autor.

Instanciada uma única vez, a classe *Domain* representa o domínio do problema. Seus principais atributos (Figura 4.10) são os *containers*, que guardam informações de cada um dos nós (*Node*), materiais (*Mat*), seções (*Sect*) e elementos (*Elem*). Algumas de suas operações são utilizadas pela *ModelBuilder* com o objetivo de popular os *containers* citados. Uma vez que o domínio do problema está montado, ou seja, assim que os arquivos de entrada são lidos, os dados desse objeto permanecem estáticos, porém, ainda guardados na memória. As operações com o prefixo *get* são utilizadas para acessar esses dados durante as etapas de processamento e pós processamento.

Dentre os atributos detidos pela classe, os elementos são os que apresentam maior complexidade na sua estrutura de dados. Como pôde ser visto na Figura 4.7, foi criado um *namespace* especial para a organização desses dados. De forma geral, todo elemento é instanciado a partir de sua classe geradora que, dependendo da dimensão desse elemento, é derivada das classes *ElementLinear*, *ElementPlane*, ou *ElementSolid*, se tratando de elementos unidimensionais, bidimensionais e tridimensionais, respectivamente. Nesta pes-

quisa foram implementados apenas elementos planos triangulares, com aproximação linear, quadrática e cúbica.

Figura 4.10 – Diagrama de classes: *Domain*



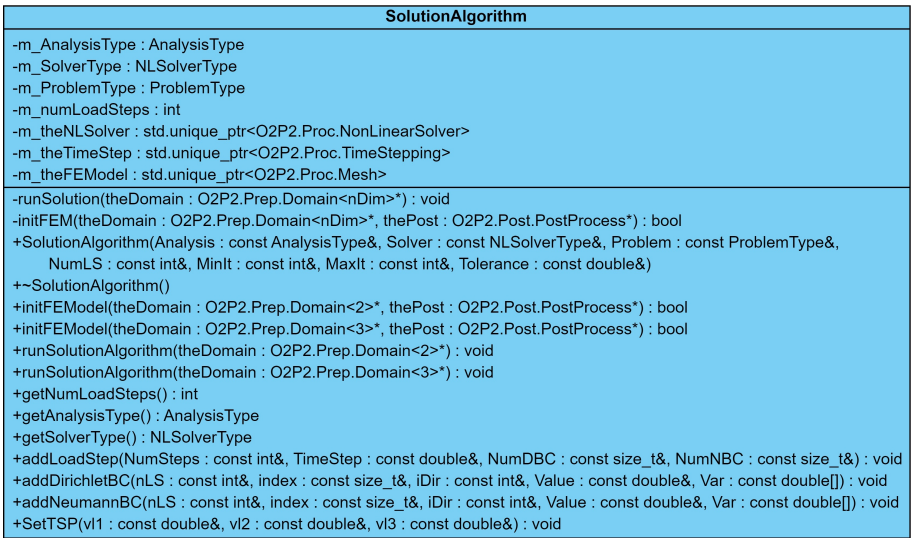
Fonte: Próprio autor.

Finalizada a construção do domínio do problema, é acionada a *SolutionAlgorithm*, que é responsável por comandar a etapa de processamento. Em um primeiro momento, a classe é encarregada de organizar todos os dados a serem utilizados na análise dentro de uma instância única da classe *Mesh*, armazenada na própria *SolutionAlgorithm*. Essa estratégia faz com que, durante o período de processamento, haja acesso e movimentação de dados somente dentro do ambiente de processamento, mantendo o ambiente de pré processamento estático, como já foi dito anteriormente. Na etapa seguinte, com todos os dados já organizados, são acionados os objetos das classes *NonLinearSolver* e *TimeStepping*, também detidos pela *SolutionAlgorithm*, para que seja realizado o procedimento iterativo e incremental de cálculo.

A única instância da classe *Mesh* é incumbida de armazenar todos os dados acessados durante a etapa de processamento. Na Figura 4.12 é possível visualizar esses atributos, principalmente os vetores contendo os nós e elementos da malha, nomeados como *meshNode* e *meshElem*, respectivamente. Um aspecto importante da classe é que seus atributos e operações foram construídos de forma mais genérica, para que ela possa se adequar aos diversos tipos de problemas (estáticos, dinâmicos, elementos imersos, térmicos, etc.). Em virtude disso, em operações que são comuns a várias tipologias, porém, que apresentam cálculos diferenciados em cada uma delas, foi utilizada a característica de polimorfismo, definindo essas operações como virtuais em sua classe pai. É o caso da *assembleSOE*, da *setTimeStep* e da *imposeNeumannBC*, responsáveis por montar o sistema de equações, definir o passo de tempo e impor condições de contorno, respectivamente. Desta forma, permite-se implementações distintas em suas classes derivadas, conforme as especificidades

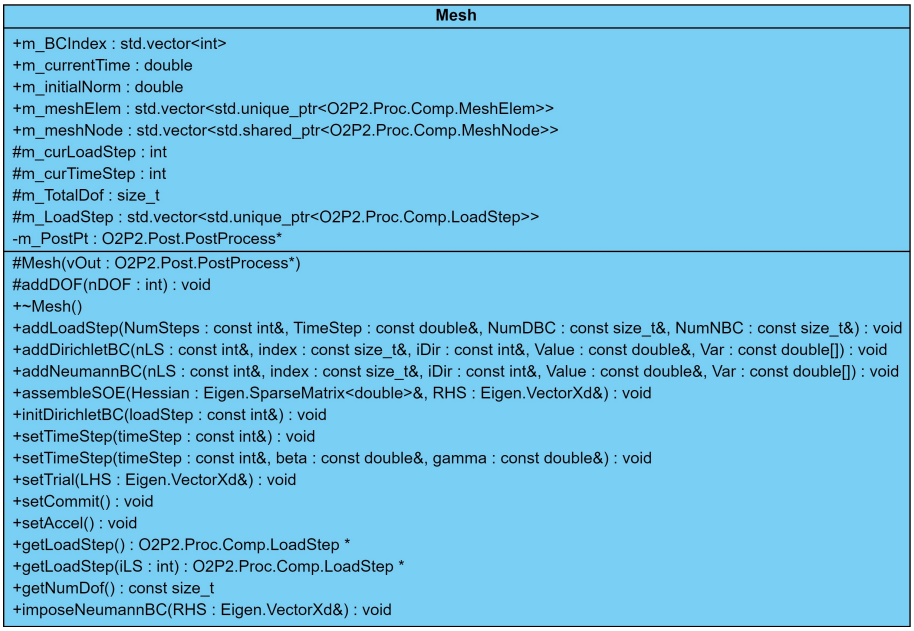
de cada uma, ainda que a operação permaneça com o mesmo nome.

Figura 4.11 – Diagrama de classes: *SolutionAlgorithm*



Fonte: Próprio autor.

Figura 4.12 – Diagrama de classes: *Mesh*



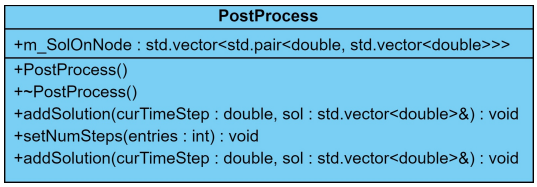
Fonte: Próprio autor.

A classe *PostProcess* contém os atributos e operações (Figura 4.13) responsáveis por tratar os dados obtidos na análise, de forma a obter a saída de dados desejada.

Já a *OutputSystem* é responsável por construir os arquivos para visualização dos resultados obtidos na análise por meio dos dados gerados pela *PostProcess*. Neste trabalho

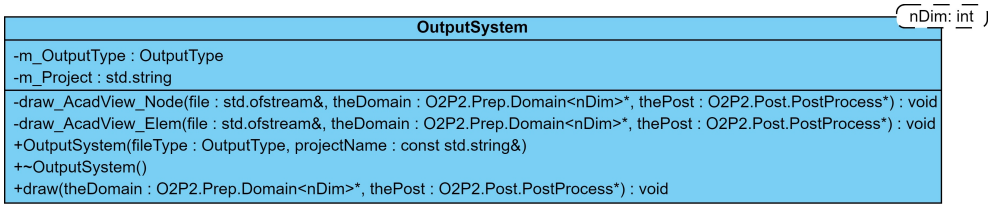
foi implementada apenas a saída de dados para visualização no software AcadView, como pode ser visto na Figura 4.14.

Figura 4.13 – Diagrama de classes: *PostProcess*



Fonte: Próprio autor.

Figura 4.14 – Diagrama de classes: *OutputSystem*



Fonte: Próprio autor.

4.2.2 Documentação do código

A documentação foi gerada por meio do *software* Doxygen®[3]. Os textos descritivos de cada classe, atributo, operação, etc. foram adicionados diretamente no código, fazendo com que a concepção da documentação ocorresse automaticamente. A inserção dessas informações devem atender a um formato específico, para que seja possível categorizá-las e organizá-las. Na Figura 4.15 pode-se observar a indicação dos dados da classe *Node*, envolvendo também o construtor, onde pode-se observar as chamadas para as informações fornecidas.

O texto final, disponibilizado por meio do repositório O2P2: Pre-alpha release (CARRAZEDO *et al.*, 2023) - assim nomeado (O2P2) devido a contração de *Object Oriented Platform for Positional FEM* - , foi disposto em arquivos html, podendo ser visualizado em qualquer navegador. Por meio da Figura 4.16 pode-se observar a página inicial da documentação, que permite, por meio do menu lateral, o acesso a todos os dados descritos no código, categorizados, conforme a sua hierarquia. No trecho inicial, foram inseridas informações gerais da plataforma, como os seus recursos, as características da licença, os autores envolvidos, forma de citação, etc. Também consta um passo a passo para a sua utilização, com orientações relacionadas ao local de acesso ao código, e também ao modo de execução.

[3] Disponível em: <https://doxygen.nl/index.html/>. Versão: 1.9.8. Acesso em: Outubro de 2023.

Figura 4.15 – Detalhe da inserção da documentação no código

```
17  namespace O2P2 {
18      namespace Geom {
19          /** @ingroup PreProcessor_Module
20           * @class Node
21           *
22           * @brief Geometry node, a Domain component.
23           * @details This base class only provides interface for derived classes, which contains initial information on nodes.
24           */
25          class Node
26          {
27          private:
28              Node() = delete;
29
30          protected:
31              /** Constructor for geometry node objects.
32               * @param index Node index.
33               * @param Tp Initial temperature.
34               */
35              explicit Node(const size_t& index, const double Tp = 0.)
36                  : mv_index(index), mv_Tp(Tp) {}
37
```

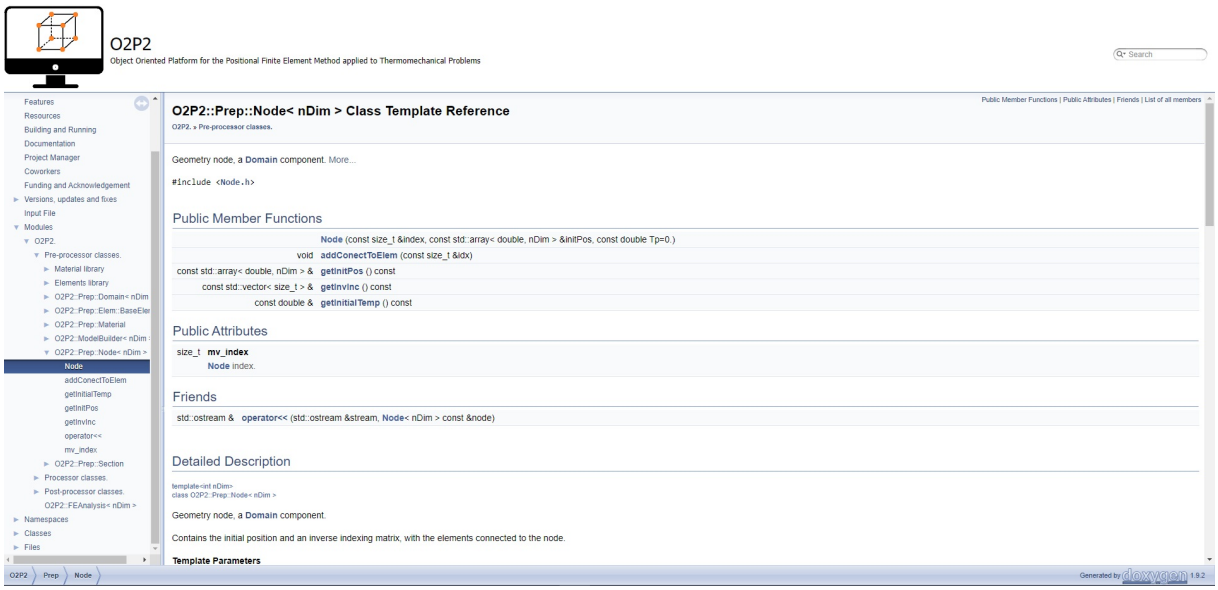
Fonte: Repositório O2P2: Pre-alpha release (CARRAZEDO *et al.*, 2023).

Figura 4.16 – Documentação: Página inicial

Fonte: Repositório O2P2: Pre-alpha release (CARRAZEDO *et al.*, 2023).

Na Figura 4.17 pode-se visualizar a organização e a forma de apresentação dos atributos, das operações e das descrições da classe *Node*. Todas as informações estão conectadas, por meio de links, facilitando o entendimento do usuário, proporcionando uma navegação rápida entre os elementos da plataforma. A hierarquia de classes também fica explícita por meio de fluxogramas interativos, fazendo com que a estrutura do código seja de entendimento fácil, sem que haja a necessidade de inspecionar códigos para compreender as relações entre as classes.

Figura 4.17 – Documentação: Classe *Node*



Fonte: Repositório O2P2: Pre-alpha release (CARRAZEDO *et al.*, 2023).

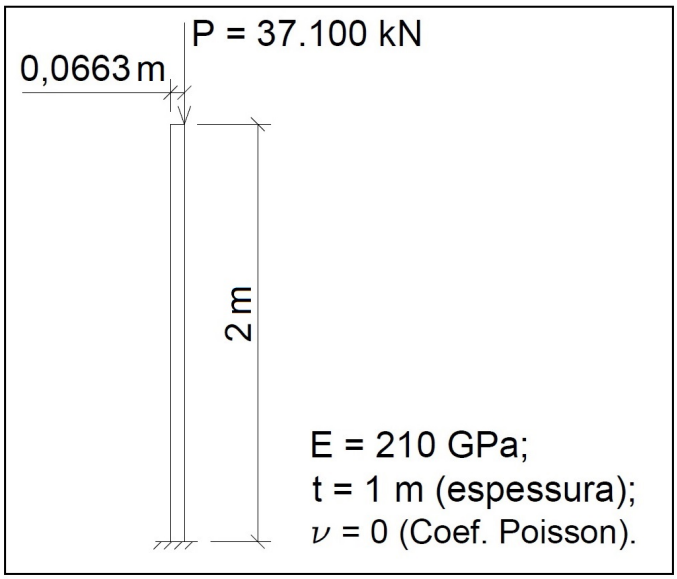
5 EXEMPLOS DE APLICAÇÃO

Para a validação do código elaborado, foram simulados alguns exemplos conhecidos, retirados de outros trabalhos. Os exemplos buscam avaliar características específicas relacionadas ao MEF Posicional, como problemas de grandes deslocamentos e/ou rotações, porém, com deformações moderadas, devido a utilização do modelo constitutivo de Saint-Venant-Kirchhoff. Para a geração da malha foi utilizado o *software* de pré-processamento *AcadMesh2D* (PIEDADE NETO; FAGA JÚNIOR; PACCOLA, 2012), e para a visualização dos resultados foi utilizado o *software* de pós-processamento *AcadView* (CODA; PACCOLA, 2005).

5.1 Exemplo 1: Linha elástica de Euler

O primeiro exemplo, retirado do trabalho de Marques (2006), trata-se de um pilar com a base engastada e o topo livre, com uma carga vertical para baixo aplicada no topo de forma excêntrica. Esse problema também é conhecido como linha elástica de Euler, podendo também ser encontrado nos trabalhos de Fujii (1983) e Simo, Hjelmstad e Taylor (1984). Os parâmetros e dimensões utilizados para essa análise foram os mesmos utilizados por Marques (2006), e podem ser visualizados na Figura 5.1.

Figura 5.1 – Exemplo 1: Linha elástica de Euler

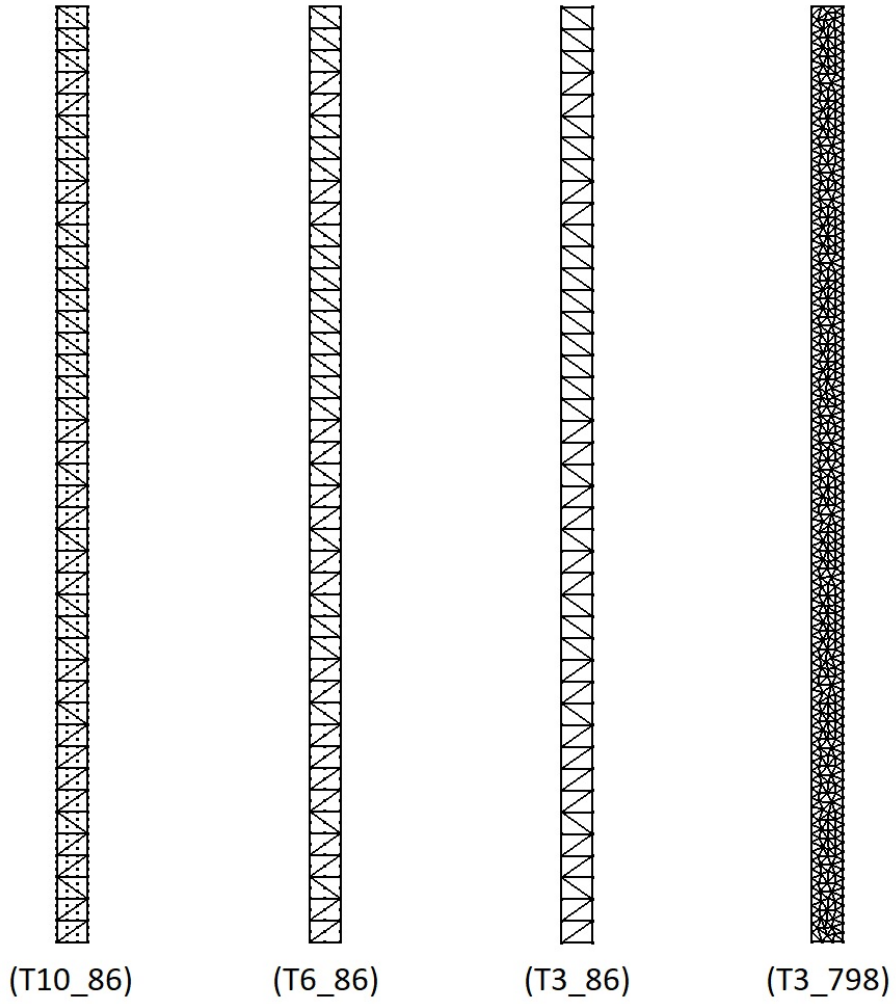


Fonte: Próprio autor.

A estrutura foi discretizada de quatro formas diferentes de modo a comparar os resultados obtidos com elementos triangulares de diferentes ordens. Essa variação de discretização pode ser visualizada por meio da Figura 5.2. O primeiro modelo, denominado

T10_86, foi discretizado em 86 elementos triangulares com aproximação cúbica (10 nós). O segundo e o terceiro, denominados, respectivamente, T6_86 e T3_86, também possuem 86 elementos triangulares, porém, com aproximação quadrática e linear. O quarto modelo, T3_798, foi discretizado com elementos triangulares lineares, porém, mantendo a mesma quantidade de nós que o primeiro modelo, totalizando 798 elementos.

Figura 5.2 – Exemplo 1: Discretização dos modelos



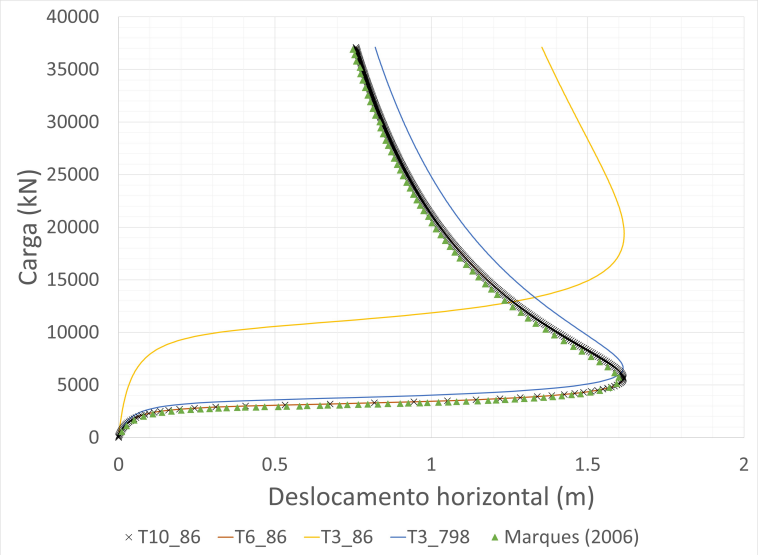
Fonte: Próprio autor.

A carga de 37.100 kN foi aplicada em 371 passos de carga, com 100 kN cada. Na Figura 5.3 pode-se visualizar a variação do deslocamento horizontal no topo do pilar em função do carregamento aplicado, para os quatro modelos de discretização, assim como para o modelo analisado por Marques (2006). A obtenção desses resultados só é possível devido à consideração da não linearidade geométrica na formulação que, no MEF Posicional, é obtida de forma natural.

Nota-se que os valores obtidos nos modelos com 86 elementos triangulares quadráticos (T6_86) e cúbicos (T10_86) estão muito próximos entre si, assim como do modelo

de referência, analisado por Marques (2006), que conta com 80 elementos triangulares de aproximação cúbica. Já o modelo com 86 elementos lineares (T3_86) não apresentou um resultado satisfatório, divergindo do modelo já validado, demonstrando que a variação linear de deformações dentro do elemento é insuficiente para a resolução deste exemplo, nesta ordem de discretização. O modelo T3_798, que conta com 798 elementos, já apresenta uma malha mais refinada, porém, embora tenha o mesmo número de nós que o primeiro modelo (T10_86), ainda demonstra resultados distintos da referência.

Figura 5.3 – Exemplo 1: Deslocamento horizontal (m) x Carga (kN)



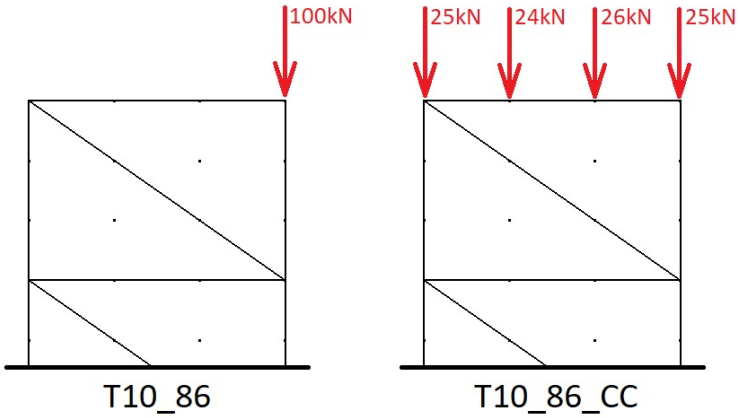
Fonte: Próprio autor.

Os resultados obtidos também foram comparados com o trabalho de Fujii (1983), onde foram plotados os valores de deslocamento horizontal (δ_h) relativo ao comprimento do pilar (L), em função da carga relativa à carga crítica de Euler (P_{cr}). Para essa análise, foi criado um quinto modelo, denominado T10_86_CC, com a mesma discretização do T10_86, ou seja, 86 elementos triangulares com aproximação cúbica. Todavia, ao invés de realizar a aplicação da carga na extremidade do pilar, foi aplicada uma carga com menor excentricidade para induzir a instabilidade no pilar. Essa diferença pode ser vista na Figura 5.4, que representa a aplicação de um passo de carga nos modelos T10_86 e T10_86_CC.

A sobreposição das curvas obtidas pode ser visualizada na Figura 5.5. Da mesma forma que pôde ser visualizado na Figura 5.3, os modelos com elementos de aproximação linear divergem da referência, que é o modelo proposto por Fujii (1983). Já os modelos com elementos de aproximação quadrática e cúbica apresentaram um resultado muito próximo ao da referência, com exceção do trecho inicial, antes de atingir a carga crítica. Essa diferença entre os modelos ocorre justamente devido à aplicação da carga de forma excêntrica, que acarreta em um deslocamento horizontal prévio. Com a aplicação da

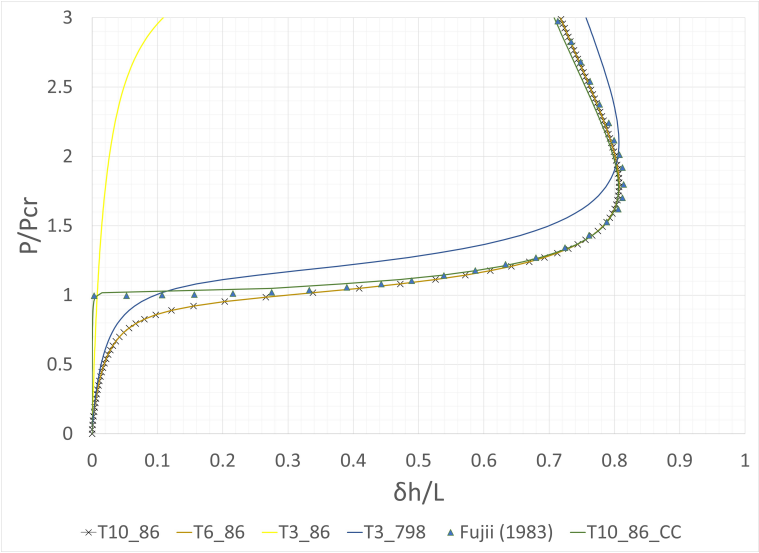
carga centrada, por meio do modelo T10_86_CC, pode-se observar que a ocorrência da instabilidade se dá muito próximo à carga crítica de Euler, demonstrando que a divergência entre a referência e os modelos propostos advém da aplicação do carregamento de forma excêntrica.

Figura 5.4 – Diferença de aplicação do carregamento



Fonte: Próprio autor.

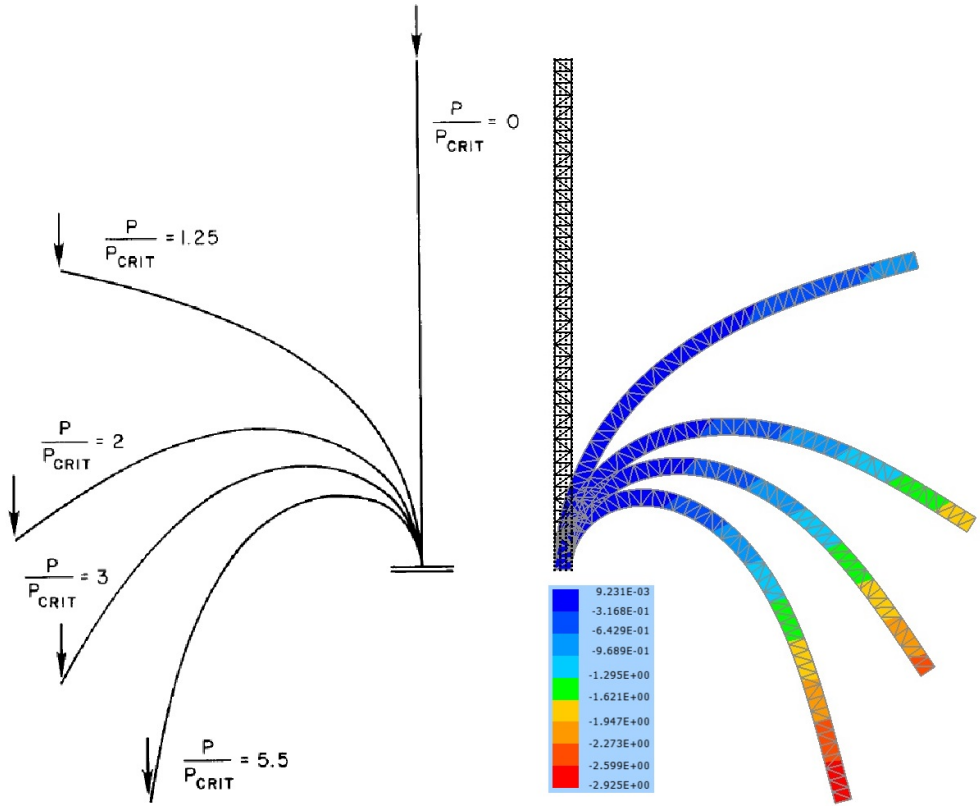
Figura 5.5 – Deslocamento horizontal relativo x Carga relativa



Fonte: Próprio autor.

Por fim, a Figura 5.6 representa a comparação entre o deslocamento vertical da referência, e do modelo T10_86_CC, para diferentes níveis de carregamento.

Figura 5.6 – Deslocamento vertical

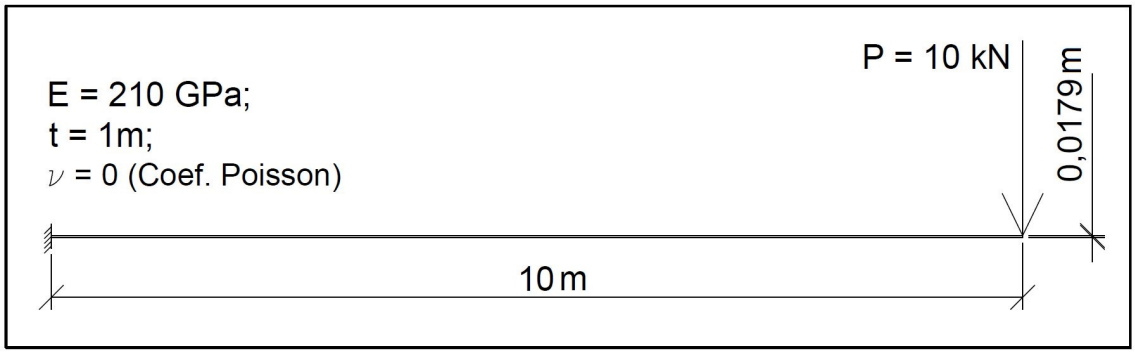


Fonte: Adaptado de Fujii (1983).

5.2 Exemplo 2: Viga engastada-livre com carga transversal aplicada na extremidade

O segundo exemplo se trata de uma viga com a extremidade esquerda engastada e a direita livre, submetida a um carregamento transversal aplicado em seu extremo livre. Os parâmetros utilizados nessa análise foram os mesmos utilizados por Greco (2004), e podem ser visualizados na Figura 5.7.

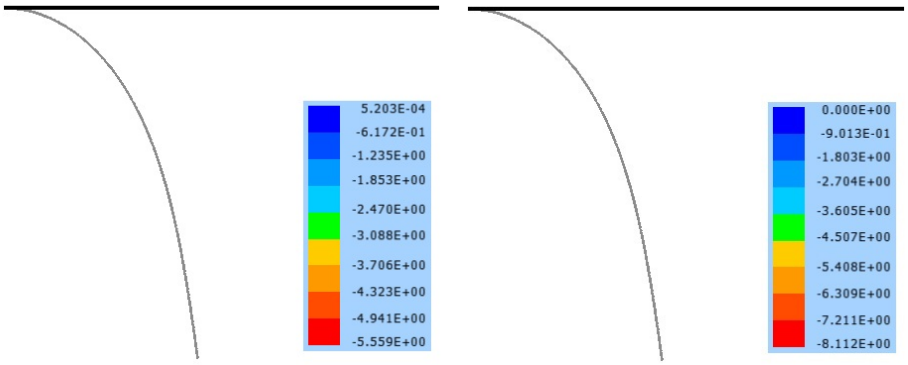
Figura 5.7 – Exemplo 2: Viga engastada-livre



Fonte: Próprio autor.

A estrutura foi discretizada em 578 elementos triangulares com aproximação cúbica (10 nós), e a carga total (10 kN) foi aplicada em 100 passos de carga de 100N cada. Os deslocamentos horizontal e vertical são apresentados na Figura 5.8, e representam a situação gerada pelo máximo carregamento aplicado (10kN).

Figura 5.8 – Exemplo 2: Deslocamento com carga máxima (10kN)

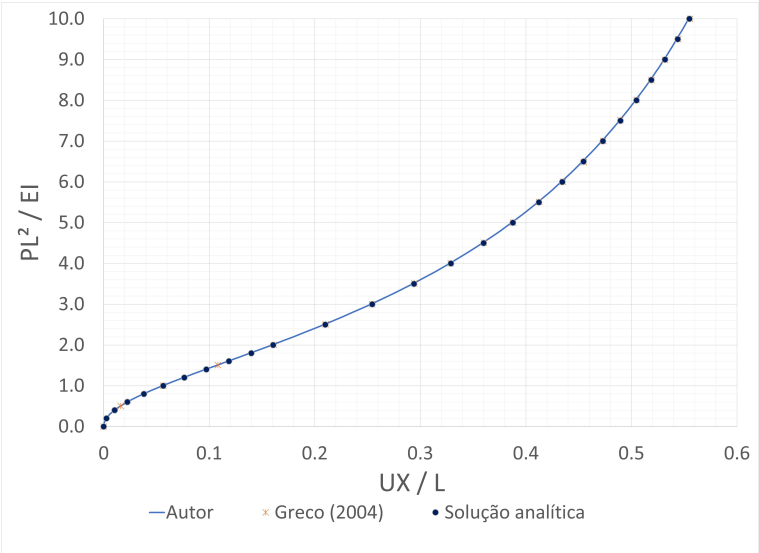


(a) Deslocamento horizontal. (b) Deslocamento vertical.

Fonte: Próprio autor.

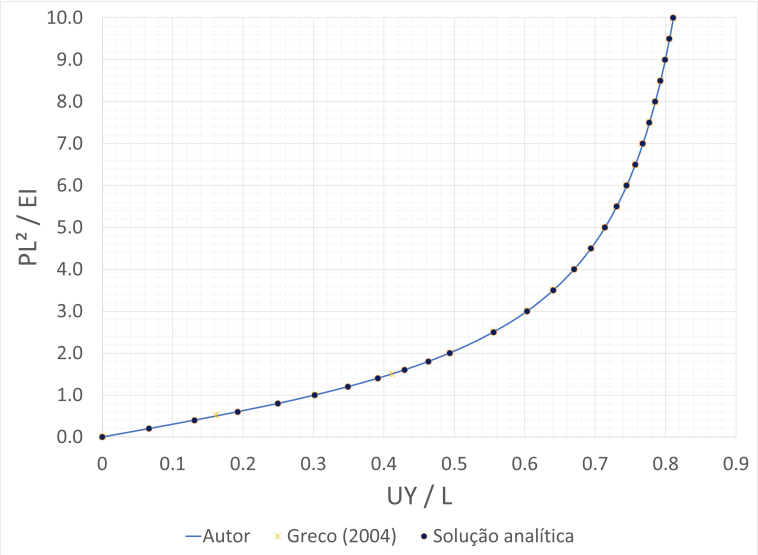
A Figura 5.9 ilustra as variações do deslocamento horizontal e a Figura 5.10 do deslocamento vertical, ambas medidas no ponto de aplicação da carga. Os resultados foram comparados à solução analítica encontrada nos trabalhos de Fujii (1983), Greco (2004) e também ao trabalho de Mattiasson (1981). É possível notar um resultado muito próximo ao encontrado por esses autores.

Figura 5.9 – Exemplo 2: Desl. horizontal x Carga (Adim.)



Fonte: Próprio autor.

Figura 5.10 – Exemplo 2: Desl. vertical x Carga (Adim.)

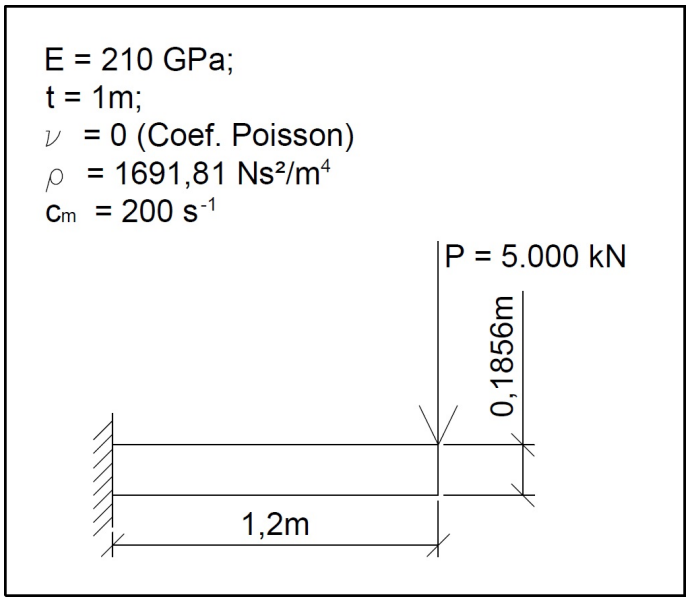


Fonte: Próprio autor.

5.3 Exemplo 3: Viga engastada-livre com amortecimento

O problema, trazido tanto por Greco (2004), como por Marques (2006), se trata de uma viga com a extremidade esquerda engastada e a direita livre, submetida a um carregamento transversal constante aplicado em seu extremo livre. Foi realizada a análise transiente, com a consideração de amortecimento. Os dados considerados no exemplo podem ser visualizados na Figura 5.11.

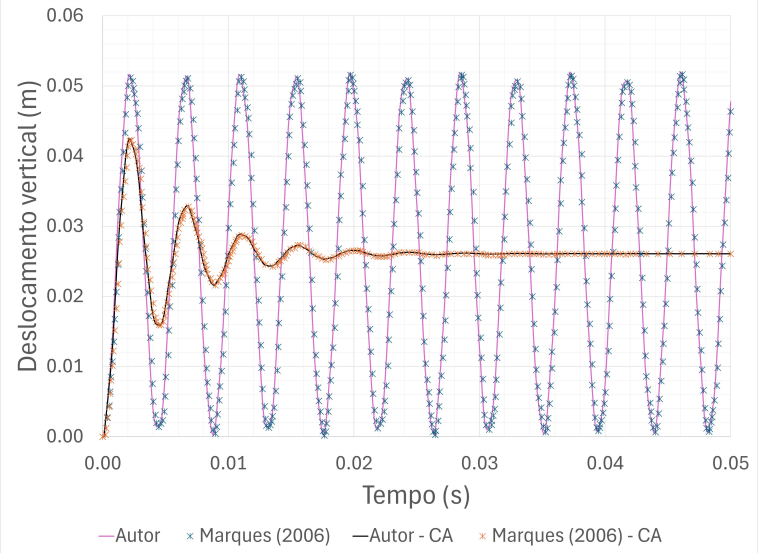
Figura 5.11 – Exemplo 3: Viga engastada-livre com amortecimento



Fonte: Próprio autor.

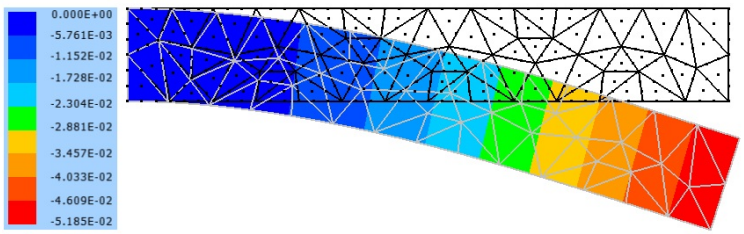
O problema foi discretizado em 82 elementos triangulares, com aproximação cúbica. A Figura 5.12 ilustra os resultados de deslocamento vertical em função do tempo, provenientes da análise transiente, comparados aos resultados obtidos por Marques (2006), tanto com a consideração de amortecimento, quanto sem essa consideração. Pode-se observar que os resultados se aproximam muito dos obtidos pela referência.

Figura 5.12 – Deslocamento vertical x tempo - Com e sem amortecimento



Fonte: Próprio autor.

Figura 5.13 – Deslocamento com t=0.0021s



Fonte: Próprio autor.

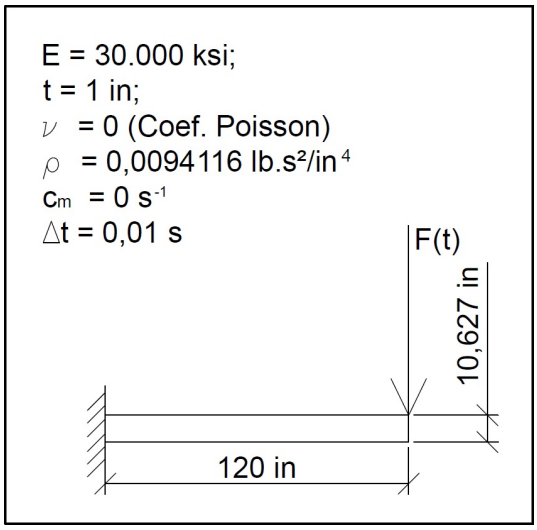
Por meio da Figura 5.13 pode ser visualizada a discretização do modelo, assim como o deslocamento da estrutura no tempo de 0.0021s, um dos períodos em que o deslocamento vertical atinge a amplitude máxima.

5.4 Exemplo 4: Viga engastada-livre com carga variável em função do tempo

Neste exemplo, avalia-se o comportamento dinâmico de uma viga engastada, com um carregamento transversal em sua extremidade livre, variando em função do tempo. Os

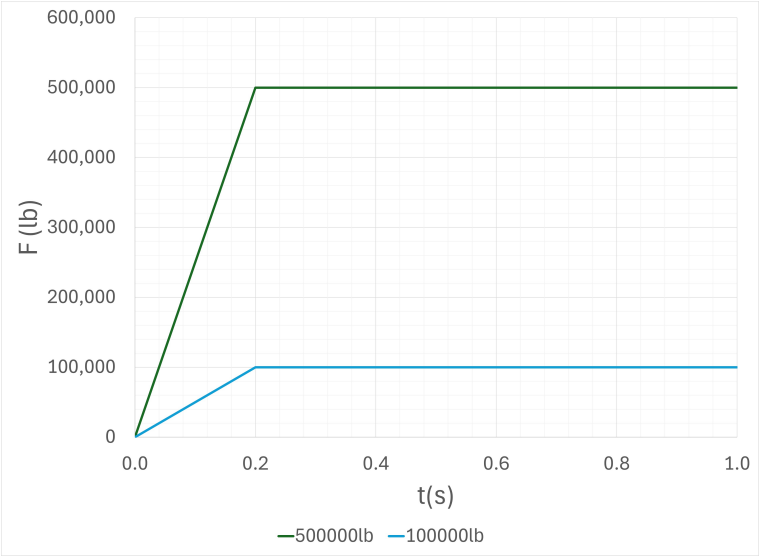
dados do problema podem ser observados por meio da Figura 5.14, e na Figura 5.15, observa-se a variação da amplitude do carregamento em função do tempo. Foram avaliadas duas situações: a primeira, com carregamento máximo de 100.000lb, e outra com carregamento máximo de 500.000lb, que foram as mesmas situações avaliadas por Marques (2006). A estrutura foi discretizada em 74 elementos triangulares, com aproximação cúbica.

Figura 5.14 – Exemplo 4: Viga engastada-livre com carga variável em função do tempo



Fonte: Próprio autor.

Figura 5.15 – Variação da força em função do tempo

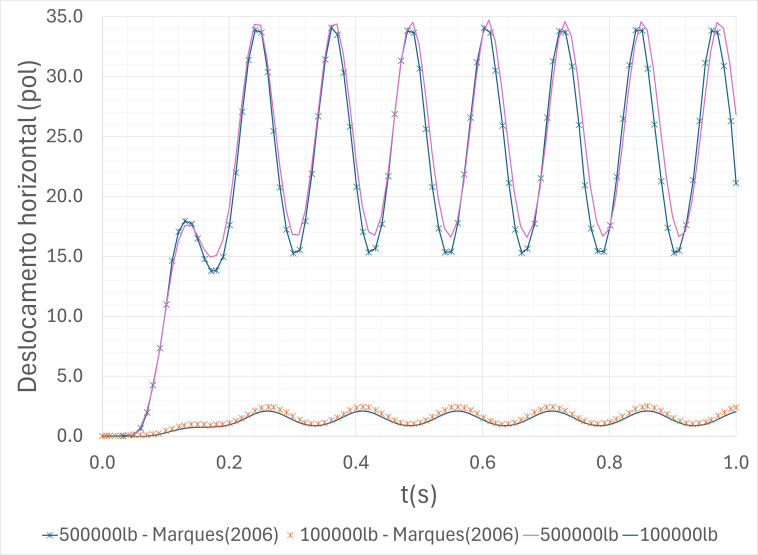


Fonte: Próprio autor.

Os resultados das análises podem ser visualizados por meio da Figura 5.16, que representa o deslocamento horizontal em função do tempo, e da Figura 5.17, que ilustra o

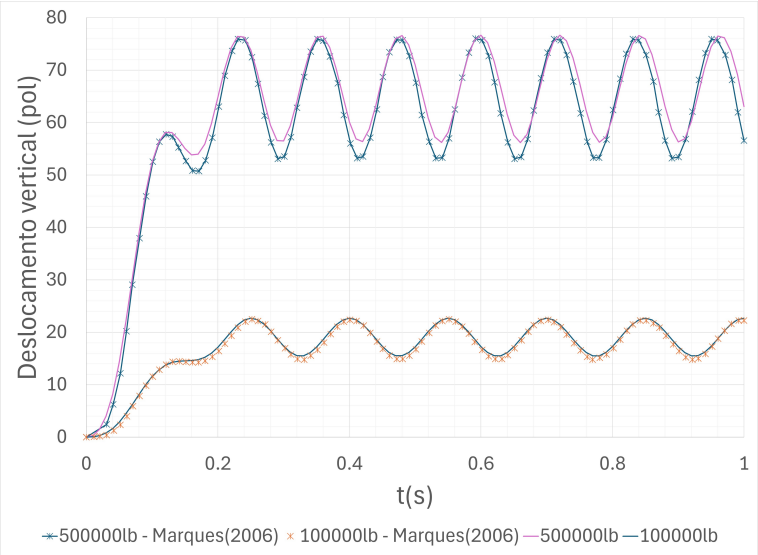
deslocamento vertical em função do tempo. Nas duas figuras há a sobreposição dos valores provenientes da análises com os resultados obtidos por Marques (2006).

Figura 5.16 – Deslocamento horizontal x tempo



Fonte: Próprio autor.

Figura 5.17 – Deslocamento vertical x tempo

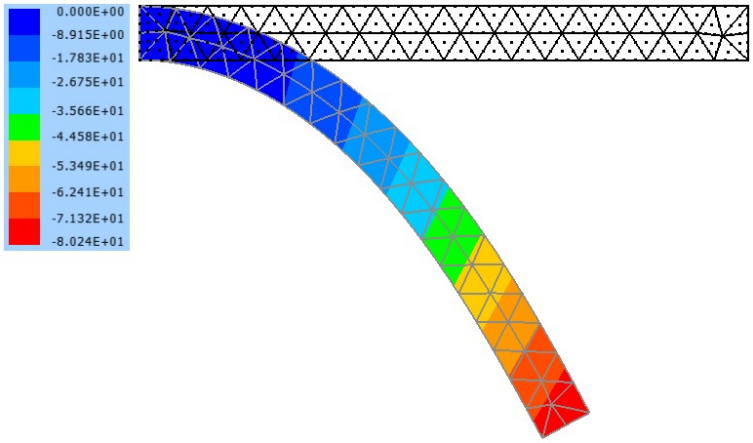


Fonte: Próprio autor.

Percebe-se a diferença entre os resultados obtidos na análise e os obtidos por Marques (2006), a medida em que se tem maiores deformações. Isso ocorre devido a utilização de diferentes medidas de deformação. Marques (2006) utilizou a deformação longitudinal de engenharia, dada por $\varepsilon = (\mid dy \mid - \mid dx \mid)/ \mid dx \mid$, enquanto que nesta análise foi utilizada a deformação de Green-Lagrange, conforme descrito no Capítulo 3.

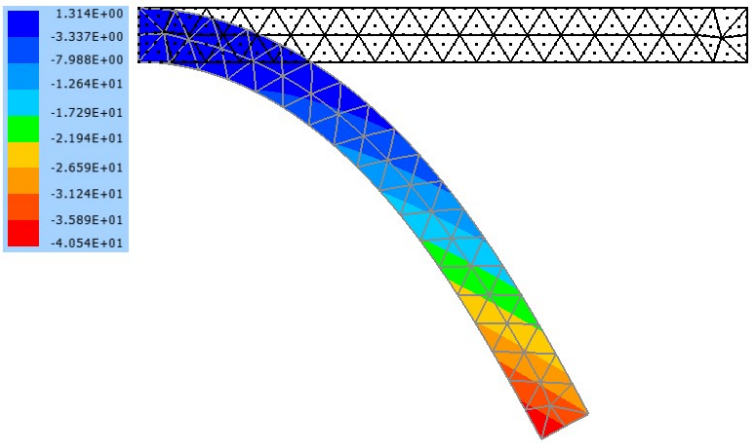
Os deslocamentos vertical e horizontal, no tempo de 0.24s, podem ser visualizados, respectivamente, pela Figura 5.18 e pela Figura 5.19, sendo esse um dos períodos em que o deslocamento atinge a amplitude máxima.

Figura 5.18 – t=0.24s - Deslocamento vertical



Fonte: Próprio autor.

Figura 5.19 – t=0.24s - Deslocamento horizontal

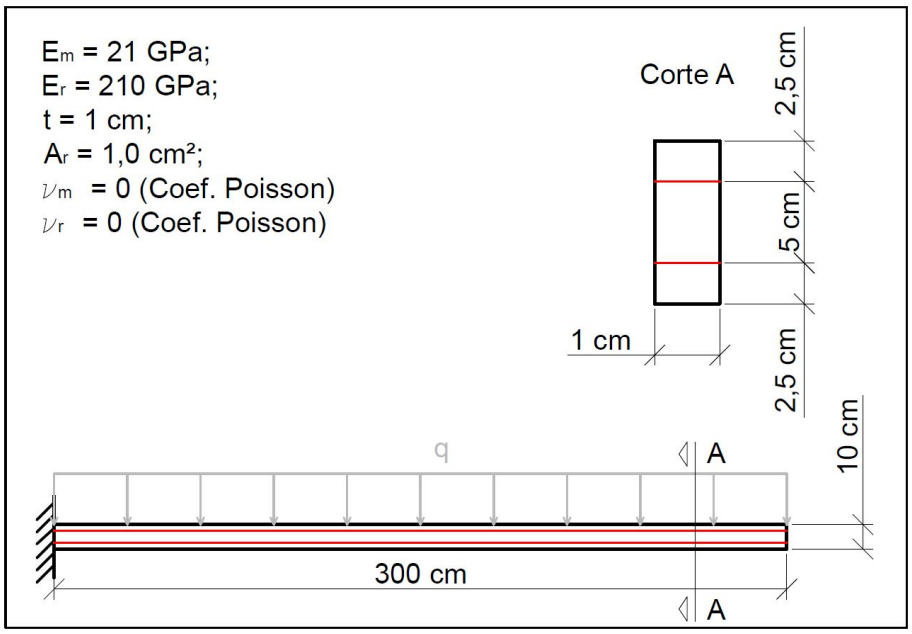


Fonte: Próprio autor.

5.5 Exemplo 5: Viga engastada-livre reforçada com carga distribuída

Este exemplo foi apresentado inicialmente por Sampaio (2014) e também por Ramos (2020), e se trata de uma viga engastada em uma de suas extremidades, e livre na outra, com um carregamento distribuído ao longo de todo o elemento. A viga é composta por dois materiais, sendo que um dos materiais compõe a matriz do elemento, e o outro compõe os reforços. A Figura 5.20 ilustra o problema, onde é possível observar a seção da viga, com a posição dos reforços.

Figura 5.20 – Exemplo 5: Viga engastada-livre reforçada com carga distribuída



Fonte: Próprio autor.

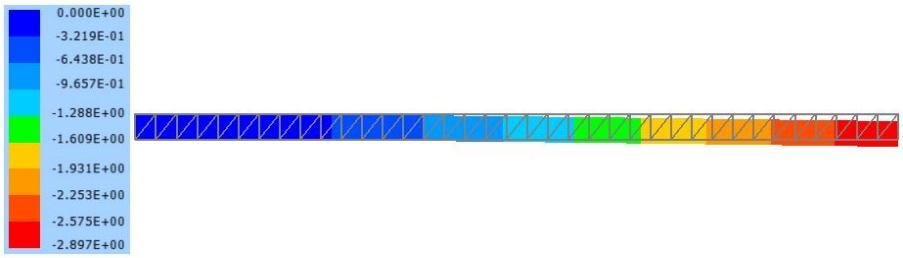
O material adotado para a matriz tem um módulo de elasticidade (E_m) de 21 GPa, enquanto o adotado para os reforços tem módulo de elasticidade (E_r) de 210 GPa. A espessura da viga (t) é de 1cm, a área de cada um dos reforços (A_r) é de 1 cm² e o coeficiente de Poisson foi adotado como 0, para ambos os materiais ($\nu_m = \nu_r = 0$).

A matriz é representada por 74 elementos triangulares, com aproximação cúbica, resultando em um total de de 448 nós. Para a simulação dos reforços, foram utilizados 224 elementos de barra simples, imersos na malha, segundo a técnica de embutimento, descrita no Capítulo 3. Foram elaborados dois modelos, sendo o primeiro somente com os elementos da matriz, e o segundo com a inserção dos reforços. Para cada um dos modelos, foram realizadas duas análises: a primeira, no regime linear, com $q \leq 0,5N/cm$; a segunda, com um nível de carregamento maior, de $q \leq 50N/cm$, já atingindo o regime não linear geométrico.

A Figura 5.21 ilustra o deslocamento vertical para o carregamento de 0,5 N/cm, no modelo sem a inserção do reforço. Já a Figura 5.22 representa o deslocamento para o mesmo carregamento, porém, com a inserção dos elementos de reforço.

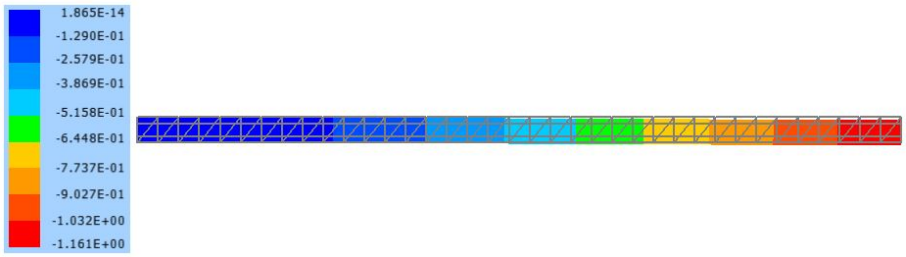
Em ambas as análises, os carregamentos foram aplicados em 10 passos de carga, considerando o valor de 0,05 N/cm para cada um deles. Os resultados obtidos na análise dos dois modelos, dentro do regime linear, foram comparados com a solução analítica do problema, por apresentar pequenos deslocamentos. Como esperado, os resultados são coincidentes, e podem ser visualizados por meio da Figura 5.23.

Figura 5.21 – Deslocamento vertical - Sem reforço - Regime linear



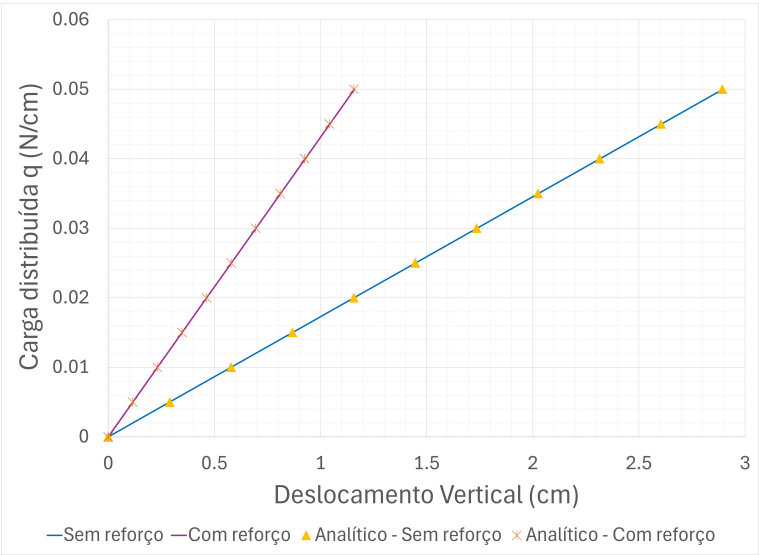
Fonte: Próprio autor.

Figura 5.22 – Deslocamento vertical - Com reforço - Regime linear



Fonte: Próprio autor.

Figura 5.23 – Comparação entre a análise e a solução analítica - Regime linear

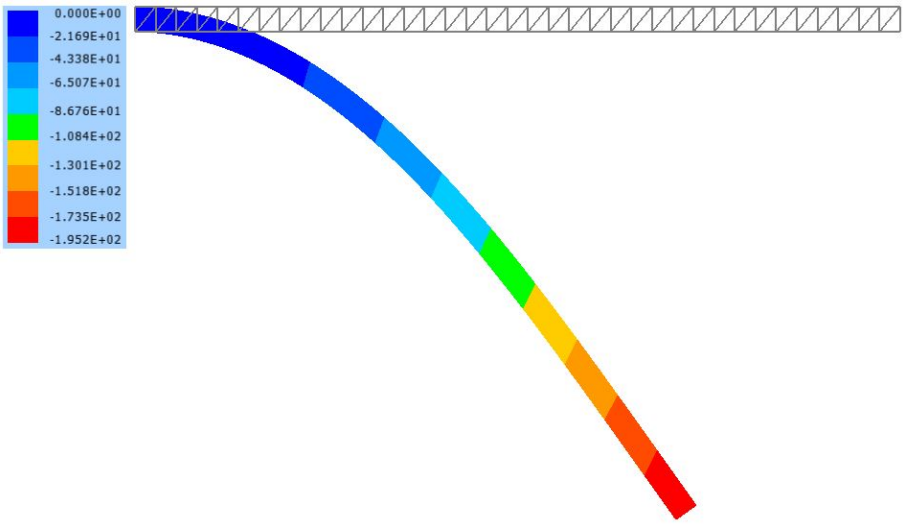


Fonte: Próprio autor.

Aumentando o nível de carregamento, o elemento passa a atingir o regime não linear geométrico. É possível observar, por meio da Figura 5.24, o deslocamento vertical para o carregamento de 50 N/cm, no modelo sem a inserção do reforço. A Figura 5.25 ilustra também, para o mesmo carregamento, o deslocamento vertical, porém, com a inserção dos elementos de reforço. Para essas análises também foram aplicados 10 passos

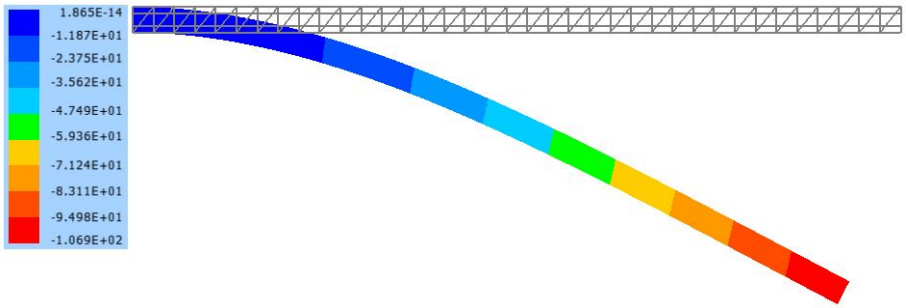
de carga, sendo cada um com o valor de 5 N/cm.

Figura 5.24 – Deslocamento vertical - Sem reforço - Regime não linear geométrico



Fonte: Próprio autor.

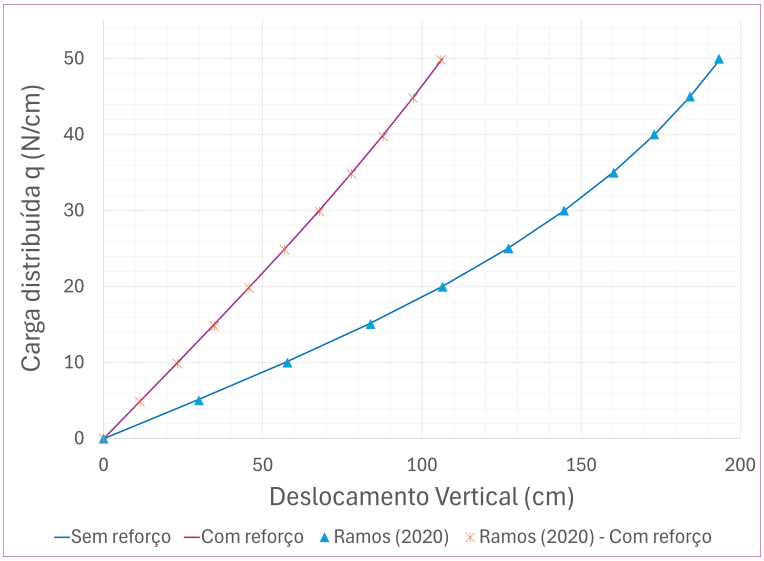
Figura 5.25 – Deslocamento vertical - Com reforço - Regime não linear geométrico



Fonte: Próprio autor.

Esses resultados, provenientes da análise no regime não linear geométrico, foram comparados aos resultados obtidos por Ramos (2020), conforme ilustra a Figura 5.26, demonstrando valores muito próximos da referência.

Figura 5.26 – Deslocamento vertical - Com reforço - Regime não linear geométrico



Fonte: Próprio autor.

6 CONCLUSÃO

Uma plataforma para análise mecânica estática e dinâmica de estruturas, com a consideração de elementos imersos na matriz, por meio do MEF posicional foi desenvolvida no paradigma de orientação a objetos.

A arquitetura foi organizada de forma generalista, com os atributos e operações de suas classes principais sendo bem abrangentes, possibilitando a implementação de novos recursos para a resolução de diferentes problemas, sem que seja necessário alterar de forma significativa a estrutura do código. Os componentes específicos utilizados em cada tipo de análise, por exemplo, foram dispostos em subclasses, herdando os atributos e operações mais genéricos das classes principais. Desta forma, em novas implementações, serão adicionadas novas classes herdeiras, sem que as principais sofram alterações significativas.

Os códigos foram desenvolvidos conforme a metodologia de orientação a objetos, usufruindo das suas principais vantagens e recursos. Os componentes que não necessitam de ajustes ou alterações, por exemplo, foram encapsulados, fazendo com que novos usuários não precisem se aprofundar a respeito do funcionamento de todas essas classes. As características de herança e polimorfismo foram amplamente utilizadas, já que existem diversos tipos de nós, elementos, materiais, seções, dentre outras categorias, que possuem muitos atributos e operações em comum. Outro recurso utilizado foi o *template*, evitando a reescrita de classes para a alteração de um único parâmetro. Destaca-se que o código não foi extensamente aprofundado em características de desempenho. O maior objetivo, no momento, era obter uma estrutura (*framework*) viável para aprimoramentos. Não obstante, todo o cálculo dos elementos é paralelo e utiliza-se solver paralelo de alto desempenho.

A documentação do código foi gerada de forma automática, com o auxílio do *software* Doxygen®, sendo que todas as informações e comentários foram inseridos diretamente no código, seguindo um mesmo padrão de entrada de dados. O repositório, nomeado como O2P2: Pre-alpha release (CARRAZEDO *et al.*, 2023), foi disponibilizado por meio do GitHub®. Desta forma, a plataforma pode ser utilizada em novas pesquisas, otimizando o tempo de desenvolvimento, além de proporcionar uma documentação muito didática e visual, facilitando o entendimento da arquitetura geral do código, assim como da implementação de cada uma de suas classes.

Todas as funcionalidades implementadas foram testadas e validadas, comparando os resultados obtidos com trabalhos de referência e/ou soluções analíticas para os mesmos problemas. Desta forma, os objetivos propostos foram atingidos, ao criar e implementar uma plataforma orientada a objetos escalonável para o Método dos Elementos Finitos Posicional, devidamente documentada e disponível para toda a comunidade científica por

meio de repositório público.

Com base nestas conclusões, visto que o método dos elementos finitos posicional já demonstrou sua eficácia em diferentes problemas de engenharia, e que a plataforma foi construída com a finalidade de auxiliar no desenvolvimento de novas pesquisas, sugerem-se as seguintes melhorias e/ou linhas de pesquisa:

a) Implementação de novos tipos de elementos, incluindo deslocamentos associados a rotação, envolvendo elementos de pórtico e casca;

b) Implementação de elementos imersos de partícula;

c) Adaptação da plataforma e implementação de elementos prismáticos de faces ativas para análise de painéis-sanduíche com núcleo em colmeia (CARRAZEDO; PACCOLA; CODA, 2018);

d) Adaptação da plataforma e implementação de novos campos de análises, como comportamento viscoelástico, térmico, termomecânico, análise de danos, etc.

REFERÊNCIAS

ANICHE, M.; YODER, J. W.; KON, F. Current challenges in practical object-oriented software design. *In*: IEEE/ACM INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: new ideas and emerging results, 41., 2019, Montreal. **Proceedings [...]**, p. 113–116. DOI: 10.1109/ICSE–NIER.2019.00037.

AVANCINI, G.; SANCHES, R. A. K. A total lagrangian position-based finite element formulation for free-surface imcompressible flows. **Finite Elements in Analysis and Design**, v. 169, 2020.

BECKER, A. A. **The boundary element method in engineering**: a complete course. London: McGraw-Hill International, 1992.

BONET, J. *et al.* Finite element analysis of air supported membrane structures. **Computer Methods in Applied Mechanics and Engineering**, v. 190, n. 5-7, p. 579–595, 2000.

BOOCH, G. *et al.* **Object-Oriented Analysis and Design with Applications**. 3. ed. [*S.l.: s.n.*]: Addison-Wesley, 2007. 720 p.

BORLAND INTERNATIONAL INC. **Turbo pascal reference manual**. 3. ed. San Jose, California: Alpha Systems Corporation, 1988. 221 p.

BOSE, A.; CAREY, G. F. A class of data structures and object-oriented implementation for finite element methods on distributed memory systems. **Computer Methods in Applied Mechanics and Engineering**, v. 171, p. 109–121, 1999.

BOWER, A. F. **Applied Mechanics of Solids**. Florida: Taylor and Francis Group, 2010.

CARRAZEDO, R. **Estudo e desenvolvimento de código computacional para análise de impacto entre estruturas levando em consideração efeitos térmicos**. 2009. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

CARRAZEDO, R.; CODA, H. B. Alternative positional FEM applied to thermomechanical impact of truss structures. **Finite Elements in Analysis and Design**, v. 46, n. 11, p. 1008–1016, 2010.

CARRAZEDO, R.; PACCOLA, R. R.; CODA, H. B. Active face prismatic positional finite element for linear and geometrically nonlinear analysis of honeycomb sandwich plates and shells. **Composite Structures**, v. 200, p. 849–863, 2018.

CARRAZEDO, R. *et al.* O2P2: Pre-alpha release (v0.2.0). Zenodo. https://doi.org/10.5281/zenodo.8283439. 2023.

CARVALHO, B. L. **Desenvolvimento de formulação alternativa em deformações finitas para sólidos viscoelásticos e fluidos viscosos pelo MEF Posicional**. 2019. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

CARVALHO, P. R. P.; CODA, H. B.; SANCHES, R. A. K. Positional finite element formulation for two-dimensional analysis of elasto-plastic solids with contact applied to cold forming processes simulation. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, 2020.

CAVALCANTE, J. P. de B.; MACIEL, D. N.; GRECO, M. Impact response of flying objects modeled by positional finite element method. **International Journal of Structural Stability and Dynamics**, v. 18, n. 6, 2018.

CODA, H. B. An exact FEM geometric non-linear analysis of frames based on position description. *In*: **International Congress of mechanical engineering**. São Carlos: ABCM, 2003.

CODA, H. B. **O Método dos Elementos Finitos Posicional : Sólidos e Estruturas – Não Linearidade Geométrica e Dinâmica**. São Carlos: EESC/USP, 2018. 248 p.

CODA, H. B.; GRECO, M. A simple fem formulation for large deflection 2d frame analysis based on position description. **Computer Methods in Applied Mechanics and Engineering**, v. 193, n. 33-35, p. 3541–3557, 2004.

CODA, H. B.; PACCOLA, R. R. **AcadView: Software para pós-processamento em elementos finitos 2D e 3D**. São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo, 2005.

CODA, H. B.; PACCOLA, R. R. An alternative positional FEM formulation for geometrically non-linear analysis of shells: Curved triangular isoparametric elements. **Computational Mechanics**, v. 40, n. 1, p. 185–200, 2007.

CODA, H. B.; PACCOLA, R. R. A positional FEM formulation for geometrical non-linear analysis of shells. **Latin American Journal of Solids and Structures**, v. 5, n. 3, p. 205–223, 2008.

COMMEND, S.; ZIMMERMANN, T. Object-oriented nonlinear finite element programming: A primer. **Advances in Engineering Software**, v. 32, n. 8, p. 611–628, 2001.

DING, J.; YU, T.; BUI, T. Q. Modeling strong/weak discontinuities by local mesh refinement variable node XFEM with object-oriented implementation. **Theoretical and Applied Fracture Mechanics**, v. 106, p. 102434, 2020.

DING, J. *et al.* An efficient variable-node XFEM for modeling multiple crack growth: A matlab object-oriented implementation. **Advances in Engineering Software**, v. 140, p. 102750, 2020.

DODDS, R. H.; LOPEZ, L. A. A generalized software system for non-linear analysis. **Advances in Engineering Software**, v. 2, n. 4, p. 161–168, 1978.

DUARTE, C. A.; BABUŠKA, I.; ODEN, J. T. Generalized finite element methods for three-dimensional structural mechanics problems. **Computers and Structures**, v. 77, n. 2, p. 215–232, 2000.

DUBOIS-PÈLERIN, Y.; ZIMMERMANN, T. Object-oriented finite element programming: III. An efficient implementation in C++. **Computer Methods in Applied Mechanics and Engineering**, v. 108, p. 165–183, 1993.

DUBOIS-PÈLERIN, Y.; ZIMMERMANN, T.; BOMME, P. Object-oriented finite element programming: II. A prototype program in Smalltalk. **Computer Methods in Applied Mechanics and Engineering**, v. 98, p. 361–397, 1992.

ÉLETRICITÉ DE FRANCE. **code-aster**. France: Életricité de France, 1989.

ERIKSSON, H.-E. *et al.* **UML 2 Toolkit**. Indianápolis, Indiana: Wiley, 2004.

FELIX, E. F. **Modelagem da Deformação do concreto armado devido à formação de produtos de corrosão**. 2018. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

FELIX, E. F. **Estudo numérico-experimental da fadiga em concretos submetidos à compressão cíclica: proposição de um modelo de dano acumulado**. 2022. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2022.

FERNANDES, J. W. D.; CODA, H. B.; SANCHES, R. A. K. ALE incompressible fluid–shell coupling based on a higher-order auxiliarymesh and positional shell finite element. **Computational Mechanics**, v. 63, p. 555–569, 2019.

FERNANDES, V. A. **Análise elastoplástica bidimensional de meios reforçados com fibras**. 2016. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

FISH, J.; BELYTSCHKO, T. **A first course in finite elements**. Chichester: John Wiley and sons, 2007.

FORDE, B. W. R.; FOSCHI, R. .; STIEME, S. F. Object-oriented finite element analysis. **Computers and Structures**, v. 34, p. 355–374, 1990.

FUJII, F. A simple mixed formulation for elastica problems. **Computers and Structures**, v. 17, p. 79–88, 1983.

GHANAM, Y.; MAURER, F.; ABRAHAMSSON, P. Making the leap to a software platform strategy: Issues and challenges. **Information and Software Technology**, v. 54, n. 9, p. 968–984, 2012. ISSN 0950-5849. Acesso em: 05/04/2023. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950584912000547.

GONG, J. *et al.* A coupled meshless-SBFEM-FEM approach in simulating soil-structure interaction with cross-scale model. **Soil Dynamics and Earthquake Engineering**, v. 136, p. 106214, 2020.

GRECO, M. **Análise de problemas de contato/impacto em estruturas de comportamento não linear pelo método dos elementos finitos**. 2004. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004.

GRECO, M.; CODA, H. B. Positional FEM formulation for flexible multi-body dynamic analysis. **Journal of Sound and Vibration**, v. 290, n. 3-5, p. 1141–1174, 2006.

GRECO, M. *et al.* Nonlinear positional formulation for space truss analysis. **Finite Elements in Analysis and Design**, v. 42, n. 12, p. 1079–1086, 2006.

HAMMER, P. C.; MARLOWE, O. J.; STROUD, A. H. Numerical integration over simplexes and cones. **Mathematical Tables and Other Aids to Computation**, X, p. 130–137, 1956.

INSANE - INTERACTIVE STRUCTURAL ANALYSIS ENVIRONMENT. Minas Gerais: UFMG, 2007.

IWAKI, T.; MAEDA, A.; ISHII, T. MISA - a general purpose FEM program. **Computers and Structures**, v. 10, p. 311–322, 1979.

JAYESH, S. W.; JEYAKARTHIKEYAN, P. V.; YOGESHWARAN, R. Object oriented programming based matlab toolbox to solve transient quasi-harmonic equation using finite element methods. *In*: **2nd International conference on Advances in Mechanical Engineering (ICAME 2018)**. Kattankulathur: IOP Publishing, 2018.

KOENIG, A.; MOO, B. E. **Accelerated C ++ Practical Programming by Example**. Upper Saddle River: Pearson, 2000. 453 p.

KOLEV, T.; DOBREV, V. **Modular Finite Element Methods (MFEM)**. Livermore, CA, USA: Lawrence Livermore National Lab. (LLNL), 2010.

KULESZA, U. **Técnicas de orientação a objetos para projeto de sistemas adaptáveis**. 2000. Dissertação (Mestrado em Ciências da Computação) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2000.

LAMPRECHT, G. **Introduction to Simula 67**. 2. ed. Braunschweig: Springer Fachmedien Wiesbaden, 1983.

LI, H.; ZHOU, Y. State of art and key problems of oop for FE programming in engineering analysis. *In*: **International Conference on Graphic and Image Processing (ICGIP 2012)**. Singapore: SPIE, 2013. v. 8768.

MACKERLE, J. Object-oriented programming in FEM and BEM: a bibliography (1990–2003). **Advances in Engineering Software**, v. 35, p. 325–336, 2004.

MACKIE, R. I. Object oriented programming of the finite element method. **International Journal for Numerical Methods in Engineering**, v. 35, p. 425–436, 1992.

MARQUES, A. N. **Modelagem numérica não-linear do concreto armado considerando a perda de aderência da armadura**. 2023. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

MARQUES, G. C. dos S. C. **Estudo e desenvolvimento de código computacional baseado no método dos elementos finitos para análise dinâmica não linear geométrica de sólidos bidimensionais**. 2006. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

MATTIASSON, K. Numerical results from large deflection beam and frame problems analysed by means of elliptic integrals. **International journal for numerical methods in engineering**, v. 17, p. 145–153, 1981.

MCKENNA, F. T. **Object-Oriented Finite Element Programming: Frameworks for Analysis, Algorithms and Parallel Computing**. 1997. Tese (PhD in Civil Engineering) — University of California, Berkeley, California, 1997.

MOURA, C. A. **Aplicação de formulação baseada no método dos elementos finitos posicional na análise bidimensional elástica de compósitos particulados**. 2015. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

NOGUEIRA, G. V.; PACCOLA, R. R.; CODA, H. B. A positional unconstrained vector layerwise (UVLWT) FEM formulation for laminated frame element modeling. **Composite Structures**, v. 148, p. 97–112, 2016.

OHTSUBO, H.; KAWAMURA, Y.; KUBOTA, A. Development of the object-oriented finite element modeling system - modify. **Engineering with Computers**, p. 187–197, 1993.

OÑATE, E. *et al.* The particle finite element method — an overview. **International Journal of Computational Methods**, v. 1, n. 2, p. 267–307, 2004.

PACCOLA, R. R.; CODA, H. B. A direct FEM approach for particulate reinforced elastic solids. **Composite Structures**, v. 141, p. 282–291, 2016.

PASCON, J. P. A large strain one-dimensional ductile damage model for space truss analysis considering gurson's porous plasticity, thermal effects and mixed hardening. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, 2022.

PATZÁK, B.; BITTNAR, Z. Design of object oriented finite element code. **Advances in Engineering Software**, v. 32, p. 759–767, 2001.

PAULINO, D. M. S.; LEONEL, E. D. Topology optimization and geometric nonlinear modeling using positional finite elements. **Optimization and Engineering**, 2021.

PIEDADE NETO, D. **On the Generalized Finite Element Method in Nonlinear Solid Mechanics Analysis**. 2013. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

PIEDADE NETO, D.; FAGA JÚNIOR, R.; PACCOLA, R. R. **AcadMesh2D**. [*S.l.: s.n.*], 2012.

RABELO, J. ao M. G. *et al.* Modeling the creep behavior of GRFP truss structures with positional finite element method. **Latin American Journal of Solids and Structures**, v. 15, n. 17, 2018.

RAMOS, E. S. **Modelagem numérica da propagação da corrosão por cloretos em estruturas de concreto armado**. 2020. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

RAMOS, E. S.; CARRAZEDO, R. Cross-section modeling of the non-uniform corrosion due to chloride ingress using the positional finite element method. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, v. 42, n. 548, 2020.

REIS, M. C. J.; CODA, H. B. Physical and geometrical non-linear analysis of plane frames considering elastoplastic semi-rigid connections by the positional FEM. **Latin American Journal of Solids and Structures**, v. 11, p. 1163–1189, 2014.

SALOMãO, R. C. **Termomecânica em compósitos reforçados com fibras e na presença de elementos particulados**. 2021. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

SAMPAIO, M. do S. M. **Análise não linear geométrica de cascas laminadas reforçadas com fibras**. 2014. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2014.

SANCHES, R. A.; CODA, H. B. Unconstrained vector nonlinear dynamic shell formulation applied to fluid structure interaction. **Computer Methods in Applied Mechanics and Engineering**, v. 259, p. 177–196, 2013.

SANCHES, R. A.; CODA, H. B. On fluid-shell coupling using an arbitrary lagrangian-eulerian fluid solver coupled to a positional lagrangian shell solver. **Applied Mathematical Modelling**, v. 38, n. 14, p. 3401–3418, 2014.

SANNER, T. A.; NIELSEN, P. Software platforms for inclusive innovation". *In*: NIELSEN, P.; KIMARO, H. C. (ed.). **Information and Communication Technologies for Development. Strengthening Southern-Driven Cooperation as a Catalyst for ICT4D**. Cham: Springer International Publishing, 2019. p. 218–230. ISBN 978-3-030-18400-1.

SCHOLZ, S. P. Elements of an object-oriented FEM++ program in C++. **Computers and Structures**, v. 43, n. 3, p. 517–529, 1992.

SEED, G. M. **An introduction to object-oriented programming in C++**. London: Springer, 1996. 1048 p.

SENGUPTA, S.; KOROBKIN, C. P. **C++ Object-oriented data structures**. New York: Springer, 1994. 728 p.

SIMO, J. C.; HJELMSTAD, K. D.; TAYLOR, R. L. Numerical formulations of elasto-vistoplastic response of beams accounting for the effect of shear. **Computer methods in applied mechanics and engineering**, v. 42, p. 301–330, 1984.

SIQUEIRA, T. M. **Ligações deslizantes para análise dinâmica não linear geométrica de estruturas e mecanismos tridimensionais pelo método dos elementos finitos posicional**. 2019. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

SMALLTALK/V 286. Tutorial and programming handbook. Los Angeles, DIGITALK, 1998. 561 p.

SOARES, H. B.; PACCOLA, R. R.; CODA, H. B. Unconstrained vector positional shell FEM formulation applied to thinwalled members instability analysis. **Thin-walled Structures**, v. 136, p. 246–257, 2019.

SOARES, H. B.; PACCOLA, R. R.; CODA, H. B. A conjugate modal force strategy for instability analysis of thin-walled structures: an unconstrained vector positional finite element approach. **Latin American Journal of Solids and Structures**, v. 18, n. 344, 2021.

TAN, B. *et al.* openVFIFE: An object-oriented structure analysis platform based on vector form intrinsic finite element method. **Buildings**, v. 11, 2021.

TAVARES, M. de G. **Simulação da perda de protensão aderente em elementos de concreto**. 2020. Dissertação (Mestrado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

VANALLI, L. **O MEC e o MEF aplicados à análise de problemas viscoplásticos em meios anisotrópicos e compostos**. 2004. Tese (Doutorado em Engenharia de Estruturas) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004.

WILLS, C. M. R.; ROE, D. NASTRAN - a finite element problem for structural analysis. **Computer-Aided Design**, p. 172–175, 1972.

WRIGGERS, P. **Nonlinear finite element methods**. Hannover: Springer, 2008.

YAMASSAKI, R. T. **Um programa de elementos finitos em GPU e orientado a objetos para análise dinâmica não linear de estruturas.** 2014. Tese (Doutorado em Engenharia Mecânica de Projeto e Fabricação) — Escola Politécnica, Universidade de São Paulo, São Paulo, 2014.

ZEGLINSKI, G. W.; HAN, R. P. S.; AITCHISON, P. Object oriented matrix classes for use in a finite element code using C++. **International Journal for Numerical Methods in Engineering**, v. 37, p. 3921–3937, 1994.

ZIENKIEWICZ, O.; HUANG, G.; LIU, Y. Adaptive FEM computation of forming processes-application to porous and non-porous materials. **International Journal for numerical methods in engineering**, v. 30, p. 1527–1553, 1990.

ZIMMERMANN, T.; DUBOIS-PÈLERIN, Y.; BOMME, P. Object-oriented finite element programming: I. Governing principles. **Computer Methods in Applied Mechanics and Engineering**, v. 98, p. 291–303, 1992.


PPG
SET


EESC • USP