UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

Heitor Lins Falcão

Desenvolvimento de ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado

Recife
2024

Heitor Lins Falcão

Desenvolvimento de ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado

Trabalho de Conclusão de Curso da Graduação em Engenharia Mecânica do Centro de Tecnologia e Geociências da Universidade Federal de Pernambuco como requisito parcial para a obtenção do título de Engenheiro Mecânico.
Orientador: Prof. Ramiro Brito Willmersdorf, Dr.

Recife
2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Falcão, Heitor Lins.
Desenvolvimento de ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado / Heitor Lins Falcão. - Recife, 2024.
91 p. : il., tab.

Orientador(a): Ramiro Brito Willmersdorf
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Mecânica - Bacharelado, 2024.
Inclui referências, apêndices.

1. MEF. 2. Dutos. 3. Corrosão. 4. Modelagem automatizada. 5. Livre acesso. 6. Malha. I. Willmersdorf, Ramiro Brito. (Orientação). II. Título.

620 CDD (22.ed.)

Heitor Lins Falcão

Desenvolvimento de ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro Mecânico" e aprovado em sua forma final pelo Curso de Graduação em Engenharia Mecânica.

Recife, 22 de março de 2024.

Banca Examinadora:

Prof. Dr. Ramiro Brito Willmersdorf
(Orientador)
Universidade Federal de Pernambuco

Prof. Dr. Adriano Dayvson Marques Ferreira
(Examinador interno)
Universidade Federal de Pernambuco

Prof. Msc. Adson Bezerra da Silva
(Examinador interno)
Universidade Federal de Pernambuco

AGRADECIMENTOS

À minha família, que sempre esteve comigo em todos os momentos, bons ou ruins, me auxiliando em tudo que precisei e que fazem parte da pessoa que eu sou hoje. Em especial à minha mãe Klécia, irmã Rebecca, pai Pedro, e avós Neusa, Floriano e Maria Célia, que acompanharam minha trajetória desde o início e me apoiaram para que eu estivesse onde estou. Um agradecimento especial também ao meu tio Evaldo e minha tia Maria José que me forneceram um suporte fundamental no começo da graduação.

Um grande agradecimento aos meus amigos de graduação, que estiveram próximos durante essa caminhada, compartilharam diversos momentos comigo e fizeram o tempo na graduação mais leve e prazeroso, em especial, Elder, Elton, Lucas e Rebeca que me acompanham e me dão apoio desde o segundo semestre de faculdade. Aos meus amigos anteriores à graduação João Victor, João Vitor e Manuela, que são companheiros de vida e, mesmo em caminhos diferentes, sempre fizeram questão de continuar presentes. À minha namorada, Thayná, pelo companheirismo de anos, pelo carinho e pelo apoio tão importantes.

Agradeço imensamente ao meu orientador, professor Ramiro Brito Willmersdorf, pela proposta desse grande desafio, pela paciência e pelo suporte durante o desenvolvimento do trabalho. Também aos professores Adson Bezerra e Adriano Dayvson pela disponibilidade para fazer parte da banca examinadora e pelos comentários construtivos importantíssimos. E aos professores e colegas que acrescentaram de alguma forma para minha caminhada.

Todas essas pessoas foram fundamentais para minha formação profissional e pessoal. Por isso, e muito mais, sou imensamente grato a todos.

RESUMO

Considerando a importância do petróleo e gás natural no setor energético brasileiro e mundial, o bom funcionamento das linhas de dutos que os transportam e distribuem é vital e acidentes nessas tubulações podem causar consequências em diversas esferas da sociedade. Uma das principais fontes de problemas para a integridade estrutural dos dutos é a corrosão, podendo chegar a causar falhas em linhas com monitoramento insuficiente. Para a avaliação, os métodos mais comumente utilizados são os semi-empíricos baseados em códigos e normas, que geram resultados rápidos e confiáveis, embora resultem, normalmente, em valores conservadores. Assim, o método dos elementos finitos se mostra uma boa opção e uma poderosa ferramenta para a avaliação de dutos com defeitos de corrosão por conseguirem representar suficientemente bem diversas condições de geometria e carregamentos. Entretanto, seu uso requer um conhecimento específico, que não é comum a todos os engenheiros de tubulação, e os *softwares* que realizam as análises muitas vezes possuem licenças de uso caras. Com isso, desenvolveu-se ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado utilizando recursos gratuitos de forma a atuar na diminuição dessas barreiras. A principal ferramenta utiliza o *software* SALOME para a criação da geometria e sua malha de forma paramétrica e para a geração de um código que permite a automatização dessas etapas. Para entender os limites desse código, diversos testes para diferentes geometria foram feitos e, para comprovar a qualidade dos modelos gerados, modelos de trabalhos relevantes foram gerados a partir da ferramenta desenvolvida e analisados no ANSYS. Além disso, outro código foi desenvolvido para converter o arquivo exportado pelo SALOME para um formato que o ANSYS consegue importar. Com isso, na seção de metodologia, são descritos os passos, parâmetros e requisitos para o desenvolvimento do código de geração dos modelos, dos testes realizados, do código de conversão e das análises estruturais. Em resultados e discussões, os códigos desenvolvidos são apresentados, os principais resultados dos testes são mostrados, juntamente com uma discussão das aplicações e limitações do código de geração dos modelos, e os resultados das análises são comparados com os trabalhos utilizados como referência. Assim, o objetivo principal de desenvolver ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado utilizando recursos gratuitos foi cumprido, auxiliando, com isso, na diminuição da barreira de conhecimento necessário para realizar análises pelo MEF, reduzindo o tempo gasto nas etapas de modelagem geométrica e geração da malha e dando um primeiro passo na diminuição da barreira dos custos dos *softwares*.

Palavras-chave: MEF; Dutos; Corrosão; Modelagem automatizada; Livre acesso; Malha.

ABSTRACT

Considering the importance of oil and natural gas in the Brazilian and global energy sector, the proper operation of the pipelines that transport and distribute them is vital and accidents on these pipelines can cause consequences in several spheres of society. One of the main sources of problems for the structural integrity of pipelines is corrosion, which may cause failures in lines with insufficient monitoring. For defect assesment, the most commonly used methods are the semi-empirical ones based on codes and standards, which generate fast and reliable results, although they usually result in conservative values. Thus, the finite element method proves to be a good option and a powerful tool for corrosion defects assesment by being able to represent sufficiently well differents geometries and load conditions. However, its use requires specific knowledge, which is not common to all pipeline engineers, and the softwares that perform the analyzes often have expensive licenses. As a result, some tools were developed to generate computational models of pipelines with an idealized corrosion defect automatically using free resources in order to reduce these barriers. The main tool uses the software SALOME to create the geometry and its mesh in a parametric way and to generate a code that allows the automation of these steps. To understand the limits of this script, several tests for different geometries were carried out and, to prove the quality of the generated models, models from relevant works were generated by the tool and analyzed in ANSYS. Additionally, other script was developed to convert the file exported by SALOME to a format that ANSYS can import. Therefore, in the methodology section, the steps, parameters and requirements for the development of the model generator script, the tests carried out, the conversion script and the structural analyzes are described. In results and discussions, the developed scripts are presented, the main results from the tests are shown, together with a discussion of the applications and limitations of the model generator code, and the analysis results are compared with the works used as reference. Thus, the main objective of developing tools for the generation of computational models of pipelines with idealized corrosion defect automatically using free resources was fulfilled, hence helping to reduce the knowledge barrier necessary to carry out analyzes by the FEM, reducing the time spent in the geometric modeling and mesh generation stages and taking a first step in reducing the software cost barrier.

Keywords: FEM; Pipe; Corrosion; Automatic modeling; Open-source; Mesh.

LISTA DE FIGURAS

Figura 1 – Corrosão uniforme causada pela atmosfera 18
Figura 2 – Corrosão por pite 18
Figura 3 – Ilustração de defeitos mecânicos e de solda 19
Figura 4 – Níveis de avaliação de defeitos 20
Figura 5 – Barra de comprimento L sob força axial P 22
Figura 6 – Barra de comprimento 2L sob força axial P 22
Figura 7 – Tensão por deformação para aço de baixo carbono 23
Figura 8 – Tensão verdadeira por deformação verdadeira para um material dúctil típico 23
Figura 9 – Corpo tridimensional arbitrário 24
Figura 10 – Volume elementar 25
Figura 11 – Modelo matemático de experimento de tensão uniaxial 26
Figura 12 – Modelo contínuo de uma barra 27
Figura 13 – Modelo aproximado discreto da barra 27
Figura 14 – Exemplo de versatilidade dos elementos triangulares 29
Figura 15 – Placa discretizada e elemento triangular 30
Figura 16 – Placa discretizada e elemento retangular 31
Figura 17 – Elemento tetraédrico isolado 33
Figura 18 – Elemento hexaédrico isolado 34
Figura 19 – Fluxograma da pesquisa parte 1 38
Figura 20 – Fluxograma da pesquisa parte 2 39
Figura 21 – Parâmetros geométricos no SALOME 40
Figura 22 – Geometria da RD 41
Figura 23 – Geometria da RT1 42
Figura 24 – Geometria da RT2 42
Figura 25 – Geometria da RT3 43
Figura 26 – Planos utilizados para eliminar as regiões sobrepostas 44
Figura 27 – Geometria final 44
Figura 28 – Subdivisões em cada RT 45
Figura 29 – Discretização das regiões próximas ao raio RA 45
Figura 30 – Arestas com sub-malhas 46
Figura 31 – Malha final da RD 46
Figura 32 – Malha final da RT1 47
Figura 33 – Conexão entre a RD e a RT1 47
Figura 34 – Malha final da RT2 48
Figura 35 – Conexão entre a RT1 e a RT2 48
Figura 36 – Malha final da RT3 49

Figura 37 – Conexão entre a RT2 e a RT3 49
Figura 38 – Malha final para defeitos externos 50
Figura 39 – Malha final para defeitos internos 50
Figura 40 – Formato das informações no arquivo .dat 52
Figura 41 – Formato das informações no arquivo .inp 53
Figura 42 – Formato das informações dos elementos piramidais 53
Figura 43 – Curva de tensão verdadeira por deformação plástica para o X80 55
Figura 44 – Curva de tensão verdadeira por deformação plástica para o X60 55
Figura 45 – Modelo 1 - MD1 56
Figura 46 – Modelo 2 - MD2 57
Figura 47 – Malha do MD1 57
Figura 48 – Malha do MD2 58
Figura 49 – Estatísticas da malha do MD1 58
Figura 50 – Estatísticas da malha do MD2 58
Figura 51 – Condições de contorno do MD1 59
Figura 52 – Condições de contorno do MD2 60
Figura 53 – Início do código e definição do parâmetro RCC 61
Figura 54 – Definição dos parâmetros 62
Figura 55 – Definições do primeiro esboço e suas geometrias 62
Figura 56 – Finalização da definição das geometrias do esboço 1 e extrusão do defeito 63
Figura 57 – Continuação do comando de extrusão 63
Figura 58 – Criação dos raios de adoçamento do defeito RC e RA 63
Figura 59 – Continuação do comando dos raios de adoçamento 63
Figura 60 – Criação dos planos que recortam as RT's 64
Figura 61 – Comando de *split* para dividir as geometrias 64
Figura 62 – Continuação dos comandos de divisão 64
Figura 63 – Remoção das geometrias desnecessárias 64
Figura 64 – Definição dos grupos 65
Figura 65 – Último grupo e fechamento do módulo da geometria 65
Figura 66 – Estudo do SHAPER 65
Figura 67 – Continuação da definição do estudo do SHAPER 65
Figura 68 – Início do módulo MESH e da definição da malha da RD 66
Figura 69 – Finalização da malha da RD 66
Figura 70 – Junção das malhas numa única malha e comandos para as malhas 1D . 67
Figura 71 – Comandos de nomenclatura 67
Figura 72 – Comando para exportar a malha e finalização do código 67
Figura 73 – Linhas do código para espelhar as malhas 67
Figura 74 – Mudança no arquivo e obtenção do número de nós e de elementos . . . 68
Figura 75 – Criação do *array* dos nós 68

Figura 76 – Criação do *array* dos elementos 68
Figura 77 – Filtragem e criação dos *arrays* dos elementos tridimensionais 69
Figura 78 – Numeração dos elementos e escrita do arquivo final 69
Figura 79 – Função para leitura e armazenamento da primeira linha do arquivo .dat 70
Figura 80 – Função para leitura e armazenamento de “x” linhas do arquivo .dat . . 70
Figura 81 – Função para leitura e armazenamento de “x” linhas do arquivo .dat começando de uma linha específica 70
Figura 82 – Função para escrita do arquivo .inp 71
Figura 83 – Face da RD da amostra 3 71
Figura 84 – Malha excessivamente refinada na RD da amostra 4 71
Figura 85 – Transição brusca na amostra 8 72
Figura 86 – Regiões de atenção nas malhas da amostra 19 72
Figura 87 – Transição brusca na região da espessura remanescente 72
Figura 88 – Malha das regiões de transição 73
Figura 89 – Tensão e deformação do MD1 74
Figura 90 – Resultados obtidos no trabalho base 1 74
Figura 91 – Falha do IDTS 2 75
Figura 92 – Falha com defeito único 75
Figura 93 – Tensão e deformação do MD2 76
Figura 94 – Distribuição de tensões obtidas no trabalho base 2 76
Figura 95 – Malha da RD da amostra 1 85
Figura 96 – Malha da RD da amostra 2 85
Figura 97 – Malha da RD da amostra 3 85
Figura 98 – Malha da RD da amostra 4 86
Figura 99 – Malha da RD da amostra 5 86
Figura 100 – Malha da RD da amostra 6 86
Figura 101 – Malha da RD da amostra 7 87
Figura 102 – Malha da RD da amostra 8 87
Figura 103 – Malha da RD da amostra 9 87
Figura 104 – Malha da RD da amostra 10 88
Figura 105 – Malha da RD da amostra 11 88
Figura 106 – Malha da RD da amostra 12 88
Figura 107 – Malha da RD da amostra 13 89
Figura 108 – Malha da RD da amostra 14 89
Figura 109 – Malha da RD da amostra 15 89
Figura 110 – Malha da RD da amostra 16 90
Figura 111 – Malha da RD da amostra 17 90
Figura 112 – Malha da RD da amostra 18 90
Figura 113 – Malha da RD da amostra 19 91

Figura 114 – Malha da RD da amostra 20 91
Figura 115 – Malha da RD da amostra 21 91

LISTA DE TABELAS

Tabela 1 – Parâmetros geométricos 40
Tabela 2 – Parâmetros geométricos das amostras 51
Tabela 3 – Propriedades mecânicas dos materiais utilizados 55
Tabela 4 – Parâmetros geométricos dos modelos 56

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS 16
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 INTEGRIDADE DE DUTOS 17
2.1.1 Definição e tipos de defeitos 17
2.1.2 Níveis de avaliação dos defeitos 20
2.2 MODELAGEM DA GEOMETRIA 20
2.3 NOÇÕES BÁSICAS DE MECÂNICA DOS SÓLIDOS 21
2.3.1 Gráfico tensão x deformação 22
2.3.2 Tensões, equilíbrio e lei de Hooke generalizada 23
2.3.3 Elasticidade vs. Plasticidade 26
2.4 MÉTODO DOS ELEMENTOS FINITOS 26
2.4.1 Matriz de rigidez 28
2.4.2 Tipos de elementos 29
2.4.2.1 Elemento triangular sob estado plano de tensão 29
2.4.2.2 Elemento retangular sob estado plano de tensão 31
2.4.2.3 Elemento tetraédrico 32
2.4.2.4 Elemento hexaédrico 34
2.4.2.5 Elementos de ordem maior 35
2.4.3 Noções básicas para análise não linear 35
3 METODOLOGIA 37
3.1 FERRAMENTAS 37
3.2 DESENVOLVIMENTO DO PROJETO 38
3.2.1 Fluxo de trabalho 38
3.2.2 Geometria 40
3.2.3 Malha 44
3.2.4 Teste da geração de malha para diferentes geometrias 51
3.2.5 Conversor do arquivo da malha 52
3.2.6 Análise estrutural 54
4 RESULTADOS E DISCUSSÕES 61
4.1 *SCRIPT* DE GERAÇÃO DO MODELO COMPUTACIONAL 61
4.2 *SCRIPT* PARA CONVERSÃO DA MALHA EM ARQUIVO .DAT PARA .INP 68
4.3 TESTES DE MODELOS VARIADOS 71
4.4 MODELOS ANALISADOS 73
5 CONCLUSÃO 77
5.1 TRABALHOS FUTUROS 78

REFERÊNCIAS **79**
APÊNDICE A – Resultados dos testes do código de geração dos modelos **85**

1 INTRODUÇÃO

Em 2022, o petróleo e o gás natural representaram 46,2% da matriz energética brasileira e, em 2021, 53,1% da mundial (EPE, 2024), destacando a importância desses recursos na infraestrutura energética do país e globalmente. Para garantir o transporte e distribuição desses combustíveis de forma segura e confiável, um dos principais meios são as extensas redes de tubulações. No Brasil, a Transpetro sozinha detém cerca de 8,5 mil quilômetros de dutos (TRANSPETRO, 2024), enquanto nos Estados Unidos, a rede de dutos de gás natural totaliza 3 milhões de milhas, equivalentes a 4,8 milhões de quilômetros (U.S. EIA, 2024). Mesmo com o intuito de reduzir o uso desses combustíveis em meio à transição energética, é provável que essas tubulações continuem a desempenhar um papel crucial, adaptando-se somente para transportar novos produtos, como o hidrogênio, por exemplo, à medida que as necessidades energéticas evoluírem.

Esse contexto evidencia a importância do bom funcionamento dessas linhas, com a falha delas gerando impactos em diversas esferas, como elevados custos com paralisação de processos e substituição de linhas ou equipamentos e, principalmente, aos colaboradores e a população afetados pelo acidente. Uma das principais causas de problemas para a integridade de tubulações é a corrosão, podendo causar acidentes em linhas com manutenção e monitoramento insuficientes (BRUÈRE; BOUCHONNEAU; MOTTA, 2019). Em geral, defeitos de corrosão são avaliados por métodos semi-empíricos baseados em códigos e normas, como o ASME B31G (ASME, 1991), o DNV RP-F101 (DNV, 2017) e o RSTRENG *Effective Area Method* (KIEFNER; VIETH, 1989), que geram resultados rápidos e confiáveis, embora resultem, normalmente, em valores conservadores, podendo acarretar em paradas ou reparos prematuros nas linhas (SOARES *et al.*, 2019). Assim, trabalhos como Soares *et al.* (2019), J. Sun e Y. F. Cheng (2018), Tee e Wordu (2020), Chen *et al.* (2015), Han, H. Zhang e J. Zhang (2016), Xu e Y.F. Cheng (2012) e Liu, Khan e Thodi (2017) demonstram a versatilidade do uso do método dos elementos finitos (MEF) ao analisarem a falha de dutos com defeitos, únicos ou múltiplos, sob apenas pressão interna ou combinações de carregamento, obtendo resultados relevantes quando comparados com dados experimentais, ao mesmo tempo que a física do problema é melhor representada em comparação com os métodos semi-empíricos.

Apesar das vantagens, poucos engenheiros de tubulações possuem o conhecimento para executar as simulações. As etapas de modelagem geométrica, discretização e configuração da simulação além de precisarem do conhecimento específico, muitas vezes são etapas iterativas manuais por parte do engenheiro. Nesse contexto, o grupo de pesquisa de Processamento de Alto Desempenho em Mecânica Computacional (PADMEC) da UFPE desenvolveu o PIPEFLAW que é uma ferramenta para a modelagem e discretização de dutos com defeitos de corrosão e o PIPEFLAW_A que é um *script* para automatizar a análise de pressão de falha no ANSYS (ANSYS, 2024). Os trabalhos Cabral (2007), Cabral

et al. (2007), Motta *et al.* (2010), Ferreira *et al.* (2010), Motta *et al.* (2017), Ferreira *et al.* (2016), Motta *et al.* (2009), Cabral *et al.* (2017), Ferreira *et al.* (2021, 2009), Pimentel *et al.* (2020) e Bruère, Bouchonneau e Motta (2019) mostram o desenvolvimento da ferramenta, assim como sua aplicação para defeitos, únicos e múltiplos, idealizados e a evolução para defeitos reais. Mesmo assim, o uso do MSC.PATRAN (HEXAGON, 2024) para desenvolver a ferramenta ainda constitui uma barreira para disseminar o uso e para o desenvolvimento de ferramentas auxiliares, pois é um *software* que requer a compra de uma licença.

Assim, o presente trabalho propõe o uso de recursos livres para o desenvolvimento de ferramentas para automatizar as etapas de modelagem da geometria e discretização dela. Com isso, diminui-se a barreira da geração de modelos para análise numérica, mesmo que a análise em si não seja afetada, também facilita a disseminação e evolução da própria ferramenta.

1.1 OBJETIVOS

O objetivo geral deste trabalho é desenvolver ferramentas para a geração paramétrica de um modelo computacional de dutos com defeito de corrosão idealizado com formato de "caixa" utilizando recursos gratuitos.

Como objetivos específicos, tem-se:

- Modelar a geometria do defeito parametrizando suas dimensões;
- Modelar o duto separando as regiões de transição de malha;
- Testar configurações de malha nas diferentes regiões do duto;
- Gerar e adaptar, a partir da malha construída, um *script* que permite que a mesma seja gerada utilizando-o no *software* CAD;
- Modificar os parâmetros de geometria para gerar diferentes configurações de dutos com defeitos a partir do *script*;
- Gerar modelos de trabalhos da área por meio da ferramenta desenvolvida para executar análises estruturais pelo MEF;
- Comparar os resultados obtidos com os trabalhos utilizados como referência.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTEGRIDADE DE DUTOS

Um pré-requisito para a operação segura de dutos é a garantia com uma alta confiabilidade da integridade durante sua vida útil. Integridade esta que pode ser comprometida por diversos fatores, dentre eles defeitos introduzidos na tubulação durante a fabricação, instalação ou operação (MMS, 2000).

Como não é viável prevenir todos os defeitos e nem todos são prejudiciais para a integridade da linha, então é importante poder distinguir aqueles toleráveis dos danosos (MMS, 2000). Normas e códigos como ASME B31.G são utilizados para a avaliação de defeitos, assim como as simulações computacionais e experimentos laboratoriais.

2.1.1 Definição e tipos de defeitos

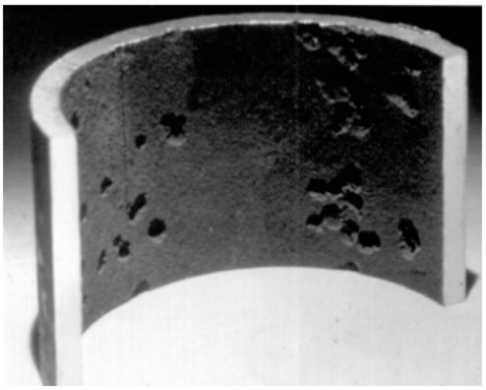
Defeito é uma descontinuidade, geométrica ou do material, ou uma irregularidade detectada por inspeção de acordo com requisitos de normas e códigos aplicáveis, sendo danoso quando é de magnitude suficiente para que seja rejeitado de acordo com os requisitos dos códigos, normas ou outros métodos utilizados para a avaliação desse defeito. Pode-se agrupá-los em três categorias de acordo com sua causa, defeitos de corrosão, mecânicos e de solda (MMS, 2000):

- Defeitos de corrosão: podem ser divididos de acordo com sua natureza, como visto abaixo. Alguns exemplos estão ilustrados nas Figuras 1 e 2.
  - Corrosão uniforme: Perda de espessura de parede do duto de forma uniforme ou gradual numa área extensa;
  - Corrosão por pite: Perda de espessura localizada, podendo reduzir a espessura para um valor menor que o de projeto;
  - Corrosão sob tensão em meio corrosivo: Acontece quando o material sob tensão trativa, aplicada ou residual, está exposto em meio corrosivo específico;
  - Trincamento sob tensão por hidrogênio: Ocorre quando o Hidrogênio migra para o interior de um material, acumula-se em falhas existentes e causa trincas internas sob baixas tensões.

Figura 1 – Corrosão uniforme causada pela atmosfera

Fonte: API (2011).

Figura 2 – Corrosão por pite

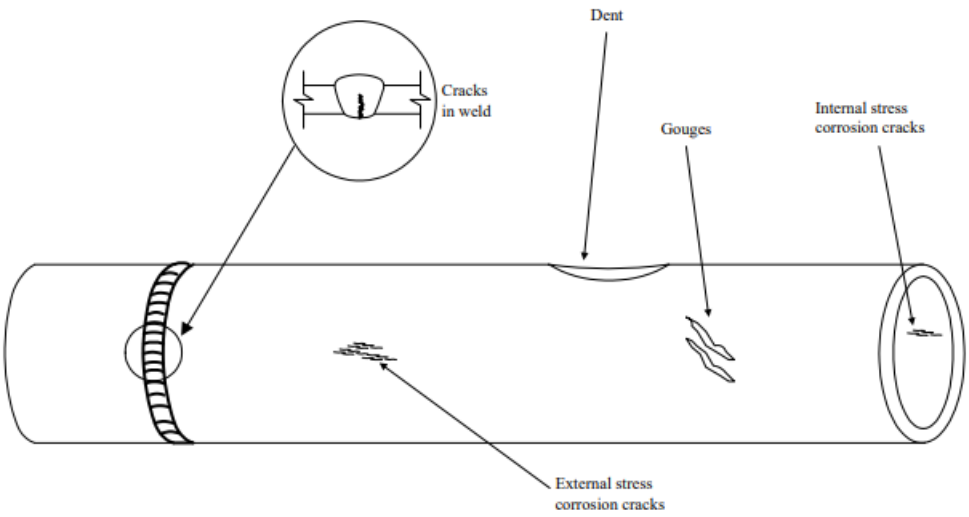


Fonte: API (2011).

- Defeitos mecânicos: A Figura 3 ilustra alguns tipos de defeitos mecânicos. Os tipos de defeitos mecânicos estão listados abaixo.
  - Mossa ou amassamento (“Dent”): Causada quando produz-se uma variação visível na curvatura da parede do duto, ou componente, sem variação na espessura da parede;
  - Rasgos superficiais (“Gouge”): Imperfeição na superfície causada pela remoção ou deslocamento de material, provocando perda de espessura da parede;
  - Ranhuras (“Grooves”): Uma ranhura pode causar concentração de tensões em um determinado ponto, podendo, assim, ser considerada um defeito;
  - Trincas superficiais: Trincas geradas na superfície do duto.

- Defeitos de solda: Na Figura 3, é possível observar uma forma de defeito de solda. Abaixo temos os tipos de defeitos de solda.
  - Queimadura de arco ("Arc burn"): Uma condição ou depósito localizado causado por um arco elétrico e consiste de metal refundido, metal termicamente afetado, uma mudança no perfil da superfície ou uma combinação desses efeitos;
  - Penetração incompleta: Quando a raiz da junta soldada não é fundida e preenchida completamente;
  - Fusão incompleta: É a ausência da união por fusão entre passes adjacentes da solda e o metal de base;
  - Concavidade interna: Preenchimento incompleto da junta;
  - Mordedura ("Undercut"): Reentrâncias agudas causadas pela ação da fonte de calor do arco entre um passe de solda e o metal de base ou outro passe adjacente;
  - Inclusões de impurezas ("Slag inclusions"): Ocorre quando sólidos não-metálicos e partículas de óxido ficam presos entre passes de solda ou entre a solda e o metal de base;
  - Porosidade: Formada pela evolução de gases, na parte posterior da poça de fusão, durante a solidificação da solda.

Figura 3 – Ilustração de defeitos mecânicos e de solda

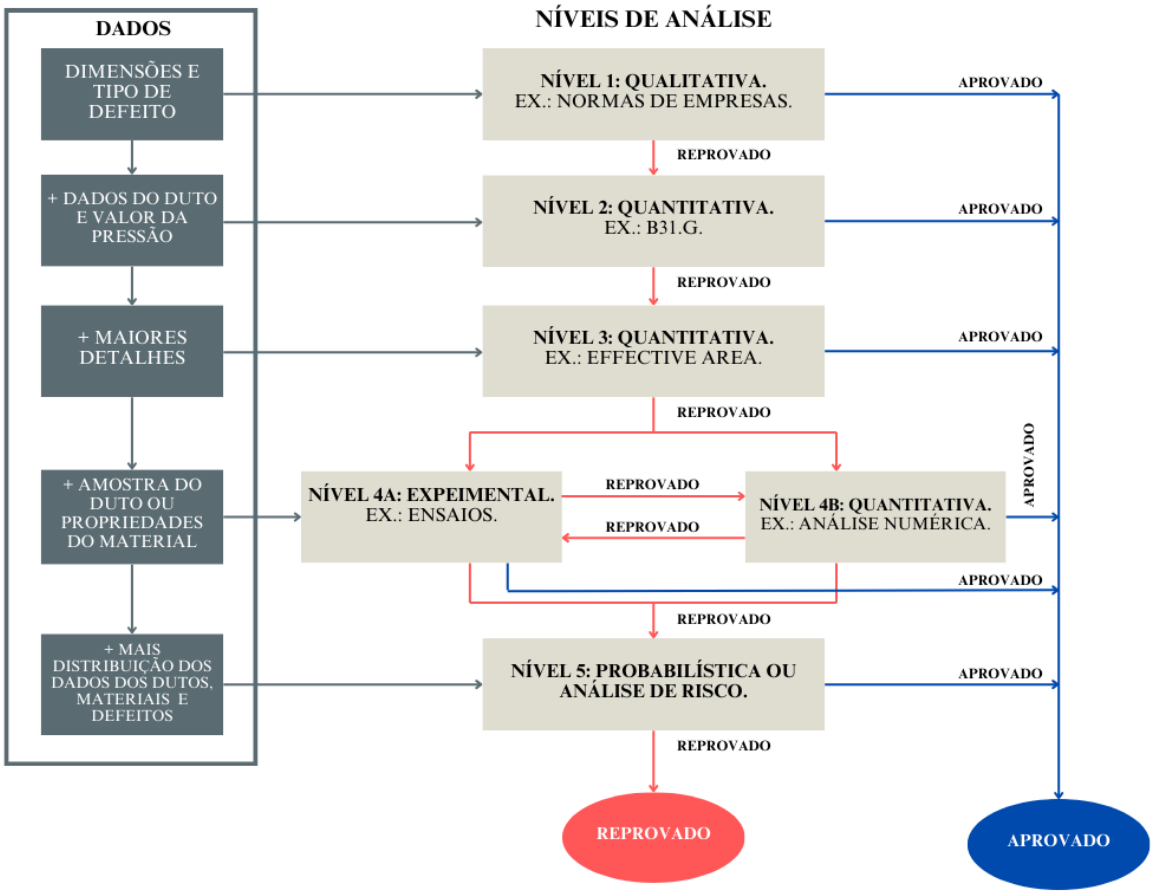


Fonte: MMS (2000).

2.1.2 Níveis de avaliação dos defeitos

Quando é definido que uma avaliação do defeito deve ser feita, deve-se definir o nível de detalhamento necessário para o problema, podendo variar de métodos mais simples e rápidos, como os semi-empíricos já mencionados, até análises de tensão pelo MEF, dependendo de vários fatores como os objetivos da avaliação e a qualidade dos dados disponíveis (COSHAM; KIRKWOOD, 2000). A Figura 4 resume os níveis de avaliação de defeitos, quais os dados necessários e como se dá a evolução da necessidade de níveis mais complexos.

Figura 4 – Níveis de avaliação de defeitos



Fonte: Adaptado de Cosham e Kirkwood (2000).

2.2 MODELAGEM DA GEOMETRIA

De acordo Zeid (2016), em projetos, como de equipamentos ou máquinas, existem duas principais abordagens ao utilizar *softwares* CAD: *bottom-up* e *top-down*. O método do *bottom-up* consiste no desenvolvimento das peças individualmente e a montagem das mesmas formando o equipamento e/ou máquina. Segundo Zeid (2016), esse método é útil para montagens com centenas ou milhares de componentes (montagens pequenas).

Já o método do *top-down*, cria-se o "todo" e depois separa-se cada peça, ou seja, o equipamento/máquina desenvolvido é modelado e então seus subsistemas e peças são separados posteriormente. Conforme Zeid (2016), essa abordagem é melhor para grandes montagens pois diminui os erros na montagem, podendo ser visto como um meio de estabelecer parâmetros e relações entre os componentes da montagem.

Estendendo isso para formas geométricas mais simples, a abordagem *bottom-up* seria a criação de geometrias tridimensionais a partir de esboços com geometrias unidimensionais e bidimensionais, por exemplo para um cilindro, caso do trabalho onde será modelado um duto, é possível gerá-lo a partir de dois círculos com os raios interno e externo do duto. Para o *top-down*, a geração de geometrias se dá por meio de formas "primitivas" como cilindros, cubos e esferas.

Assim, a principal vantagem do método *bottom-up* é ter um passo a passo mais intuitivo. Enquanto o *top-down* traz a configuração final já pronta no início, permitindo, assim, ter uma noção melhor da distribuição de subcomponentes e seus respectivos espaços.

2.3 NOÇÕES BÁSICAS DE MECÂNICA DOS SÓLIDOS

Um dos principais motivos do estudo da mecânica dos sólidos é possuir ferramentas para analisar e projetar máquinas e estruturas e em ambos, análise e projeto, os conceitos de tensão e deformação estão presentes (BEER *et al.*, 2012). Componentes de máquina transmitem força de um local para outro, podendo ser visto como um fluxo ou uma distribuição de força que pode ser melhor visualizada ao isolar superfícies internas do componente. Força distribuída sobre uma superfície leva ao conceito de tensão, componentes da tensão e transformação da tensão para todas as possíveis superfícies num determinado ponto (BUDYNAS; NISBETT, 2011).

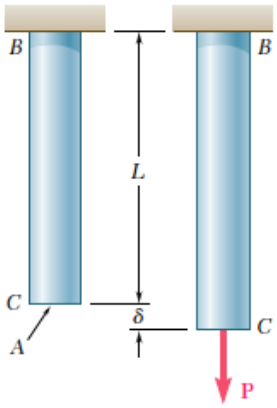
De forma simples, força por unidade de área, ou a intensidade das forças distribuídas numa dada seção, é chamada de tensão (BEER *et al.*, 2012):

$$\sigma = \frac{P}{A} \tag{1}$$

Se uma força axial é aplicada numa barra de seção transversal constante, ela vai deformar uma quantidade δ, Figura 5. Se a mesma carga é aplicada para uma mesma barra, mas de comprimento duas vezes maior, a deformação será duas vezes maior, Figura 6, mesmo que a tensão seja a mesma nos dois casos. Em ambas barras, a proporção da deformação é igual, δ/L, levando ao conceito, simplificado, de deformação unitária ("*strain*"): a deformação unitária normal numa barra sob força axial é a deformação por unidade de comprimento (BEER *et al.*, 2012):

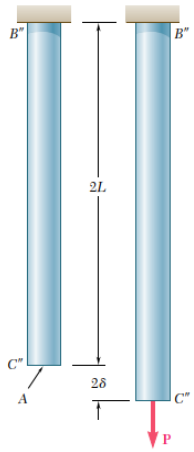
$$\epsilon = \frac{\delta}{L} \tag{2}$$

Figura 5 – Barra de comprimento L sob força axial P



Fonte: Beer *et al.* (2012).

Figura 6 – Barra de comprimento 2L sob força axial P



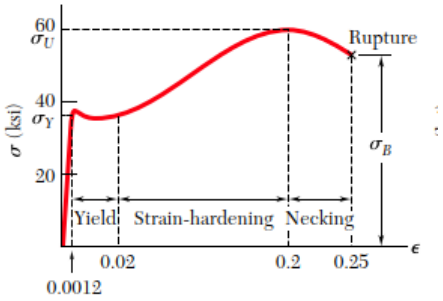
Fonte: Beer *et al.* (2012).

2.3.1 Gráfico tensão x deformação

Ao plotar a tensão pela deformação unitária, obtém-se uma curva que é característica dos materiais, a curva tensão x deformação, sem depender de fatores geométricos, Figura 7. No gráfico , há uma região em que a tensão varia linearmente com a deformação e obedece à lei de Hooke, equação (3). A constante de proporcionalidade E é chamada de módulo de elasticidade ou módulo de Young. O maior valor de tensão em que a lei de Hooke é válida é chamada de limite de proporcionalidade (BEER *et al.*, 2012).

$$\sigma = E\epsilon \tag{3}$$

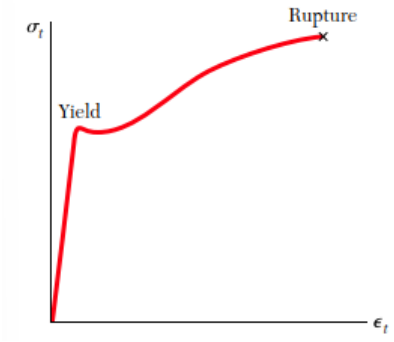
Figura 7 – Tensão por deformação para aço de baixo carbono



Fonte: Beer *et al.* (2012).

Para materiais dúcteis que possuem um ponto de escoamento bem definido, o limite de proporcionalidade quase coincide com o ponto de escoamento. Ao escoar, o material começa a sofrer grandes deformações com incrementos de carga relativamente pequenos. Quando chega num certo ponto, a tensão começa a diminuir ao mesmo tempo que as deformações continuam crescendo, isso se dá pelo efeito do "empescoçamento" do corpo de prova, onde há uma redução na seção transversal devido à instabilidades locais (BEER *et al.*, 2012). Mesmo com a redução da seção transversal, o cálculo da tensão é feito com o valor da área inicial, por isso que a tensão diminui. Ao contabilizar a redução da seção, o gráfico tensão deformação muda e é chamado de curva de tensão verdadeira por deformação verdadeira, Figura 8, enquanto que o anterior é chamado de tensão de engenharia por deformação de engenharia.

Figura 8 – Tensão verdadeira por deformação verdadeira para um material dúctil típico



Fonte: Beer *et al.* (2012).

2.3.2 Tensões, equilíbrio e lei de Hooke generalizada

Conforme Chandrupatla e Belengundu (2015), seja o corpo de volume V e superfície S da Figura 9, pontos do corpo são descritos pelas coordenadas x, y e z e o deslocamento de um ponto é dado pelas três componentes do seu deslocamento:

$$\{u\} = [u, v, w]^T \tag{4}$$

A força distribuída por unidade de volume, por exemplo o peso por unidade de volume, é o vetor f:

$$\{f\} = [f_x, f_y, f_z]^T \tag{5}$$

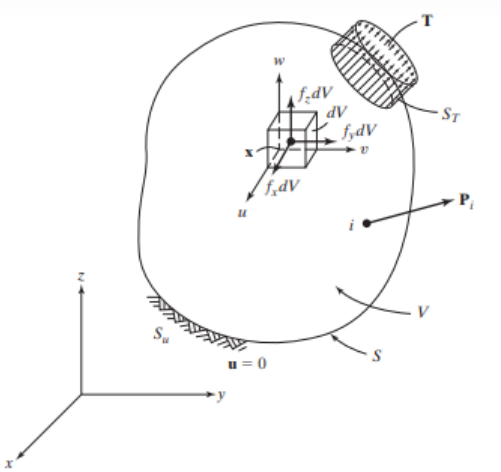
A tração superficial é dada pelos valores das componentes em pontos da superfície:

$$\{T\} = [T_x, T_y, T_z]^T \tag{6}$$

E a força P atuando num ponto i é dada por:

$$\{P\}_i = [P_x, P_y, P_z]_i^T \tag{7}$$

Figura 9 – Corpo tridimensional arbitrário



Fonte: Chandrupatla e Belengundu (2015).

Quando o volume dV da Figura 10 se aproxima de um ponto, o tensor de tensões é representado por uma matriz 3x3 com suas componentes, mas é possível representar pelas seis componentes independentes (CHANDRUPATLA; BELENGUNDU, 2015):

$$\{\sigma\} = [\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau yz, \tau zx]^T \tag{8}$$

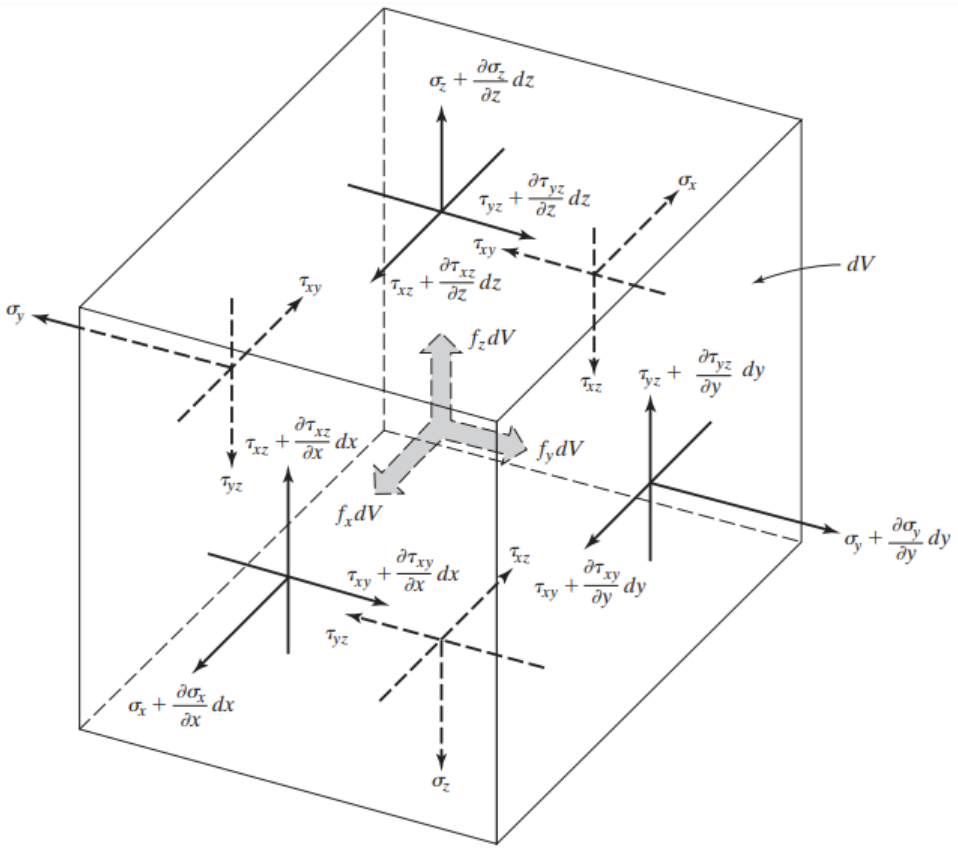
Ao multiplicar as tensões pela área, para obter forças, o equilíbrio dado por $\sum F_x = 0$, $\sum F_y = 0$ e $\sum F_z = 0$ fica como:

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + f_x = 0 \tag{9}$$

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + f_y = 0 \tag{10}$$

$$\frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + f_z = 0 \tag{11}$$

Figura 10 – Volume elementar



Fonte: Chandrupatla e Belengundu (2015).

Segundo Chandrupatla e Belengundu (2015), de forma análoga às tensões, as deformações podem ser escritas como:

$$\{\epsilon\} = \left[\epsilon_x, \epsilon_y, \epsilon_z, \gamma_{xy}, \gamma_{yz}, \gamma_{zx}\right]^T \tag{12}$$

A relação entre deformação e deslocamento, para pequenas deformações, é dada por

$$\{\epsilon\} = \left[\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial w}{\partial z}, \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}, \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right] \tag{13}$$

Já a relação da tensão com a deformação, dada pela lei de Hooke, é, para um estado generalizado de tensão:

$$\{\sigma\} = [D]\{\epsilon\} \tag{14}$$

com

$$[D] = \frac{E}{(1+\upsilon)(1-2\upsilon)} \begin{bmatrix} 1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,5-\upsilon & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,5-\upsilon & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,5-\upsilon \end{bmatrix} \tag{15}$$

2.3.3 Elasticidade vs. Plasticidade

Se a deformação de um corpo "desaparece" quando a carga é retirada, diz-se que o material está no regime elástico, e se possui um ponto de escoamento bem definido, então o limite de proporcionalidade, o limite elástico e o ponto de escoamento são essencialmente iguais. Se o material escoa, quando a carga é removida, a tensão e a deformação diminuem de forma linear, mas a deformação não volta a zero, Figura 11, indicando que há uma deformação residual, chamada de deformação plástica (BEER *et al.*, 2012).

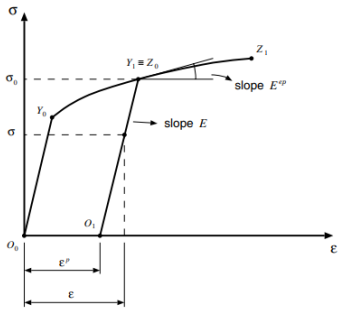
Na teoria da plasticidade para pequenas deformações, uma das principais hipóteses é a possibilidade da decomposição da deformação axial total em duas componentes, como na equação abaixo (DE SOUZA NETO; PERIC; OWEN, 2008). Ramberg e Osgood (1943) propuseram uma formulação para descrever a curva tensão por deformação utilizando três parâmetros, equação (17), em que o primeiro termo do lado direito é a componente elástica e o segundo é a componente plástica, esta última será importante nas entradas dos materiais utilizados nas análises deste trabalho.

$$\epsilon = \epsilon_e + \epsilon_p \tag{16}$$

$$\epsilon = \frac{\sigma}{E} + K\left(\frac{\sigma}{E}\right)^n \tag{17}$$

Sendo K e n constantes.

Figura 11 – Modelo matemático de experimento de tensão uniaxial



Fonte: de Souza Neto, Peric e Owen (2008).

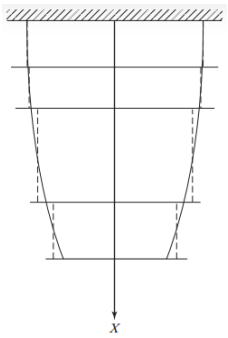
2.4 MÉTODO DOS ELEMENTOS FINITOS

As técnicas desenvolvidas pela mecânica dos sólidos tratam a estrutura como um meio contínuo e resultam em uma solução também contínua. As análises envolvem elementos diferenciais e, consequentemente, a modelagem matemática resultante são equações diferenciais. Entretanto, essas técnicas se limitam a geometrias simples com apoios muito bem comportados, enquanto a maioria das estruturas reais são muito complexas para serem analisadas por essas teorias (ALVES FILHO, 2000). A solução analítica das equações

diferenciais para estruturas com geometrias ou comportamentos complexos pode se tornar muito custosa ou impossível. Com isso, a solução analítica torna-se inviável.

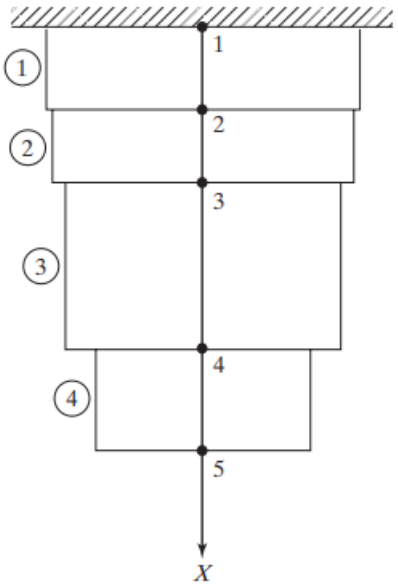
O método dos elementos finitos (MEF) é utilizado quando o problema é muito complicado para ser resolvido por essas técnicas clássicas. No MEF, o modelo contínuo é considerado como uma "montagem" de elementos menores, sendo cada elemento com geometria simples e, por isso, mais fácil de ser analisado. O procedimento do método gera muitas equações algébricas simultâneas, que são geradas e resolvidas computacionalmente. Os resultados são, em geral, inexatos, mas os erros podem ser reduzidos ao processar mais equações, discretizar o componente utilizando mais elementos), e resultados com precisão suficiente são obtidos com um custo computacional razoável (COOK; MALKUS; PLESHA, 1989). Abaixo, nas Figuras 12 a 13, temos ilustrado as duas formas de análise, contínua e discreta.

Figura 12 – Modelo contínuo de uma barra



Fonte: Chandrupatla e Belengundu (2015).

Figura 13 – Modelo aproximado discreto da barra



Fonte: Chandrupatla e Belengundu (2015).

2.4.1 Matriz de rigidez

No caso de uma análise estrutural, o objetivo ao simular um componente é obter sua forma deformada e, consequentemente, informações sobre as forças e tensões atuantes. Essa configuração é determinada pelos deslocamentos nos pontos de conexão dos elementos, chamados de nós, independente da geometria e do carregamento. É a partir desses deslocamentos que é possível calcular as forças internas e o campo de tensões e, com isso, avaliar a resistência da estrutura analisada. Assim, os parâmetros que descrevem o comportamento do objeto analisado são os deslocamentos nodais e, por isso, são chamados de variáveis de estado, pois governam e descrevem o estado de equilíbrio da estrutura. Com isso, o ponto de partida do método dos elementos finitos é a relação entre forças nodais e deslocamentos nodais. Apesar das forças em situações reais não serem exclusivamente nodais, quando aplica-se as condições de carregamento no modelo computacional, os *softwares* de análise convertem essas forças fora dos nós para forças nodais equivalentes (ALVES FILHO, 2000).

A ideia dessa relação entre forças e deslocamentos está relacionada com o conceito de rigidez. A ideia de rigidez é facilmente entendida por meio de uma mola linear, onde a constante elástica é a medida quantitativa da rigidez da mesma e pode ser entendida como um coeficiente de rigidez, por relacionar força com deslocamento (f = kd) (ALVES FILHO, 2000).

A ideia em um elemento finito é similar a da mola, porém de forma mais ampla, podendo envolver carregamentos não axiais e até relações não lineares entre forças e deslocamentos. Nesse caso, as forças serão computadas numa matriz coluna, assim como os deslocamentos, e os coeficientes de rigidez estarão numa matriz, a matriz de rigidez do elemento (ALVES FILHO, 2000). Essa relação para o elemento pode ser visualizada da seguinte forma:

$$\{f\}_j = [k]^e_{j\times j}\{u\}_j \tag{18}$$

Nessa matriz, o significado físico de um coeficiente k_{ij} qualquer é representar a força no grau de liberdade i para um deslocamento unitário no grau de liberdade j, mantendo os outros graus de liberdade fixos (ALVES FILHO, 2000). Com isso, a força i pode ser escrita como:

$$f_i = k_{i1}u_1 + k_{i2}u_2 + ... + k_{ij}u_j \tag{19}$$

Portanto, tem-se um sistema de equações lineares, principal vantagem do método, que será solucionado a partir de rotinas computacionais nos *softwares* de análise. Apesar desse conceito ajudar a entender o método, ele não é exato para elementos bidimensionais e tridimensionais, devido à resistência dos materiais elementar não quantificar relações diretas entre forças e deslocamentos e a dificuldade intrínseca do comportamento do elemento em seu contorno. Então, pode-se entender o termo k_{ij} como uma força relacionada a um deslocamento unitário, mas que de alguma forma deveria contabilizar a situação de

seu contorno com o elemento adjacente (ALVES FILHO, 2000). Assim, de forma geral, para elementos quaisquer, conforme Alves Filho (2000), a matriz de rigidez do elemento é dada por:

$$\{k\}^e = \int_{volume} [B]^T [D][B] dV \tag{20}$$

Em que k^e é a matriz de rigidez do elemento, B é a matriz que relaciona deformações com deslocamentos nodais e D é a matriz de elasticidade que relaciona tensões e deformações, como a matriz da equação (15) proveniente da lei de Hooke generalizada. Da mesma forma que em um conjunto de molas, a rigidez da estrutura é obtida a partir de uma rigidez equivalente utilizando a rigidez de cada elemento do modelo discretizado (ALVES FILHO, 2000):

$$[K] = \sum [k]_i^e \tag{21}$$

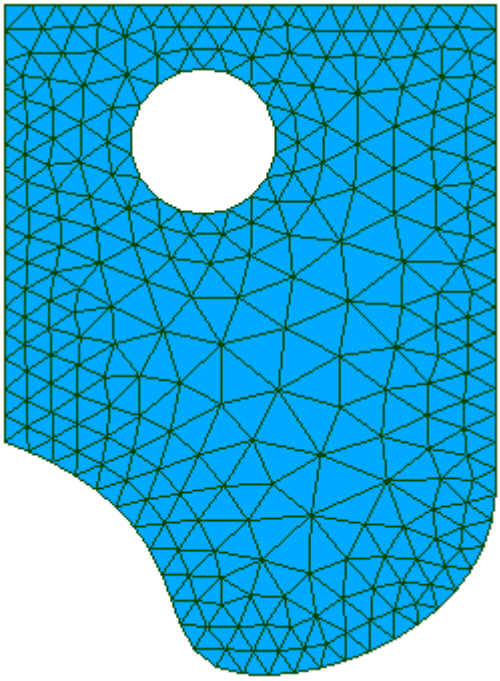
2.4.2 Tipos de elementos

Nesta seção serão apresentadas algumas formulações e comportamentos importantes de alguns elementos comuns nas análises.

2.4.2.1 Elemento triangular sob estado plano de tensão

Devido à flexibilidade da geometria, um corpo qualquer pode ser aproximado por um conjunto de triângulos, por exemplo a chapa com curvatura e furo da Figura 14.

Figura 14 – Exemplo de versatilidade dos elementos triangulares

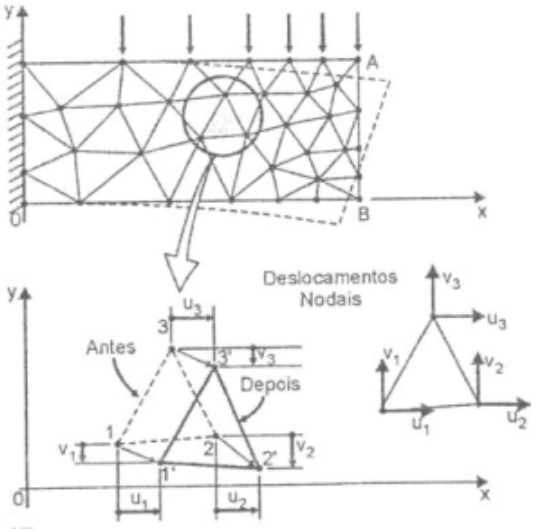


Fonte: Autor (2024).

Para ilustrar o comportamento, a Figura 15 mostra uma chapa retangular simples discretizada com elementos triangulares que está sob carregamento paralelo ao seu plano, ou seja, tem-se um estado plano de tensões, em que os deslocamentos são lineares e nas direções x e y. Ao isolar um elemento, o triângulo obtido possui três nós com cada nó possuindo dois graus de liberdade. Com isso, o elemento em si possui seis graus de liberdade e sua matriz de rigidez será 6x6 (ALVES FILHO, 2000):

$$\{f\}_6 = [k]^e_{6\times 6}\{u\}_6 \tag{22}$$

Figura 15 – Placa discretizada e elemento triangular



Fonte: Alves Filho (2000).

Com seis graus de liberdade, a função de interpolação escolhida para representar o comportamento interno do elemento possuirá seis coeficientes. Assim, sem dar prioridade a nenhuma das coordenadas:

$$u(x,y) = C_1 + C_2x + C_3y \tag{23}$$

$$v(x,y) = C_4 + C_5x + C_6y \tag{24}$$

Portanto, o deslocamento varia linearmente dentro do elemento. Por isso, os deslocamentos são contínuos dentro do elemento (ALVES FILHO, 2000). Calculando as deformações:

$$\epsilon_x = \frac{\partial u}{\partial x} = C_2 \tag{25}$$

$$\epsilon_x = \frac{\partial v}{\partial y} = C_6 \tag{26}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = C_3 + C_5 \tag{27}$$

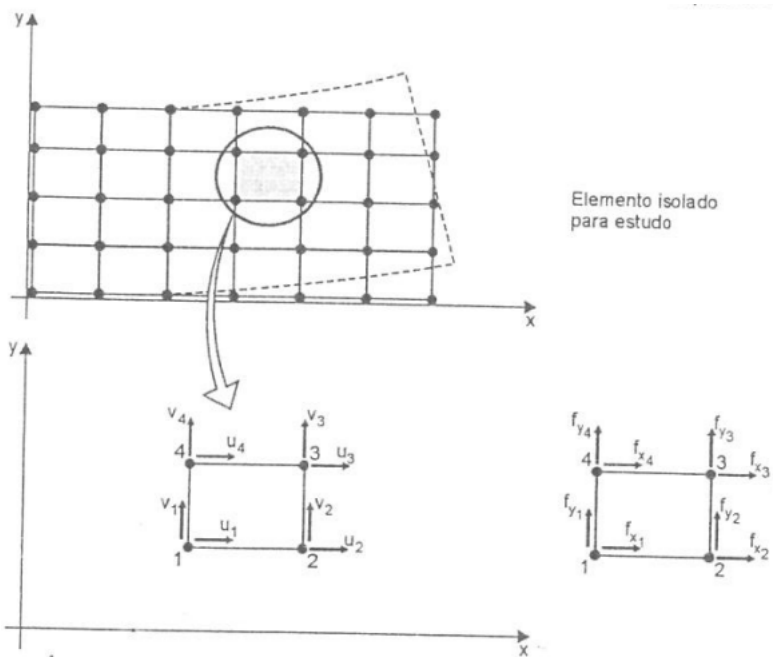
Com isso, percebe-se que as deformações dentro do elemento são **constantes** e, consequentemente, também as tensões. Isso é a principal desvantagem desse tipo de elemento, pois, se tivermos um elemento com altura igual a altura da chapa da Figura 15, então não haverá variação da tensão ao longo da altura, será constante. Precisa-se, então, de uma grande quantidade de elementos para que essa variação seja melhor representada. Além disso, no contorno entre dois elementos, as tensões e deformações assumem dois valores distintos dependendo do elemento de referência. O ideal seria que esses valores não fossem muito diferentes. Para isso, os nós não compartilhados não podem estar muito afastados, ou seja, os elementos precisam ser pequenos (ALVES FILHO, 2000). Assim, este elemento deve ser usado com cautela, ponderando a quantidade de elementos necessária para uma representação válida da estrutura real e nas vantagens da flexibilidade da geometria, que permite a representação de contornos irregulares, e da formulação simplificada.

2.4.2.2 Elemento retangular sob estado plano de tensão

De forma similar ao elemento anterior, tem-se uma placa discretizada e o elemento isolado na Figura 16. O retângulo tem um nó a mais que o elemento triangular, totalizando oito graus de liberdade e, com isso, uma matriz de rigidez 8x8:

$$\{f\}_8 = [k]^e_{8\times 8}\{u\}_8 \tag{28}$$

Figura 16 – Placa discretizada e elemento retangular



Fonte: Alves Filho (2000).

A função de interpolação escolhida para representar o comportamento interno do elemento possuirá oito coeficientes. Assim, sem dar prioridade a nenhuma das coordenadas (ALVES FILHO, 2000):

$$u(x, y) = C_1 + C_2x + C_3y + C_4xy \tag{29}$$

$$v(x, y) = C_5 + C_6x + C_7y + C_8xy \tag{30}$$

Com a adição de um novo termo, o deslocamento agora varia bilinearmente dentro do elemento. Calculando as deformações:

$$\epsilon_x = \frac{\partial u}{\partial x} = C_2 + C_4y \tag{31}$$

$$\epsilon_x = \frac{\partial v}{\partial y} = C_7 + C_8x \tag{32}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = C_3 + C_4x + C_6 + C_8y \tag{33}$$

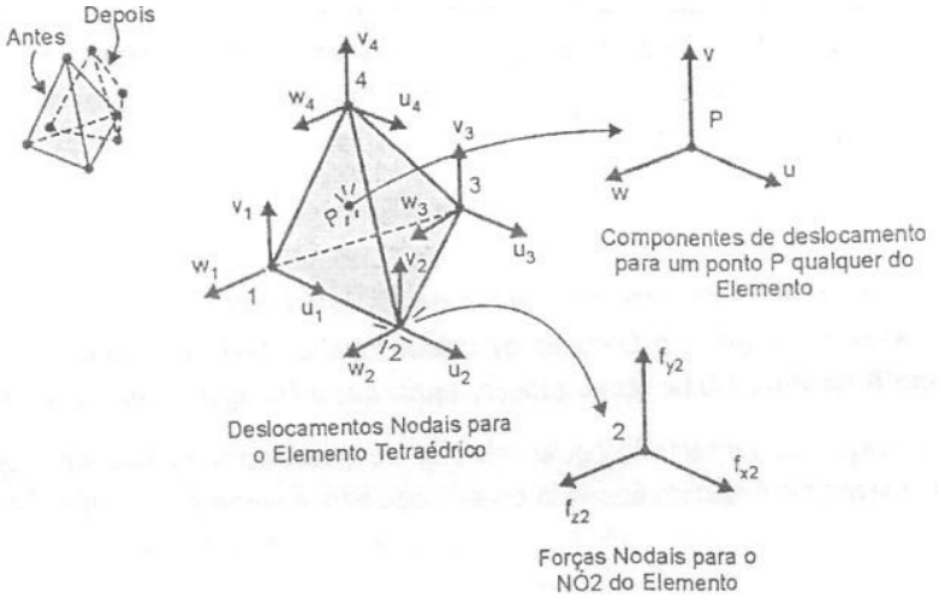
Assim, as deformações irão variar de forma linear dentro do elemento. Apesar de ser um comportamento melhor que o elemento triangular, ainda há limitações. Considerando a deformação em x, ϵ_x, ela varia de forma linear porém não depende de x, ou seja, para um y fixo, a deformação é constante. Assim, caso uma estrutura sofre uma deformação acentuada em determinada direção, deve-se evitar elementos muito longos nessa direção (ALVES FILHO, 2000). Isso também é verdade para as tensões, em que elementos muito longos numa direção que haja uma variação acentuada nas tensões não irão representar bem essa variação.

2.4.2.3 Elemento tetraédrico

De forma análoga ao elemento triangular, a geometria tetraédrica também possui uma flexibilidade de forma que pode representar qualquer geometria sólida (ALVES FILHO, 2000). Ao isolar um elemento tetraédrico de um sólido, tem-se 4 nós com três graus de liberdade cada, Figura 17. Assim, o elemento possui 12 graus de liberdade e sua matriz de rigidez será 12x12:

$$\{f\}_{12} = [k]^e_{12\times12}\{u\}_{12} \tag{34}$$

Figura 17 – Elemento tetraédrico isolado



Fonte: Alves Filho (2000).

A função de interpolação escolhida para representar o comportamento interno do elemento possuirá doze coeficientes. Sem dar prioridade a nenhuma das coordenadas (ALVES FILHO, 2000):

$$u(x, y, z) = C_1 + C_2x + C_3y + C_4z \tag{35}$$

$$v(x, y, z) = C_5 + C_6x + C_7y + C_8z \tag{36}$$

$$w(x, y, z) = C_9 + C_{10}x + C_{11}y + C_{12}z \tag{37}$$

Calculando as deformações:

$$\epsilon_x = \frac{\partial u}{\partial x} = C_2 \tag{38}$$

$$\epsilon_x = \frac{\partial v}{\partial y} = C_7 \tag{39}$$

$$\epsilon_z = \frac{\partial w}{\partial z} = C_{12} \tag{40}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = C_3 + C_6 \tag{41}$$

$$\gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = C_4 + C_{10} \tag{42}$$

$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} = C_8 + C_{11} \tag{43}$$

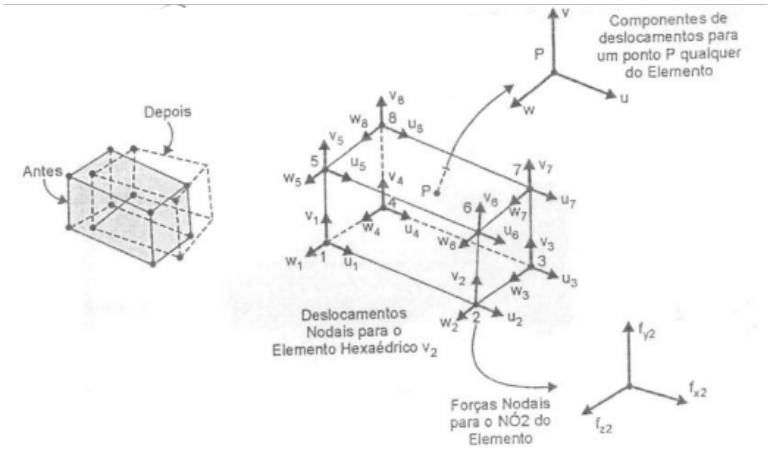
Novamente, tem-se um elemento com deformações e, consequentemente, tensões constantes. As limitações desse elemento são similares ao elemento triangular e são contornadas também ao refinar a malha. O elemento tetraédrico faz no espaço aquilo que o triangular faz no plano (ALVES FILHO, 2000).

2.4.2.4 Elemento hexaédrico

A Figura 18 mostra o elemento hexaédrico isolado de um componente discretizado. O elemento apresentado tem oito nós e três graus de liberdade em cada, totalizando vinte e quatro graus de liberdade no elemento em si e, com isso, uma matriz de rigidez 24x24:

$$\{f\}_{24} = [k]^e_{24\times 24}\{u\}_{24} \tag{44}$$

Figura 18 – Elemento hexaédrico isolado



Fonte: Alves Filho (2000).

Assim, a função de interpolação será (ALVES FILHO, 2000):

$$u(x,y,z) = C_1 + C_2x + C_3y + C_4z + C_5xy + C_6xz + C_7yz + C_8xyz \tag{45}$$

$$v(x,y,z) = C_9 + C_{10}x + C_{11}y + C_{12}z + C_{13}xy + C_{14}xz + C_{15}yz + C_{16}xyz \tag{46}$$

$$w(x,y,z) = C_{17} + C_{18}x + C_{19}y + C_{20}z + C_{21}xy + C_{22}xz + C_{23}yz + C_{24}xyz \tag{47}$$

Calculando as deformações:

$$\epsilon_x = \frac{\partial u}{\partial x} = C_2 + C_5y + C_6z + C_8xy \tag{48}$$

$$\epsilon_x = \frac{\partial v}{\partial y} = C_{11} + C_{13}x + C15z + C_{16}xz \tag{49}$$

$$\epsilon_z = \frac{\partial w}{\partial z} = C_{20} + C_{22}x + C_{23}y + C_{24}xy \tag{50}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = C_3 + C_5 x + C_7 z + C_8 xz + C_{10} + C_{13} y + C_{14} z + C_{16} yz \tag{51}$$

$$\gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = C_4 + C_6 x + C_7 y + C_8 xy + C_{18} + C_{21} y + C_{22} z + C_{24} yz \tag{52}$$

$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} = C_{12} + C_{14} x + C_{15} y + C_{16} xy + C_{19} + C_{21} x + C_{23} z + C_{24} xz \tag{53}$$

Assim como o elemento tetraédrico faz no espaço o que o triangular faz no plano, o hexaédrico faz no espaço o que o retangular faz no plano. Aqui, as deformações também variam de forma linear dentro do elemento. A limitação principal é similar a do retângulo, onde a variação linear em determinada direção é independente dela, então não deve-se utilizar elementos muito longos na direção que tenha deformações acentuadas (ALVES FILHO, 2000). Essa mesma observação é válida para a tensão.

2.4.2.5 Elementos de ordem maior

Todos os elementos discutidos anteriormente possuem limitações na forma como representam as deformações e o campo de tensões. Uma melhor representação só seria possível se as funções que exprimem os deslocamentos possuíssem um grau maior. Isso implica em funções com mais coeficientes, mas estes mesmos estão ligados à quantidade de graus de liberdade do elemento. Como a quantidade de graus de liberdade por nós está limitada pela teoria da elasticidade (dois para o estado plano de tensões e três para o estado triaxial de tensões), então a única forma de aumentar a quantidade de graus de liberdade de um elemento é aumentar a quantidade de nós. Essa é a ideia por trás de elementos com ordem superior a um. Os elementos lineares possuem nós apenas nos vértices, elementos parabólicos possuem um nó intermediário entre os vértices e elementos cúbicos possuem dois nós intermediários. Com isso, as funções de interpolação serão mais ricas na representação dos deslocamentos, assim como suas derivadas, as deformações (ALVES FILHO, 2000).

2.4.3 Noções básicas para análise não linear

A análise é considerada não linear se a matriz de rigidez ou o vetor da carga dependem dos deslocamentos. Há dois tipos de fontes de não linearidade em estruturas, geométrica (associada com mudanças na configuração da estrutura, como grandes deformações) e de material (associada com mudanças nas propriedades do material, como a plasticidade). Apesar de muitos problemas poderem ser bem aproximados por equações lineares, algumas situações físicas possuem não linearidades muito grandes para serem

ignoradas. Alguns exemplos de situações de não linearidades são: cargas oscilatórias devido a vórtices causados pelo escoamento de um fluido sobre uma estrutura e o processo de soldagem que causa mudanças em diversas propriedades do material (COOK; MALKUS; PLESHA, 1989). Utilizar modelos não lineares faz com que a solução da análise numérica se torne iterativa, pois as forças e a matriz de rigidez mudam para cada mudança nos deslocamentos, precisando recalculá-las (CABRAL, 2007). Com isso, a análise não linear deve ser utilizada apenas quando necessário, já que o custo computacional é muito maior que a análise linear.

3 METODOLOGIA

O trabalho utilizou, principalmente, o método hipotético dedutivo, utilizando-se de estudo de casos, mas também tomando como base trabalhos realizados na mesma área. É uma pesquisa exploratória, descrevendo processos comuns a todas as análises numéricas de dutos com defeitos de corrosão, e básica estratégia, sem atacar um problema em específico, mas os resultados finais irão auxiliar possíveis aplicações.

Além disso, a pesquisa foi dividida em subtarefas de forma a organizar o passo a passo. No total são seis etapas ou *sprints* divididas da seguinte forma:

- *Sprint* 1: Modelagem da geometria do defeito, regiões de transição da malha e do duto;
- *Sprint* 2: Geração e ajuste da malha para cada região;
- *Sprint* 3: Geração do *script*, ajustes e testes para diferentes geometrias;
- *Sprint* 4: Elaboração do *script* para converter o arquivo gerado pelo *software* CAD para o formato do *software* CAE;
- *Sprint* 5: Realização de análises computacionais de integridade dos dutos utilizando modelos de trabalhos da área gerados pelas ferramentas e avaliação da qualidade ao comparar os resultados.

3.1 FERRAMENTAS

A ferramenta principal é o *software* SALOME (SALOME, 2024), na versão 9.9.0, que é um programa de livre acesso com mais de vinte anos e possui poderosos módulos para atender às etapas de pré e pós processamento de análises numéricas, sendo utilizados neste trabalho o SHAPER, para a geração da geometria e sua parametrização, e o MESH, para gerar a malha e ajustá-la com os algoritmos disponíveis. No *software*, é possível, após a geração paramétrica do modelo ou da malha, gerar um arquivo de comandos, *script*, na linguagem Python (PYTHON, 2024) que permite refazer o modelo ou alterá-lo sem a necessidade da repetição manual das etapas de desenho e discretização. Além de atender às necessidades do trabalho, o fato de ser livre acesso facilitará o acesso e evolução da ferramenta posteriormente. No final, foram feitas análises de integridade utilizando o *software* comercial ANSYS na versão 2023 R1 estudantil, de forma a comprovar a eficácia do modelo gerado ao comparar os resultados com trabalhos da área.

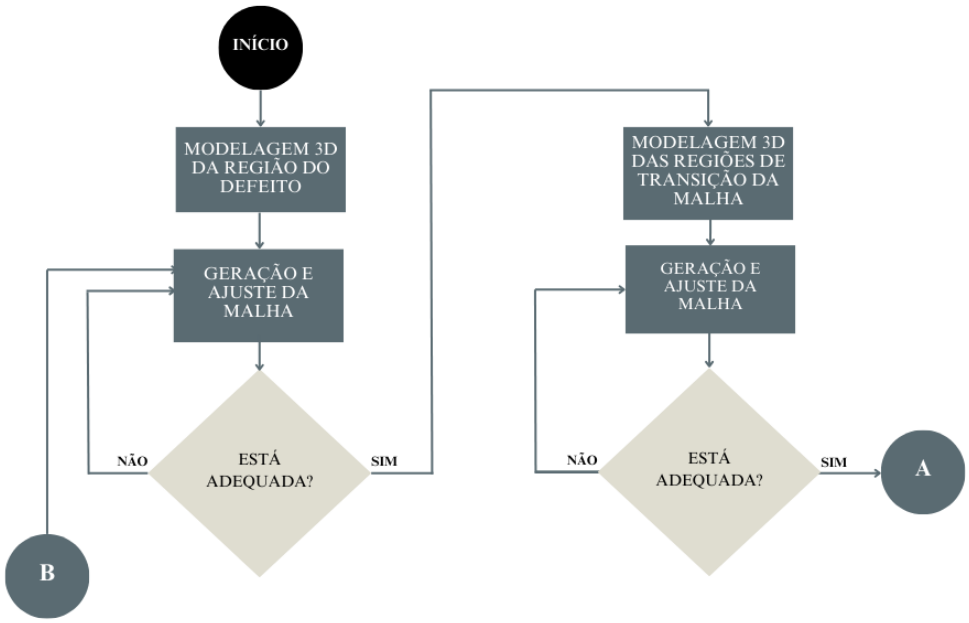
Além disso, foi desenvolvido um *script* para a conversão do arquivo da malha do SALOME para um formato importável pelo ANSYS. Esse *script* é na linguagem Python, por ser simples e acessível, e contou com o auxílio da inteligência artificial da OpenAI, ChatGPT (OPENAI, 2024), para montar as funções do código, mas ajustes e modificações precisaram ser feitos para garantir o formato adequado do arquivo final, proporcionando, assim, a comunicação entre *softwares*.

3.2 DESENVOLVIMENTO DO PROJETO

3.2.1 Fluxo de trabalho

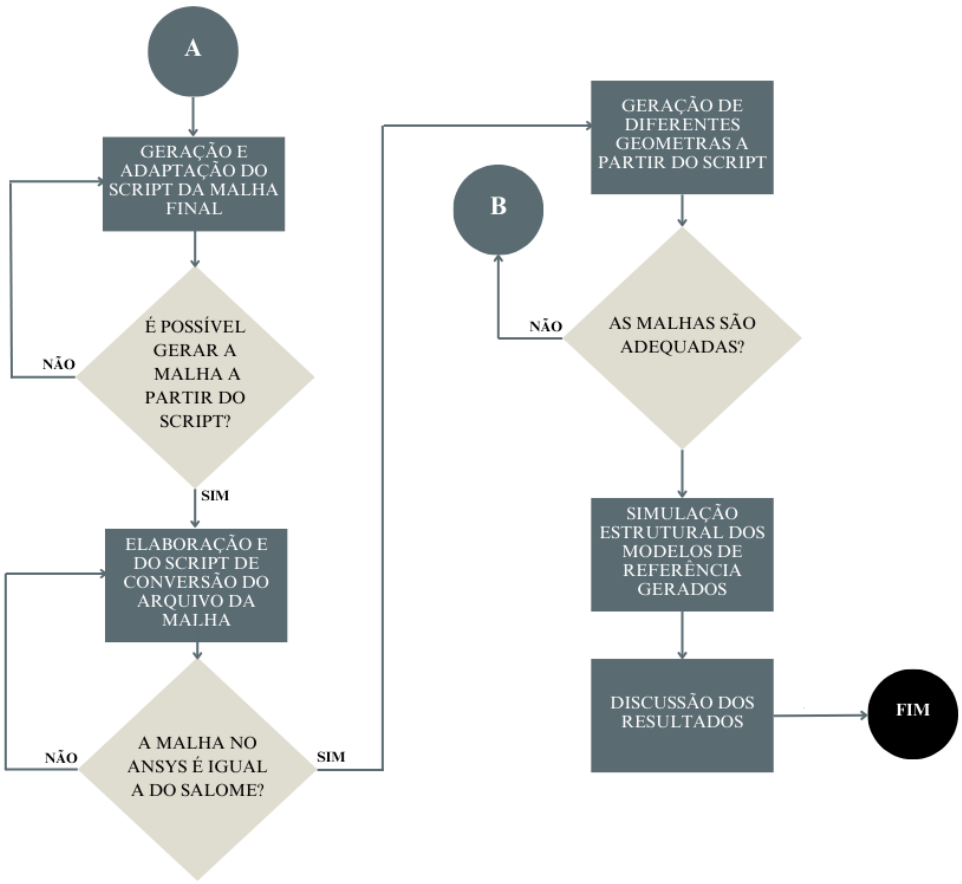
Para a realização da pesquisa, a execução dos estudos de caso foi da seguinte forma: gerou-se a geometria de um defeito arbitrário, desenha-se então uma região ao redor do defeito, que será referenciada por RD ao longo do trabalho, com dimensões baseadas no defeito e é onde a malha é mais refinada. Com isso, a malha dessa região é criada, sendo ajustada até estar satisfatória. Depois, repete-se esses passos para as regiões de transição da malha. Gera-se então o *script* por meio do comando do SALOME *dump study* e ajusta-se até ser possível gerar o modelo final e o arquivo com as informações da malha utilizando apenas o *script* e o comando *load script*. Elabora-se, então, um *script* em Python para converter o arquivo da malha exportado no SALOME para um formato no qual o ANSYS consegue importar. Por fim, modela-se geometrias já estudadas em outros trabalhos por meio dos recursos desenvolvidos e uma análise não linear é realizada para comparar os resultados. As Figuras 19 e 20 ilustram o processo descrito.

Figura 19 – Fluxograma da pesquisa parte 1



Fonte: Autor (2024).

Figura 20 – Fluxograma da pesquisa parte 2



Fonte: Autor (2024).

3.2.2 Geometria

Antes da criação de cada geometria, definiu-se no SHAPER os parâmetros da Tabela 1 com valores arbitrários, sendo todos os valores lineares em mm e os angulares em graus. A Figura 21 mostra os parâmetros definidos no *software* e a Figura 22 ilustra as dimensões na RD.

Tabela 1 – Parâmetros geométricos

Parâmetro	Descrição	Valor
DO	Diâmetro externo do duto	450
T	Espessura do duto	10
RO	Raio externo do duto	DO/2
RI	Raio interno do duto	RO-T
D	Profundidade do defeito	5
AF	Comprimento angular do defeito	10
L	Comprimento longitudinal do defeito	50
TH	Comprimento angular da RD	1.3 × AF/2
C	Comprimento longitudinal da RD	1.3 × L/2
RA	Raio de adoçamento na parede do defeito	3
RC	Raio de adoçamento na quina do defeito	10

Fonte: Autor (2024).

Figura 21 – Parâmetros geométricos no SALOME

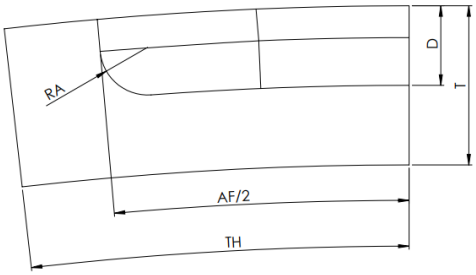
Name	Expression	Result
Parameters		
DO	450	450
T	10	10
RO	DO/2	225
RI	RO-T	215
D	5	5
AF	10	10
L	50	50
TH	AF*1.3/2	6.5
C	L*1.3/2	32.5
RA	3	3
RC	10	10

Fonte: Autor (2024).

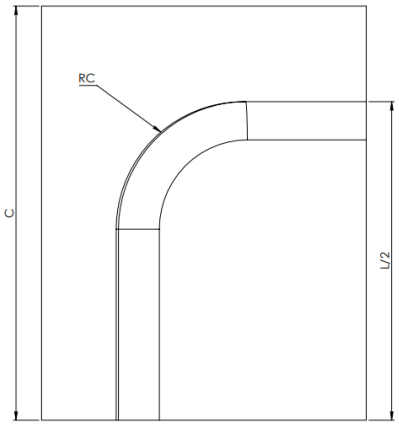
O método utilizado para a modelagem da geometria foi o *bottom-up*, criando a regiões individualmente para no final formar o duto com o defeito, e foi feito para um quarto do duto tanto para aproveitar da simetria em dois planos, economizando poder computacional durante a análise, mas também por ser possível espelhar a malha no módulo MESH. Assim, a partir dos parâmetros listados, o defeito foi modelado, assim como a RD. Com o comando de *split*, removeu-se a região da RD equivalente ao defeito. A escolha da RD ser 30% maior que o defeito foi para não ter uma região de malha fina extensa

ao mesmo tempo que representa bem a região com espessura menor. A Figura 22 ilustra como os parâmetros foram utilizados na geometria e mostra como ficou o resultado final da RD.

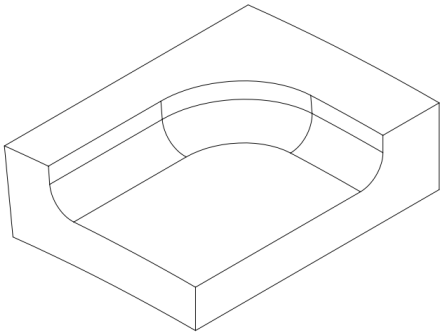
Figura 22 – Geometria da RD



(a) Vista frontal da RD.



(b) Vista de topo da RD.

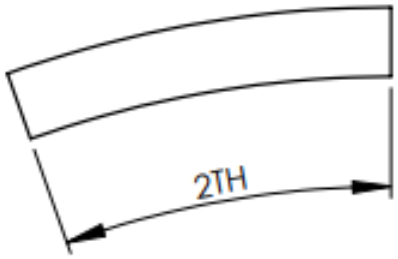


(c) Resultado final da RD

Fonte: Autor (2024).

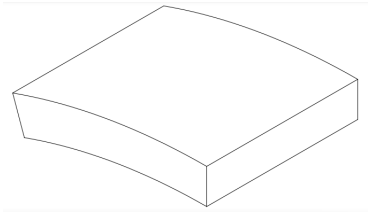
Já para a primeira região de transição (RT1) e a segunda região de transição RT2, os comprimentos angular e longitudinal são o dobro da região anterior, RT1 dobra TH e C, que são as dimensões de RD, e RT2 dobra RT1, facilitando a conexão das malhas entre as regiões, as Figuras 23 e 24 ilustram as dimensões das geometrias.

Figura 23 – Geometria da RT1



(a) Vista frontal da RT1.

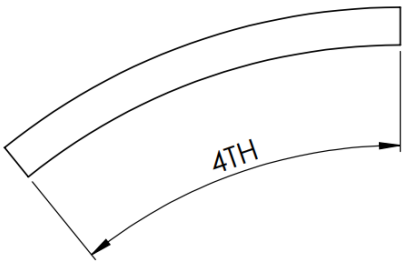
(b) Vista de topo da RT1.



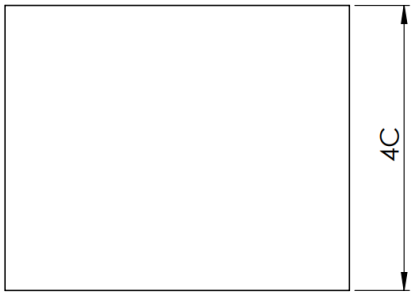
(c) Resultado final da RT1

Fonte: Autor (2024).

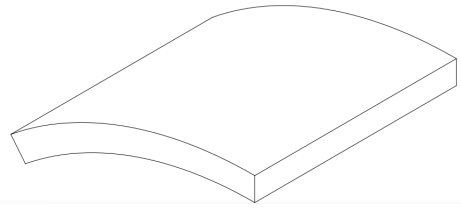
Figura 24 – Geometria da RT2



(a) Vista frontal da RT2.



(b) Vista de topo da RT2.

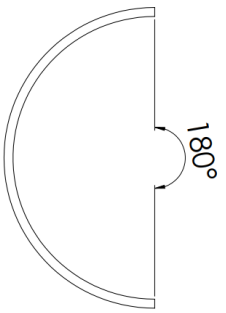


(c) Resultado final da RT2

Fonte: Autor (2024).

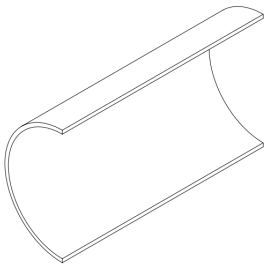
Para a região de transição (RT3), a modelagem foi feita para o quarto do duto completo com o comprimento sendo definido como 1,7 vezes o diâmetro do duto a partir da face do defeito até a extremidade, (DO*1.7)+(L/2) do centro do defeito à extremidade, de forma que as extremidades não interfiram no comportamento na RD durante a análise (BENJAMIN *et al.*, 2005). A Figura 25 mostra o quarto do duto completo e suas dimensões.

Figura 25 – Geometria da RT3



(a) Vista frontal da RT3.

(b) Vista de topo da RT3.

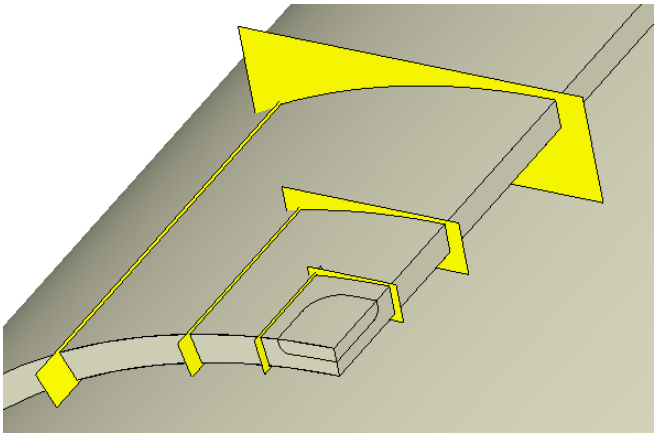


(c) Resultado final da RT3

Fonte: Autor (2024).

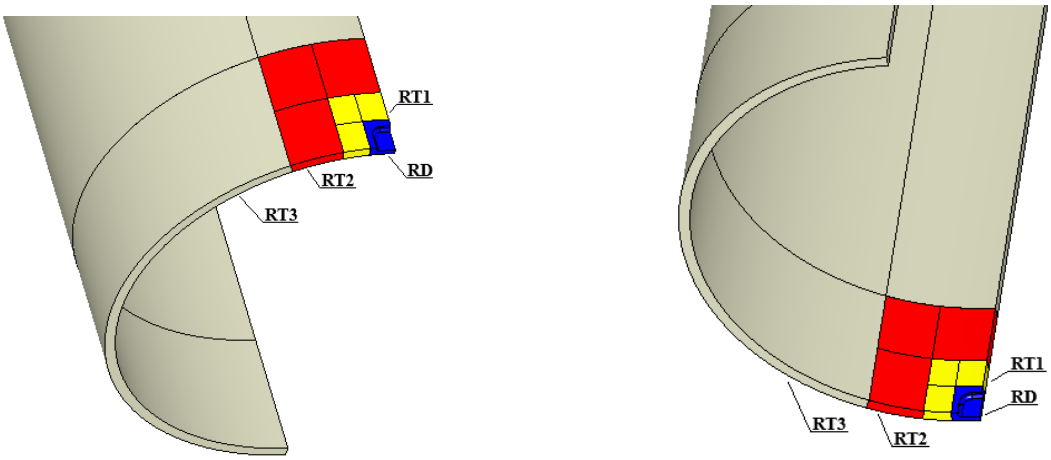
As regiões maiores foram desenhadas de forma que sobrepõem as menores. Então, para remover esses pedaços desnecessários, a função *split* foi utilizada novamente usando planos localizados nas faces das regiões. O planos utilizados para a divisão podem ser vistos na Figura 26, estão em amarelo, e a geometria final dividida pode ser vista na Figura 27a, onde a região azul é a RD, em amarelo é a RT1, em vermelho é a RT2 e, em cinza, a RT3. De forma análoga, também foi gerada uma geometria para o caso de defeitos de corrosão internos Figura 27b.

Figura 26 – Planos utilizados para eliminar as regiões sobrepostas



Fonte: Autor (2024).

Figura 27 – Geometria final



(a) Defeito externo.

(b) Defeito interno.

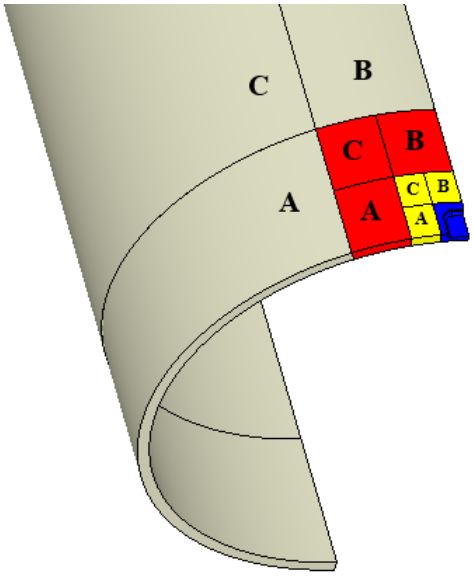
Fonte: Autor (2024).

3.2.3 Malha

Para a construção da malha, a metodologia geral foi criar grupos no SHAPER com os sólidos a serem discretizados, ilustrados na Figura 28 com a RD e as sub-regiões das RT. Em cada uma dessas regiões, mais grupos de geometrias específicas, arestas e faces, foram criados com o intuito de controlar a malha nelas, como será descrito logo abaixo. As configurações descritas nesta seção são resultados de testes por tentativa e erro até conseguir uma malha adequada.

Para a RD, criou-se um parâmetro RCC no *notebook* do SALOME com o valor igual ao comprimento do arco do raio de adoçamento RA dividido por quatro, πRA/16, sendo utilizado para controlar o tamanho em geometrias específicas da RD. Globalmente,

Figura 28 – Subdivisões em cada RT

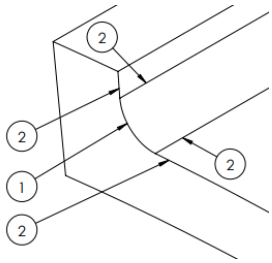


Fonte: Autor (2024).

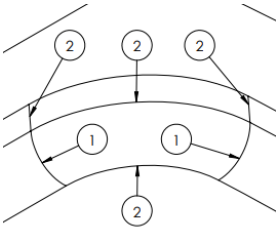
as configurações de malha são NETGEN 3D, alterando apenas o parâmetro de tamanho mínimo do elemento para RCC, NETGEN2D, parâmetros automáticos do *software* (*default*), e *wire discretization*. O primeiro algoritmo configura a divisão de sólidos em tetraedros, o segundo é similar mas para elementos 2D e o último configura a divisão das arestas de acordo com uma hipótese, no caso, 8 elementos divididos de forma equidistante. Essas configurações globais funcionam para as regiões em que não há uma sub-malha que as sobreponham.

Controlando de forma específica, definiu-se quatro elementos nas arestas dos arcos do filete RA e nas arestas adjacentes foi definido elementos do mesmo tamanho utilizando o *local length*, por isso a definição da variável RCC e globalmente esse será o menor tamanho de elemento, como citado nas configurações globais. A Figura 29 mostra em 1 as arestas com definição de 4 elementos e em 2 aquelas com controle no tamanho do elemento.

Figura 29 – Discretização das regiões próximas ao raio RA



(a) Lado da RD.

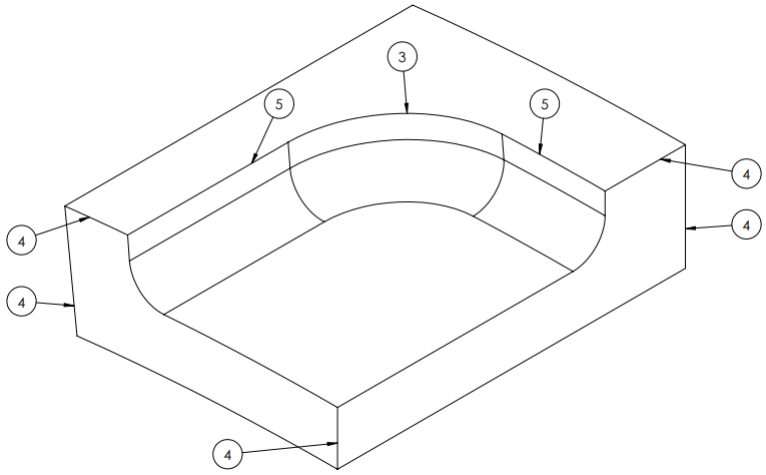


(b) Região próxima ao RC.

Fonte: Autor (2024).

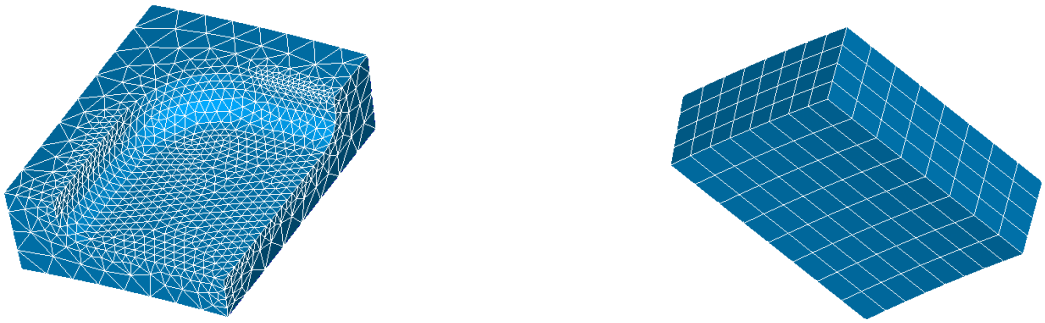
Além dessas regiões próximas ao RA, as arestas indicadas na Figura 30 também tiveram o número de elementos imposto, sendo a aresta indicada por 3 com 12 elementos, as indicadas por 4 possuem 4 elementos e as indicadas por 5 possuem 16. As que não possuem sub-malha obedecem à configuração global, 8 elementos. Para as faces, aquelas em contato com a RT1 e a da superfície interna do duto possuem uma imposição que os elementos devem ser quadriláteros, como mostra a Figura 31b. A configuração final pode ser vista na Figura 31.

Figura 30 – Arestas com sub-malhas



Fonte: Autor (2024).

Figura 31 – Malha final da RD



(a) Discretização da região de perda de material.

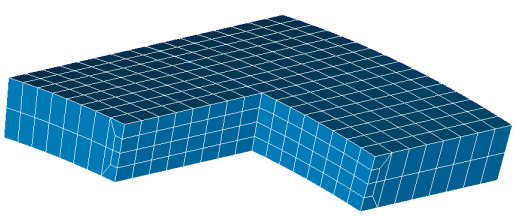
(b) Lados com a malha quadrilateral.

Fonte: Autor (2024).

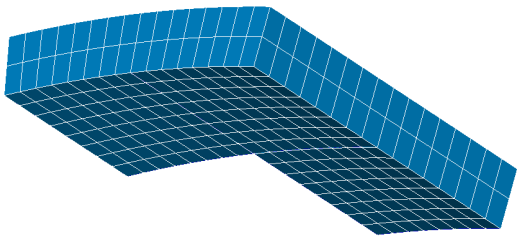
Para a RT1, a malha faz uma transição na quantidade de elementos na espessura, saindo de quatro elementos na RD para dois. O tipo de transição que melhor funcionou durante o desenvolvimento foi por meio de quadriláteros, vista na Figura 32. Para as configurações globais das RT1 A e RT1 B, o algoritmo para os elementos 3D é o hexahedron (i,j,k), com configuração *default*, que divide os sólidos em hexaedros. Para a configuração 2D, tem-se *quadrangle mapping*, configurando para as transições possuírem a propriedade

já citada *quadrangle preference*, dividindo os elementos 2D em quadriláteros. E, por fim, *wire discretization* para elementos 1D definindo 8 elementos nas arestas, assegurando a compatibilidade com a RD, que também possui 8 elementos nas interface com a RT1. Os controles específicos se dão nas arestas radiais, sendo nas da interface com a RD 4 elementos na espessura e nas da interface com a RT2 com 2 elementos. Para a RT1 C, devido ao formato da transição, vista na Figura 32c, precisou-se definir as configurações globais para elementos 3D como NETGEN 3D, para que possam ser gerados elementos piramidais e a malha internamente à região consiga ficar coerente. Com isso, o resultado pode ser visto na Figura 32 e a compatibilidade com a RD pode ser vista na Figura 33.

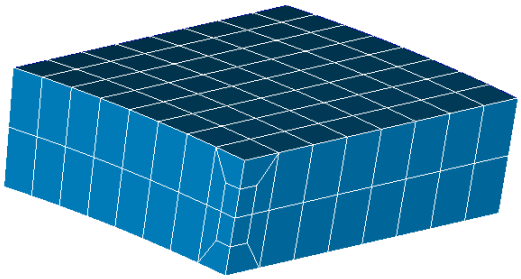
Figura 32 – Malha final da RT1



(a) Lado com a interface com a RD.



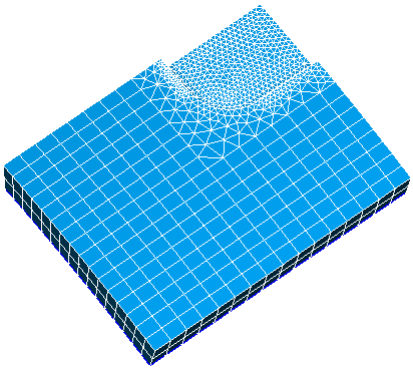
(b) Lado da interface com a RT2.



(c) Transição na região RT1 C.

Fonte: Autor (2024).

Figura 33 – Conexão entre a RD e a RT1

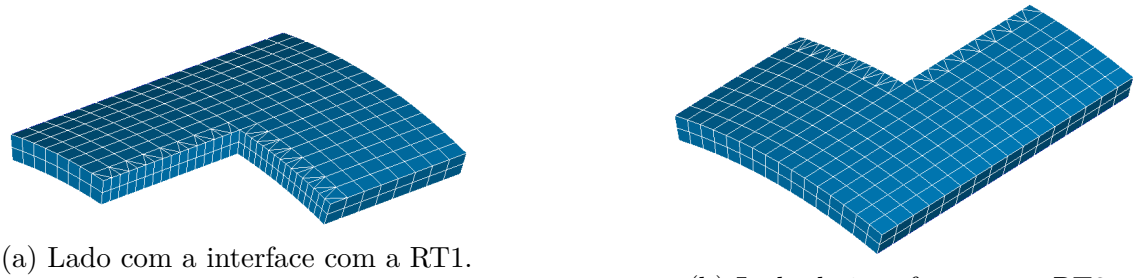


Fonte: Autor (2024).

Para a RT2 e RT3, a transição da malha se dará na superfície e o tamanho do elemento é dobrado em relação à região anterior. Nessas regiões, a transição que melhor

funcionou foi com elementos triangulares, como mostra a Figura 34. Para as configurações globais, na RT2 como um todo, tem-se novamente hexahedron (i,j,k) para a configuração 3D, *quadrangle mapping* para 2D nas regiões A e B (*default* na C), e *wire discretization* determinando 8 elementos nos elementos 1D. Para os controles específicos, são definidos 2 elementos nas arestas radiais e nas arestas da interface com a RT1 só de definir o mesmo número de elementos já garante a compatibilidade, pois, graças ao jeito que foram modeladas (cada região possuindo o comprimento linear e angular sendo o dobro da anterior), o comprimento linear da RT2 A é igual ao da RT1 A + RT1 C, assim como RT2 B é igual a RT1 B + RT1 C circunferencialmente, ou seja, define-se 16 elementos nas arestas de interface para compatibilizar com 8 elementos em cada região, A e C ou B e C, da RT1. Já para RT2 C, o único controle é feito para definir 2 elementos nas arestas radiais, com todas as outras obedecendo à configuração global. A malha final da região pode ser vista na Figura 34 e a compatibilização na Figura 35.

Figura 34 – Malha final da RT2

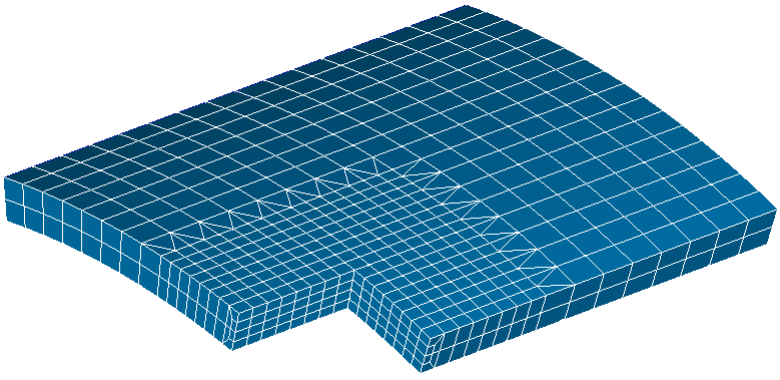


(a) Lado com a interface com a RT1.

(b) Lado da interface com a RT3.

Fonte: Autor (2024).

Figura 35 – Conexão entre a RT1 e a RT2

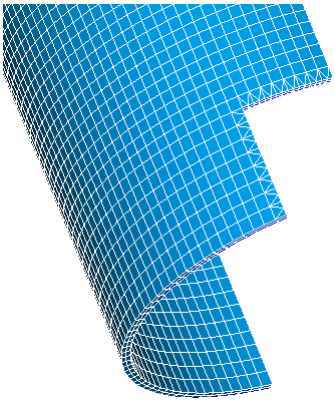


Fonte: Autor (2024).

Para a RT3, como os comprimentos angulares e lineares das regiões são maiores, são feitos mais controles específicos, mas as configurações seguem de forma semelhante à RT2, com exceção da região B que foi invertida a configuração global para 16 elementos

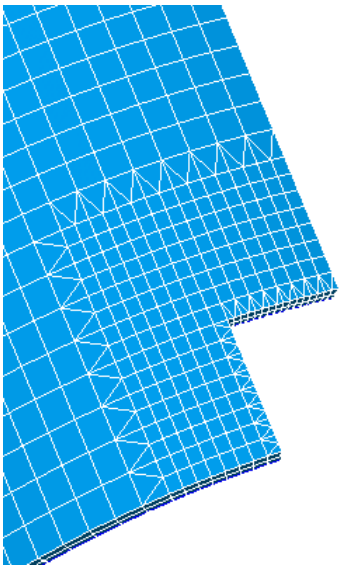
e as arestas circunferenciais da extremidades definidas para possui 8 elementos (ao contrário da RT2, que a global é 8 e o controle local é feito na interface), porém não muda o resultado final, podendo ser feito igual à RT2, sendo feito dessa forma por conta de testes durante o desenvolvimento. Para os controles específicos, todas as regiões possuem a definição de 2 elementos na espessura. Além disso, na RT3 A e C, definiu-se 48 elementos circunferencialmente e, na RT3 B e C, 48 elementos no sentido longitudinal. Para os controles na interface, a região A possui 16 elementos, de forma similar ao que foi comentado anteriormente na RT2, e, na região B, o controle foi feito na outra extremidade, como já comentado, com 8 elementos. A malha final da região RT3 pode ser vista na Figura 36 e a interface com a RT2 na Figura 37.

Figura 36 – Malha final da RT3



Fonte: Autor (2024).

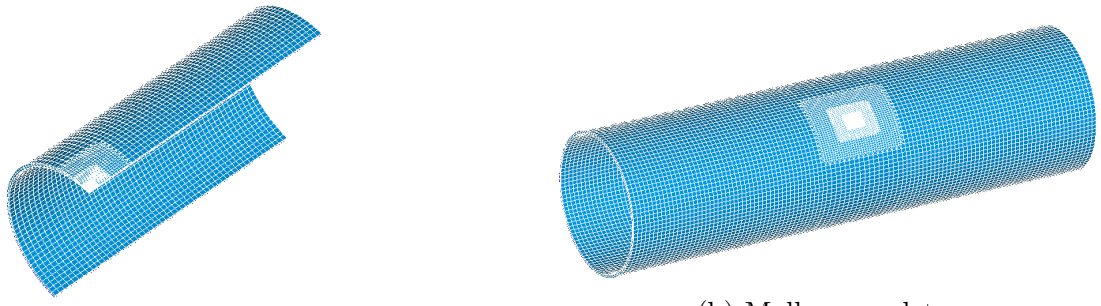
Figura 37 – Conexão entre a RT2 e a RT3



Fonte: Autor (2024).

Por fim, dois caminhos foram seguidos. O primeiro utilizou o recurso *compound mesh* do SALOME que junta todas as malhas anteriormente descritas formando uma única malha, e exclui os elementos repetidos, exportando em seguida essa malha no formato de arquivo .dat e, finalizando, gerando o *script* de todo esse passo a passo por meio do *dump study*. O segundo caminho foi para gerar uma malha do duto por completo, não somente um quarto dele, utilizando o recurso de simetria e configurando para espelhar as malhas, juntando-as em seguida por meio do *compound mesh*, novamente, exportando a malha completa no formato .dat e gerando o *script*. Todos esses processos descritos nessa seção também foram feitos para a geometria com defeitos internos e todas as malhas finais podem ser vistas nas Figuras 38 e 39. O código final, para defeitos externos, é apresentado na seção dos resultados.

Figura 38 – Malha final para defeitos externos

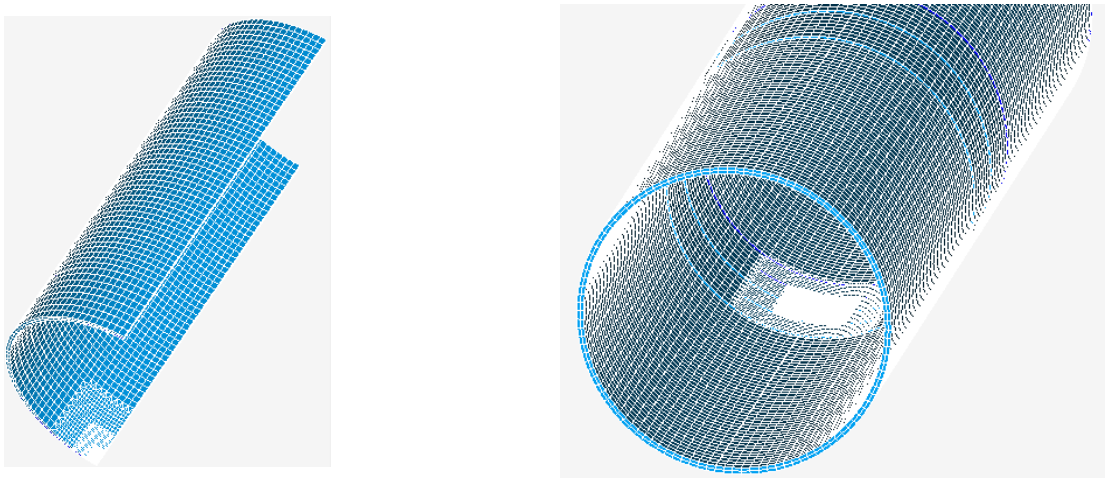


(a) Um quarto da malha.

(b) Malha completa.

Fonte: Autor (2024).

Figura 39 – Malha final para defeitos internos



(a) Um quarto da malha.

(b) Malha completa.

Fonte: Autor (2024).

3.2.4 Teste da geração de malha para diferentes geometrias

Com os *scripts* prontos, definiu-se algumas geometrias como forma de testar a versatilidade das definições feitas até aqui. A descrição da geometria de cada amostra está na Tabela 2 e os valores para os parâmetros foram escolhidos de forma a verificar a malha em diâmetros diferentes, 10" e 24" (qualquer valor entre esse diâmetros vão se comportar de forma semelhante), com schedule STD e XS possuindo defeitos com diferentes porcentagens de profundidade em relação à espessura do duto, verificando assim o comportamento quando há uma baixa espessura remanescente e quando há muita. Também variou-se o comprimento linear do defeito para diferentes comprimentos circunferenciais, ângulos. As amostras foram importantes para entender as aplicações do código e propor hipóteses para o comportamento em valores fora dos escolhidos, como será discutido na seção Resultados. Ambos os valores de diâmetro externo e espessura da parede foram retirados da ASME 36.10 (ASME, 2004), já para os raios de adoçamento, RC foi escolhido de forma que suavizasse o canto do defeito, por isso muda conforme o ângulo dele, e RA tentou-se usar uma proporção próxima de 0,5 em relação à profundidade do defeito que evita concentração de tensões (FERREIRA *et al.*, 2016).

Tabela 2 – Parâmetros geométricos das amostras

Nº	DO	T	D(%)	D	AF	L	RA	RC
1	273	9,53	75	7,1	5	50	2	3
2	273	12,7	25	3,2	5	50	1,6	3
3	273	12,7	50	6,4	5	50	2	3
4	273	12,7	75	9,5	5	50	2	3
5	273	12,7	25	3,2	15	50	1,6	8
6	273	12,7	50	6,4	15	50	3,2	8
7	273	12,7	75	9,5	15	50	4,8	8
8	273	12,7	25	3,2	25	50	1,6	15
9	273	12,7	50	6,4	25	50	3,2	15
10	273	12,7	75	9,5	25	50	4,8	15
11	610	12,7	25	3,2	5	50	1,6	3
12	610	12,7	50	6,4	5	50	2	3
13	610	12,7	75	9,5	5	50	2	3
14	610	12,7	75	9,5	15	50	4,8	8
15	610	12,7	75	9,5	25	50	4,8	15
16	610	12,7	75	9,5	5	150	2	3
17	610	12,7	75	9,5	15	150	4,8	8
18	610	12,7	75	9,5	25	150	4,8	15
19	610	12,7	75	9,5	5	250	2	3
20	610	12,7	75	9,5	15	250	4,8	8
21	610	12,7	75	9,5	25	250	4,8	15

Fonte: Autor (2024).

3.2.5 Conversor do arquivo da malha

Como citado na seção do desenvolvimento da malha, o arquivo final da malha é no formato .dat e o uso desse tipo de arquivo foi decidido a partir de testes feitos com a ferramenta de conversão de malha meshio (MESHIO, 2024), que consegue converter diversos formato de malha entre si. Dentre os testes, considerando as opções que o SALOME consegue exportar e as opções que o ANSYS consegue importar, o mais promissor foi a conversão de .dat para .inp, arquivo do ABAQUS (DASSAULT SYSTEMES, 2024), pois foi o que de fato funcionou quase 100%, fazendo com que a malha do SALOME fosse transferida para o ANSYS e ainda gerando uma geometria sólida, facilitando o preparo das análises, porém acabou não gerando os elementos piramidais presentes na malha.

Com isso, decidiu-se elaborar um *script* que modificasse o arquivo .dat para os moldes do .inp. O formato do arquivo .dat consiste em ter a primeira linha informando o número de nós e de elementos, nessa ordem, seguida de linhas com as informações dos nós, sendo a sequência das informações, separadas por espaço, o número do nó (se é o nó 1, 2, ...) e depois suas coordenadas, como visto na Figura 40a, e, com o fim dos nós, são listadas as informações dos elementos, com o número sequencial do elemento, seguido pelo tipo de elemento (o primeiro dígito é sobre a dimensão do elemento e o terceiro é a quantidade de nós, por exemplo 306 que indica um elemento 3D com 6 nós, um prisma triangular, ou 102 que indica um elemento 1D com 2 nós), pode-se ver a transição de nós para elementos na Figura 40b em que há o nó 298736 e em seguida os elementos 1 e 2.

Figura 40 – Formato das informações no arquivo .dat

```
298736 1487330
1 0.00000000000000e+00 -7.95942668090479e+00 -4.63211347757295e+01
2 0.00000000000000e+00 -6.13473103434165e+00 -4.65979084845692e+01
```

(a) Linha com o número de nós e elementos seguida das informações dos nós 1 e 2.

```
298736 1.71895833333333e+02 2.48545959000194e+00 4.84362724683317e+01
1 102 1 21
2 102 21 22
```

(b) Linha do nó 298736 seguida das informações dos elementos 1 e 2.

Fonte: Autor (2024).

Para o arquivo .inp, as informações precisam estar de forma diferente. Começando pelos nós, a primeira linha é composta por "*NODE", seguida pelas linhas com o número sequencial do nó e suas coordenadas, mas aqui as cada informação é separada por vírgula e um espaço, Figura 41a. Já nos elementos, antes de cada elemento, vem uma linha indicando o tipo do elemento, C3D8 para hexaedros, Figura 41b, C3D6 para primas de base triangular, Figura 41c, C3D5 para pirâmides e C3D4 para tetraedros, Figura 41d, com cada informação separada apenas por vírgula. Importante notar que, para o .inp, não há informação do elemento nas linhas e sim nessa linha antecedente às informações dos elementos e existe uma equivalência nessas informações, 308 é o elemento C3D8, 306 é o C3D6, 305 é o C3D5 e 304 é o C3D4.

Figura 41 – Formato das informações no arquivo .inp

```
*NODE
1, 0.00000000000000e+00, 0.00000000000000e+00, 2.21400000000000e+02
2, 0.00000000000000e+00, 0.00000000000000e+00, 2.24010000000000e+02
```

(a) Linha indicando o começo das informações dos nós e informações dos nós 1 e 2.

```
*ELEMENT, TYPE=C3D8
1,1255,1293,1416,1285,1278,1366,1426,1359
2,1278,1366,1426,1359,1277,1365,1427,1358
```

(b) Linha indicando o começo das informações dos hexaedros e elementos 1 e 2.

```
*ELEMENT, TYPE=C3D6
7113,1254,1286,2298,1264,1360,2340
7114,1264,1360,2340,1253,1272,2290
```

(c) Linha indicando o começo das informações dos prismas e elementos 7113 e 7114.

```
*ELEMENT, TYPE=C3D4
7305,693,92,345,93
7306,1094,241,1153,55
```

(d) Linha indicando o começo das informações dos tetraedros e elementos 7305 e 7306.

Fonte: Autor (2024).

Com isso, a ideia do *script* é ler a primeira informação sobre o número de nós e, assim, armazenar as linhas dos nós num *array*, depois lê as linhas dos elementos e filtra os elementos que contém os números 308, 306, 305 e 304 na segunda coluna, que são os elementos sólidos (descartando elementos 2D e 1D, pois não são necessários), e armazenando-os em diferentes *arrays*. Nos elementos prismáticos e tetraédricos, o ANSYS reconhece e converte para as formas prismática e tetraédrica do elemento SOLID185, porém isso não aconteceu para o elemento piramidal e, para contornar o problema, ele foi considerado como hexaedro e teve seu quinto nó repetido mais três vezes, Figura 42, a forma piramidal do SOLID185 (ANSYS, 2023), antes de ser armazenado no mesmo *array* dos hexaedros.

Figura 42 – Formato das informações dos elementos piramidais

```
6783,12412,10063,5690,10156,12413,10110,5691,10157
6784,12413,10110,5691,10157,10204,5644,5503,5692
6785,6,44,553,152,644,644,644,644
6786,152,553,554,153,645,645,645,645
```

(a) Elementos piramidais 6785 e 6786, com o quinto nó repetido, comparados com os hexaédricos 6783 e 6784.

Fonte: Autor (2024).

Assim, possuindo todas as informações armazenadas, adicionou-se as informações das linhas que precisam anteceder cada informação, como *NODE nos nós, "*ELEMENT, TYPE=C3D8" para os hexaedros e pirâmides, etc, e em seguida o arquivo .inp é escrito concatenando as informações, respeitando as separações (vírgulas e espaços) do tipo do arquivo. A estrutura das funções foi fornecida pela IA da OpenAI, ChatGPT, e ajustadas de acordo com as especificações do arquivo, principalmente na hora de concatenar que precisou definir como as informações seriam separadas (com vírgula, vírgula e espaço ou

sem separação, para as linhas já citadas que precedem cada informação diferente). O resultado final é apresentado na seção de resultados.

3.2.6 Análise estrutural

Como citado no fluxo de trabalho, foram feitas análises estruturais como forma de validar os códigos desenvolvidos. Nessas análises, dois modelos foram gerados com as ferramentas desenvolvidas e são baseados nas análises realizadas por Cabral (2007) e Tee e Wordu (2020), que também construíram modelos numéricos para análise da pressão de ruptura e compararam com dados experimentais. O trabalho de Cabral (2007) é baseado nos testes realizados por Benjamin *et al.* (2005) em trechos de dutos com defeitos usinados por eletroerosão, com o teste IDTS2 sendo a referência por possuir apenas um defeito. Outros trabalhos como Tee e Wordu (2020), Motta *et al.* (2017) e Soares *et al.* (2019) também utilizaram os experimentos de Benjamin *et al.* (2005) como meio de validar seus modelos computacionais, mas o modelo de Tee e Wordu (2020) que será utilizado aqui, ET5.1, é baseado no trabalho de Freire *et al.* (2006) em que, também, foram realizados testes laboratoriais em dutos com defeitos usinados até atingir a ruptura. Os modelos estudados serão denominados MD1 e MD2 respectivamente.

A primeira definição para ser possível a análise é das propriedades mecânicas dos materiais utilizados. Para o modelo 1, MD1, o material foi o API 5L X80 utilizado por Benjamin *et al.* (2005) e Andrade *et al.* (2006), suas propriedades mecânicas estão na Tabela 3 e sua relação tensão-deformação verdadeiros pode ser descrita pela seguinte equação de Ramberg-Osgood (ANDRADE *et al.*, 2006):

$$\epsilon_t = \frac{\sigma_t}{E} + 0,0788174 \left(\frac{\sigma_t}{\sigma_{tu}}\right)^{12,642026} \tag{54}$$

Em que: ϵ_t é a deformação verdadeira, σ_t é a tensão verdadeira, E é o módulo de Young do material e $\sigma_t u$ é a tensão última verdadeira do material. Como $\frac{\sigma_t}{E}$ é a definição da deformação elástica conforme a lei de Hooke, então a deformação plástica é dada pelo segundo termo do lado direito da equação. Possuindo essa relação, definiu-se pontos e os dados foram inseridos no ANSYS em forma de tabela, com o gráfico representando esse comportamento plástico na Figura 43.

De forma semelhante, para o MD2, o material é o API 5L X60, utilizado por Bedairi *et al.* (2012) e S. S. Al-Owaisi, Becker e W. Sun (2016), e as propriedades mecânicas estão na Tabela 3, também existe uma relação representativa para a relação entre deformação e tensão verdadeira (BEDAIRI *et al.*, 2012):

$$\epsilon_t = \frac{\sigma_t}{E} + \alpha \left(\frac{\sigma_t}{E}\right) \left(\frac{\sigma_t}{\sigma_y}\right)^{n-1} \tag{55}$$

Com σ_y sendo a tensão de escoamento do material, α igual a 1,75 e n igual a 9,35. Novamente, a deformação plástica é representada pelo segundo termo do lado direito da

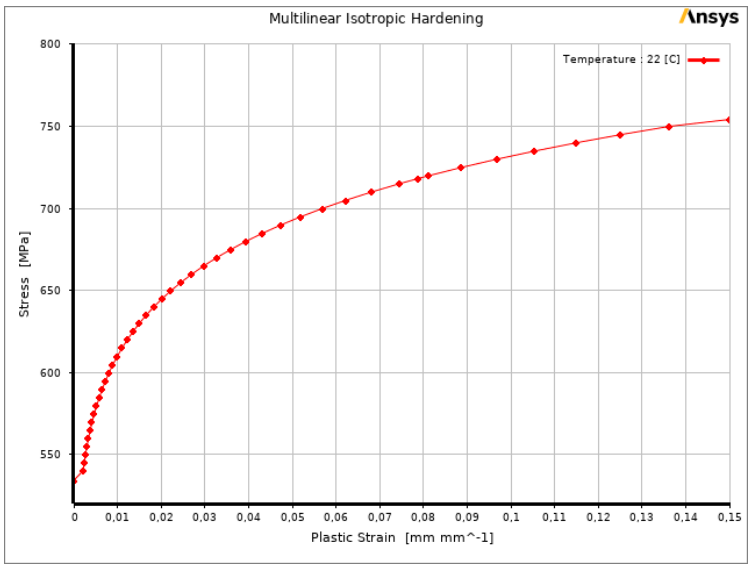
equação e foi utilizado para gerar pontos para a inserção no ANSYS, gráfico dos pontos na Figura 44.

Tabela 3 – Propriedades mecânicas dos materiais utilizados

Modelo	Mat.	E	σ_y	σ_{ut}	υ
MD1	X80	200 GPa	534,1 MPa	718,2 MPa	0,3
MD2	X60	207 GPa	435,0 MPa	631,0 MPa	0,3*

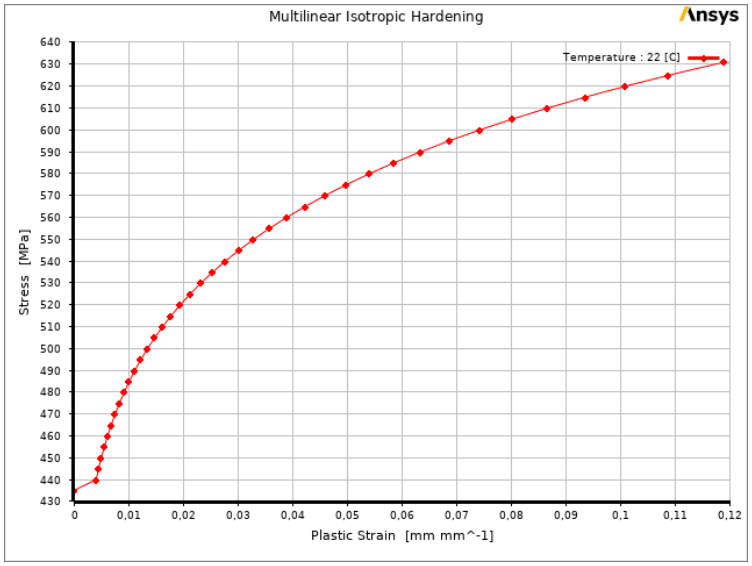
Fonte: Autor (2024).

Figura 43 – Curva de tensão verdadeira por deformação plástica para o X80



Fonte: Autor (2024).

Figura 44 – Curva de tensão verdadeira por deformação plástica para o X60



Fonte: Autor (2024).

Importante perceber que, em ambas curvas de tensão verdadeira por deformação plástica, assume-se zero deformação plástica no ponto de escoamento, gerando uma região próxima ao escoamento com comportamento diferente ao descrito pelas equações, mas essa hipótese é necessária para os dados poderem ser inseridos no ANSYS e não afeta a análise, dado que a pressão de ruptura causa tensões maiores que a tensão de escoamento e, assim, o estudo ocorre longe dessa região.

Em seguida, definiu-se as geometrias para serem utilizadas no *script*, sendo cada uma conforme os trabalhos referenciados Cabral (2007) e Tee e Wordu (2020). Para o modelo MD2, RC e RA tiveram que ser estimados baseando-se nas imagens presentes no trabalho, pois não são mencionados. Os parâmetros utilizados estão na Tabela 4.

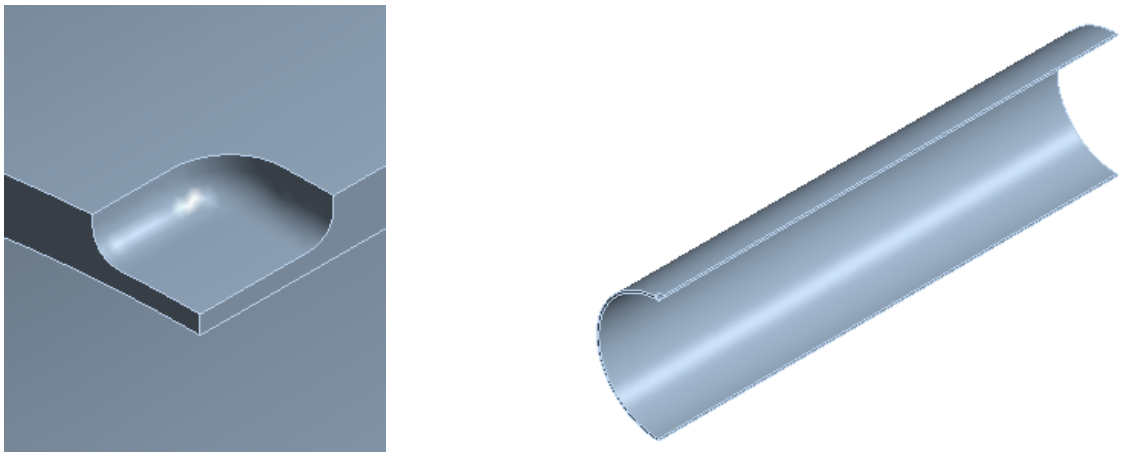
Tabela 4 – Parâmetros geométricos dos modelos

Modelo	DO	T	D	L	AF	RA	RC	L_{pipe}
MD1	458,8	8,1	5,39	39,6	8	3.5	8	1700
MD2	323,9	9,5	6,67	256	33,7	1	5	2300

Fonte: Autor (2024).

Sendo o parâmetro L_{pipe} o comprimento do duto analisado. O comprimento automático descrito na seção de geometria não foi utilizado aqui para poder deixar os modelos iguais aos utilizados nos trabalhos. As geometrias finais no ANSYS estão nas Figuras 45 e 46.

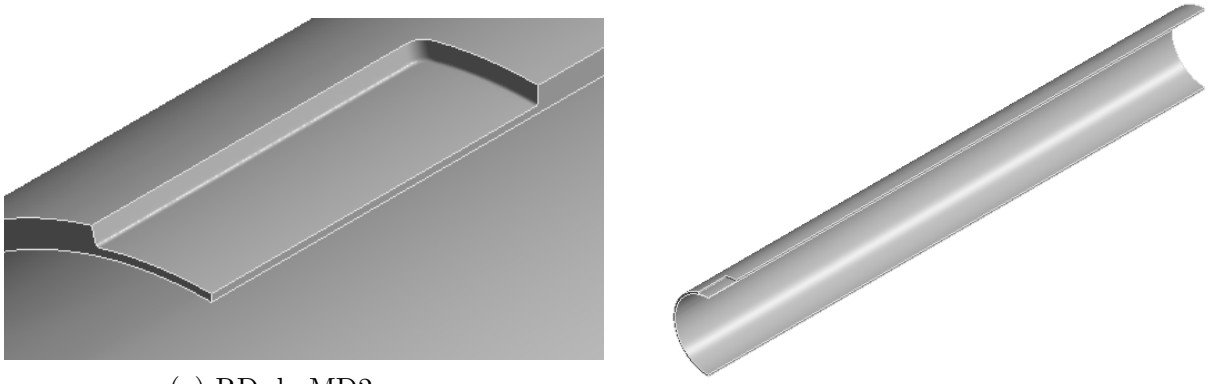
Figura 45 – Modelo 1 - MD1



(a) RD do MD1. (b) Duto completo do MD1.

Fonte: Autor (2024).

Figura 46 – Modelo 2 - MD2



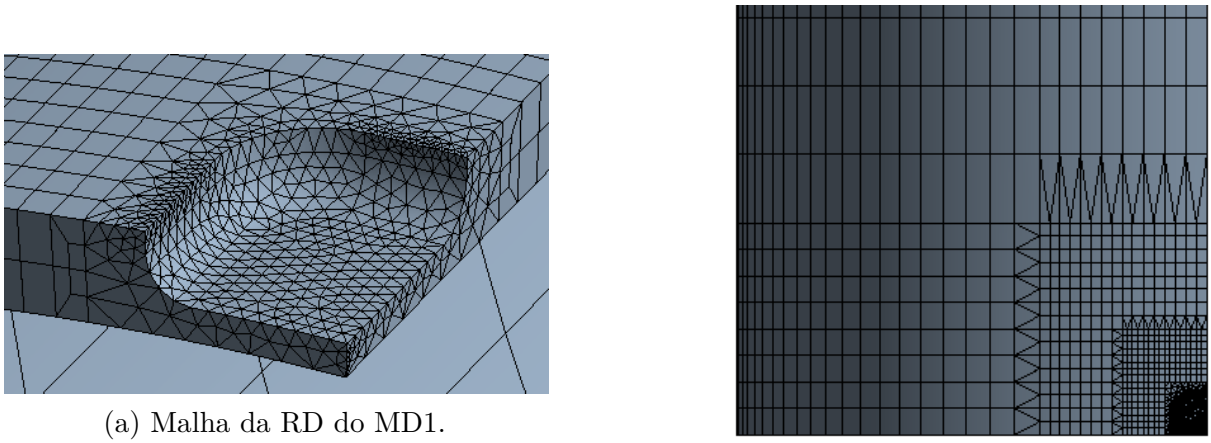
(a) RD do MD2.

(b) Duto completo do MD2.

Fonte: Autor (2024).

Juntamente com a geração da geometria, a malha gerada para cada modelo está nas Figuras 47 e 48. A malha do MD1 possui 12395 nós e 14458 elementos e do MD2 possui 19520 nós e 34989 elementos, com as estastísticas do *skewness* de cada malha nas Figuras 49 e 50, em que o *skewness* indica a variação do ângulos dos elementos da malha em relação aos elementos ideais, com 0 sendo o elemento ideal e 1 sendo um elemento degenerado (SIMSCALE, 2024). Com isso, percebe-se que a malha do MD2 possui diversos elementos muito distorcidos, pois o defeito é muito longo e acaba causando comportamentos indesejados de malha, como é comentado na seção de resultados. Já para a malha do MD1, poucos elementos estão totalmente degenerados, sendo a maior parte os elementos piramidais que foram modelados como hexaedros com o quinto nó repetido mais três vezes, por isso o programa os considera com qualidade ruim, sendo, assim, uma boa malha.

Figura 47 – Malha do MD1

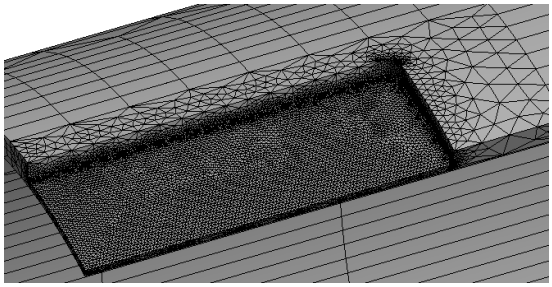


(a) Malha da RD do MD1.

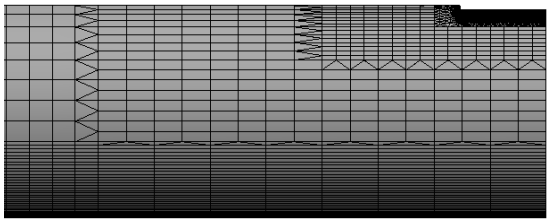
(b) Malha do duto do MD1.

Fonte: Autor (2024).

Figura 48 – Malha do MD2



(a) Malha da RD do MD2.



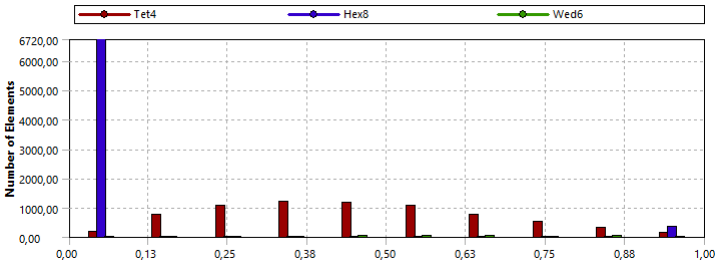
(b) Malha do duto do MD2.

Fonte: Autor (2024).

Figura 49 – Estatísticas da malha do MD1

Mesh Metric	Skewness
Min	1,8056e-003
Max	1,
Average	0,26082
Standard Deviation	0,28495

(a) Estatísticas do *skewness* da malha do MD1.



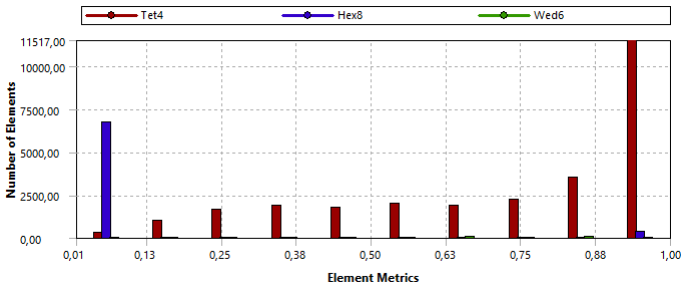
(b) Distribuição dos elementos de MD1 de acordo com o *skewness*

Fonte: Autor (2024).

Figura 50 – Estatísticas da malha do MD2

Mesh Metric	Skewness
Min	6,5748e-003
Max	1,
Average	0,59502
Standar...	0,37182

(a) Estatísticas do *skewness* da malha do MD2.

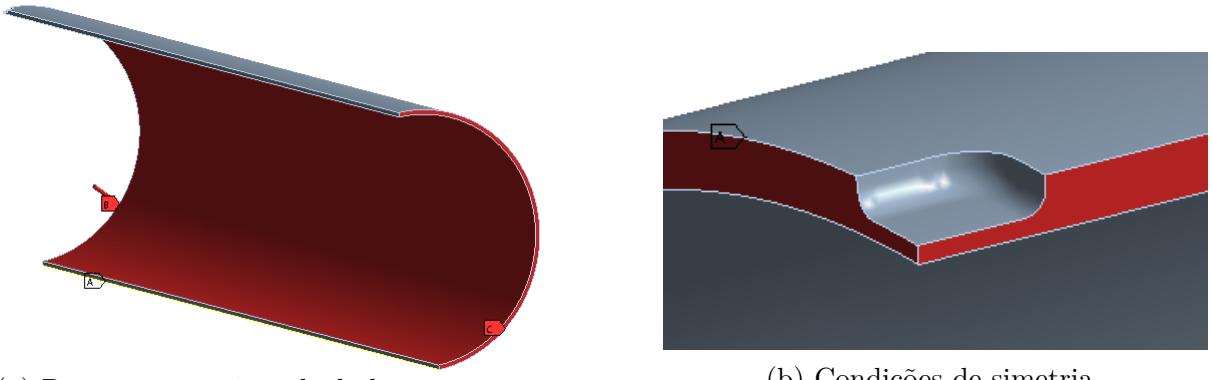


(b) Distribuição dos elementos de MD2 de acordo com o *skewness*

Fonte: Autor (2024).

Com isso, os modelos numéricos estão definidos e a análise deve ser configurada. Primeiramente, para o MD1, as condições de contorno são definidas por Cabral (2007), sendo uma pressão interna aplicada na superfície interna do duto, uma carga de tração devido ao tampo de vedação utilizado durante o teste, definida aqui como uma pressão na face do duto oposta ao defeito, condição de simetria nas faces dos planos de simetria e uma restrição de deslocamento vertical, em Z, na aresta inferior do duto, para evitar movimentos de corpo rígido, todas essas condições podem ser observadas na Figura 51. A análise de ruptura leva o material para o regime plástico que faz com que seja preciso ativar a condição *large deflection*, para computar as não linearidades geométricas, fazendo a análise ser iterativa. O critério de convergência para força foi definido como 0,001 e o resto das convergências foram conforme o programa. Além disso, as pressões foram aplicadas com diversos incrementos no trabalho de Cabral (2007) de forma a descobrir a pressão de ruptura, mas como a intenção desta análise é apenas verificar a qualidade dos modelos gerados pelo código desenvolvido, então utilizou-se do valor encontrado por Cabral (2007) como ponto de partida para mapear a pressão que causa a ruptura. Sendo assim, as pressões foram aplicadas em três passos e o valor inicial é 22 MPa, aplicada em um único sub-passo (que não causa a ruptura, segundo o trabalho base), depois 22,5 MPa aplicada do passo 1 para o 2 em 50 sub-passos, e, por fim, 23 MPa, novamente em 50 sub-passos do passo 2 para o passo 3. Os resultados serão mostrados e discutidos na seção de resultados.

Figura 51 – Condições de contorno do MD1



(a) Pressões e restrição de deslocamento em Z.

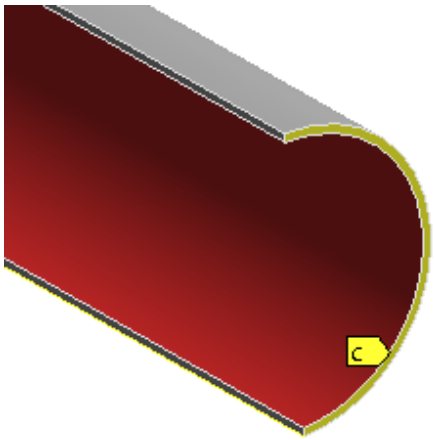
(b) Condições de simetria.

Fonte: Autor (2024).

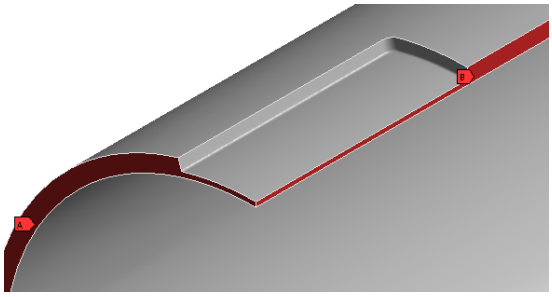
Para o MD2, as condições e configurações foram semelhantes, mas não utilizou-se da pressão de tração devido aos tampos, o trabalho Tee e Wordu (2020) definiu nessa mesma região um suporte “fixo”, interpretado aqui como uma restrição aos deslocamentos lineares nas três direções. Também não foi configurado um valor para a convergência da força, deixando como *program controlled*. A pressão interna foi aplicada de forma similar ao MD1, em três passos e com valores próximos ao valor de ruptura obtido no trabalho de

Tee e Wordu (2020), iniciando com 13 MPa, o segundo passo com 13,5 MPa e o terceiro 14 MPa. A Figura 52 mostra as condições de contorno do MD2 no ANSYS.

Figura 52 – Condições de contorno do MD2



(a) Pressão interna e restrições de deslocamento.



(b) Condições de simetria.

Fonte: Autor (2024).

4 RESULTADOS E DISCUSSÕES

4.1 *SCRIPT* DE GERAÇÃO DO MODELO COMPUTACIONAL

Nas seções anteriores, foi descrito o processo de desenvolvimento do *script* para a geração automática do modelo computacional, que é o principal objetivo deste trabalho. Nessa seção, será apresentado o código final para defeitos externos com descrição das seções, mas sem entrar em detalhe em relação à estrutura das funções e comando que são particulares ao SALOME, além de serem gerados de forma automática por ele. Apesar do código ser gerado automaticamente, foram precisos pequenos ajustes para deixá-lo 100% funcional e para organizá-lo melhor, por exemplo modificar ou adicionar parâmetros de malha diretamente no código sem utilizar o SALOME. O código para defeitos internos é muito semelhante e, por isso, não será apresentado.

Assim, o código começa importando módulos e com a definição do *notebook* em que há o parâmetro RCC já mencionado, onde o 3,14 é um valor aproximado de π e o 0,1 é o valor de RA, que deve ser modificado manualmente quando a geometria muda, Figura 53.

Figura 53 – Início do código e definição do parâmetro RCC

```
#!/usr/bin/env python

###
### This file is generated automatically by SALOME v9.9.0 with dump python functionality
###

import sys
import salome

salome.salome_init()
import salome_notebook
notebook = salome_notebook.NoteBook()
sys.path.insert(0, r'C:/Users/heito/OneDrive/Documentos/UFPE/TCC1/SCRIPTS')

##################################################
##       Begin of NoteBook variables section      ##
##################################################
notebook.set("RCC", "3.14*0.1/8")
##################################################
##        End of NoteBook variables section       ##
##################################################
###
### SHAPER component
###
```

Fonte: Autor (2024).

Em seguida, o código “entra” no módulo SHAPER , cria a *part* (numa montagem, pode-se criar várias partes, mas aqui todas as regiões foram modeladas numa única *part*) e cria os parâmetros, que são alterados manualmente para cada geometria, Figura 54.

Figura 54 – Definição dos parâmetros

```
from SketchAPI import *

from salome.shaper import model

model.begin()
partSet = model.moduleDocument()

### Create Part
Part_1 = model.addPart(partSet)
Part_1_doc = Part_1.document()
model.addParameter(Part_1_doc, "DO", '100')
model.addParameter(Part_1_doc, "T", '3')
model.addParameter(Part_1_doc, "RO", 'DO/2')
model.addParameter(Part_1_doc, "RI", 'RO-T')
model.addParameter(Part_1_doc, "D", '1')
model.addParameter(Part_1_doc, "AF", '15')
model.addParameter(Part_1_doc, "L", '10')
model.addParameter(Part_1_doc, "TH", 'AF*1.3/2')
model.addParameter(Part_1_doc, "C", 'L*1.3/2')
model.addParameter(Part_1_doc, "RA", '0.1')
model.addParameter(Part_1_doc, "RC", '1')
```

Fonte: Autor (2024).

Depois vem a construção das geometrias, iniciando pela definição do esboço no plano YOZ, seguida pela criação de retas e arcos com cotas, *constraints*, definindo as dimensões de acordo com os parâmetros e, por fim, extrudando a região entre os arcos e retas, já descritas de forma visual anteriormente. As Figuras 55, 56 e 57 ilustram essas etapas para a modelagem do defeito, geometria equivalente à perda de material do duto e que será utilizada para remover material da RD, mas as outras geometrias são modeladas de forma similar.

Figura 55 – Definições do primeiro esboço e suas geometrias

```
### Create Sketch
Sketch_1 = model.addSketch(Part_1_doc, model.defaultPlane("YOZ"))

### Create SketchProjection
SketchProjection_1 = Sketch_1.addProjection(model.selection("VERTEX", "PartSet/Origin"), False)
SketchPoint_1 = SketchProjection_1.createdFeature()

### Create SketchLine
SketchLine_1 = Sketch_1.addLine(0, 0, 0, 50)
Sketch_1.setCoincident(SketchAPI_Point(SketchPoint_1).coordinates(), SketchLine_1.startPoint())
Sketch_1.setVertical(SketchLine_1.result())

### Create SketchLine
SketchLine_2 = Sketch_1.addLine(0, 0, -6.526309611002588, 49.57224306869055)
Sketch_1.setCoincident(SketchAPI_Point(SketchPoint_1).coordinates(), SketchLine_2.startPoint())

### Create SketchConstraintAngle
Sketch_1.setAngle(SketchLine_2.result(), SketchLine_1.result(), "AF/2", type = "Direct")

### Create SketchArc
SketchArc_1 = Sketch_1.addArc(0, 0, 0, 50, -6.526309611002588, 49.57224306869055, False)
Sketch_1.setCoincident(SketchPoint_1.result(), SketchArc_1.center())
Sketch_1.setRadius(SketchArc_1.results()[1], "RO")
Sketch_1.setCoincident(SketchLine_2.endPoint(), SketchArc_1.endPoint())
Sketch_1.setCoincident(SketchLine_1.endPoint(), SketchArc_1.startPoint())
```

Fonte: Autor (2024).

Figura 56 – Finalização da definição das geometrias do esboço 1 e extrusão do defeito

```
74  ### Create SketchArc
75  SketchArc_2 = Sketch_1.addArc(0, 0, 1.200153863164406e-14, 49, -6.395783418782954, 48.58079820731665, False)
76  Sketch_1.setCoincident(SketchAPI_Point(SketchPoint_1).coordinates(), SketchArc_2.center())
77  Sketch_1.setRadius(SketchArc_2.results()[1], "RO-D")
78  Sketch_1.setCoincident(SketchArc_2.endPoint(), SketchLine_2.result())
79  Sketch_1.setCoincident(SketchArc_2.startPoint(), SketchLine_1.result())
80  model.do()
81
82  ### Create Extrusion
83  Extrusion_1 = model.addExtrusion(Part_1_doc, [model.selection("FACE", "Sketch_1/Face-SketchLine_1f-SketchArc_1
```

Fonte: Autor (2024).

Figura 57 – Continuação do comando de extrusão

```
82
83  l/Face-SketchLine_1f-SketchArc_1_2f-SketchLine_2r-SketchArc_2_2r")], model.selection(), "-L/2", 0, "Faces|Wires")
```

Fonte: Autor (2024).

Com as cinco geometrias modeladas (defeito, D, RT1, RT2 e RT3) modeladas, faz-se os raios de adoçamento do defeito, RC e RA, “Fillet_1” e “Fillet_2” respectivamente, Figura 58 e Figura 59.

Figura 58 – Criação dos raios de adoçamento do defeito RC e RA

```
231  ### Create Fillet
232  Fillet_1 = model.addFillet(Part_1_doc, [model.selection("EDGE", "[Extrusion_1_1/Generate
233
234  ### Create Fillet
235  Fillet_2 = model.addFillet(Part_1_doc, [model.selection("EDGE", "[Fillet_1_1/MF:Fillet&S
```

Fonte: Autor (2024).

Figura 59 – Continuação do comando dos raios de adoçamento

```
232  _1/Generated_Face&Sketch_1/SketchLine_2][Extrusion_1_1/To_Face]")], "RC", keepSubResults = True)
233
234
235  MF:Fillet&Sketch_1/SketchLine_2][Fillet_1_1/MF:Fillet&Sketch_1/SketchArc_2_2]")], "RA", keepSubResults = True)
```

Fonte: Autor (2024).

Em seguida é a definição dos planos que recortam as regiões RT1, RT2 e RT3, Figura 60. Os recortes são feitos, em que a RD tem material removido equivalente à geometria do defeito e as RT’s são divididas pelos planos anteriores, Figuras 61 e 62, e, depois, são removidas as geometrias desnecessárias, o defeito, que já cumpriu o papel de remover material da RD, e as regiões que se sobrepõem das RT’s, Figura 63.

Figura 60 – Criação dos planos que recortam as RT's

```
### Create Plane
Plane_4 = model.addPlane(Part_1_doc, model.selection("FACE", "Extrusion_2_1/Generated_Face&Sketch_2/SketchLine_4"), 0, False)

### Create Plane
Plane_5 = model.addPlane(Part_1_doc, model.selection("FACE", "Extrusion_2_1/To_Face"), 0, False)

### Create Plane
Plane_6 = model.addPlane(Part_1_doc, model.selection("FACE", "Extrusion_3_1/Generated_Face&Sketch_3/SketchLine_6"), 0, False)

### Create Plane
Plane_7 = model.addPlane(Part_1_doc, model.selection("FACE", "Extrusion_3_1/To_Face"), 0, False)

### Create Plane
Plane_8 = model.addPlane(Part_1_doc, model.selection("FACE", "Extrusion_4_1/Generated_Face&Sketch_4/SketchLine_8"), 0, False)

### Create Plane
Plane_9 = model.addPlane(Part_1_doc, model.selection("FACE", "Extrusion_4_1/To_Face"), 0, False)
```

Fonte: Autor (2024).

Figura 61 – Comando de *split* para dividir as geometrias

```
### Create Split
Split_1 = model.addSplit(Part_1_doc, [model.selection("SOLID", "Extrusion_2_1")], [mode

### Create Split
Split_2 = model.addSplit(Part_1_doc, [model.selection("SOLID", "Extrusion_3_1")], [mode

### Create Split
Split_3 = model.addSplit(Part_1_doc, [model.selection("SOLID", "Extrusion_4_1")], [mode

### Create Split
Split_4 = model.addSplit(Part_1_doc, [model.selection("SOLID", "Extrusion_5_1")], [mode
```

Fonte: Autor (2024).

Figura 62 – Continuação dos comandos de divisão

```
ion_2_1")], [model.selection("SOLID", "Fillet_2_1")], keepSubResults = True)


ion_3_1")], [model.selection("FACE", "Plane_1"), model.selection("FACE", "Plane_2")], keepSubResults = True)


ion_4_1")], [model.selection("FACE", "Plane_3"), model.selection("FACE", "Plane_4")], keepSubResults = True)


ion_5_1")], [model.selection("FACE", "Plane_5"), model.selection("FACE", "Plane_6")], keepSubResults = True)
```

Fonte: Autor (2024).

Figura 63 – Remoção das geometrias desnecessárias

```
### Create Remove_SubShapes
Remove_SubShapes_1 = model.addRemoveSubShapes(Part_1_doc, model.selection("COMPSOLID", "Split_1_1"))
Remove_SubShapes_1.setSubShapesToRemove([model.selection("SOLID", "Split_1_1_2")])

### Create Remove_SubShapes
Remove_SubShapes_2 = model.addRemoveSubShapes(Part_1_doc, model.selection("COMPSOLID", "Split_2_1"))
Remove_SubShapes_2.setSubShapesToRemove([model.selection("SOLID", "Split_2_1_4")])

### Create Remove_SubShapes
Remove_SubShapes_3 = model.addRemoveSubShapes(Part_1_doc, model.selection("COMPSOLID", "Split_3_1"))
Remove_SubShapes_3.setSubShapesToRemove([model.selection("SOLID", "Split_3_1_4")])

### Create Remove_SubShapes
Remove_SubShapes_4 = model.addRemoveSubShapes(Part_1_doc, model.selection("COMPSOLID", "Split_4_1"))
Remove_SubShapes_4.setSubShapesToRemove([model.selection("SOLID", "Split_4_1_4")])
```

Fonte: Autor (2024).

Assim, a geometria está finalizada, pode ser visualizada na Figura 27a, e começam as definições dos "grupos" que serão utilizados para associar as definições da malha, Figura

64. Na Figura 65, tem-se a definição do último grupo e a finalização da parte geométrica. Em seguida, o SALOME define um estudo do SHAPER com as geometrias e grupos definidos anteriormente, Figura 66 e Figura 67.

Figura 64 – Definição dos grupos

```
### Create Group
Group_1 = model.addGroup(Part_1_doc, "Solids", [model.selection("SOLID", "Remove_SubShapes_1_1")])
Group_1.setName("DEFECT")
Group_1.result().setName("DEFECT")

### Create Group
Group_2 = model.addGroup(Part_1_doc, "Solids", [model.selection("SOLID", "Remove_SubShapes_2_1_3")])
Group_2.setName("CIRCULAR1")
Group_2.result().setName("CIRCULAR1")

### Create Group
Group_3 = model.addGroup(Part_1_doc, "Solids", [model.selection("SOLID", "Remove_SubShapes_2_1_2")])
Group_3.setName("LONGI1")
Group_3.result().setName("LONGI1")
```

Fonte: Autor (2024).

Figura 65 – Último grupo e fechamento do módulo da geometria

```
### Create Group
Group_45_objects = [model.selection("EDGE", "Remove_SubShapes_4_1_2/Modified_
                    model.selection("EDGE", "Remove_SubShapes_4_1_2/Modified_
                    model.selection("EDGE", "[Remove_SubShapes_4_1_2/Modifie
                    model.selection("EDGE", "[Remove_SubShapes_4_1_2/Modifie
Group_45 = model.addGroup(Part_1_doc, "Edges", Group_45_objects)
Group_45.setName("LONG3LONGI")
Group_45.result().setName("LONG3LONGI")

model.end()
```

Fonte: Autor (2024).

Figura 66 – Estudo do SHAPER

```
model.publishToShaperStudy()
import SHAPERSTUDY
# This shape does not exist among the SHAPER results; if it is referenced by SMESH, this may cause an error
# Extrusion_1_1, = SHAPERSTUDY.shape("dead01_8:54")
# This shape does not exist among the SHAPER results; if it is referenced by SMESH, this may cause an error
# Extrusion_2_1, = SHAPERSTUDY.shape("dead02_8:112")
# This shape does not exist among the SHAPER results; if it is referenced by SMESH, this may cause an error
# Extrusion_3_1, = SHAPERSTUDY.shape("dead03_8:154")
# This shape does not exist among the SHAPER results; if it is referenced by SMESH, this may cause an error
# Extrusion_4_1, = SHAPERSTUDY.shape("dead04_8:295")
# This shape does not exist among the SHAPER results; if it is referenced by SMESH, this may cause an error
# Extrusion_5_1, = SHAPERSTUDY.shape("dead05_8:336")
# This shape does not exist among the SHAPER results; if it is referenced by SMESH, this may cause an error
# Fillet_2_1, = SHAPERSTUDY.shape("dead06_8:338")
Remove_SubShapes_1_1, DEFECT, DEFECTFILLETRAD, DEFECTFILLETLONG, DEFECTFILLETCIRC, DEFECTFILLETCORNER, DEFECTCORNER, DEFECTRADPE
Remove_SubShapes_2_1, CIRCULAR1, LONGI1, CORNER1, CIRC1RADINT, CIRC1RADEXT, LONGI1RADINT, LONGI1RADEXT, CORNER1RADINT, CORNER1RA
Remove_SubShapes_3_1, CIRCULAR2, LONGI2, CORNER2, CIRC2RAD, CIRC2LONGINT, LONGI2RAD, LONGI2CIRCINT, CORNER2RAD, = SHAPERSTUDY.sh
Remove_SubShapes_4_1, CIRCULAR3, LONGI3, CORNER3, CIRC3RAD, CIRC3CIRC, CIRC3LONGEXT, LONG3RAD, LONG3CIRCEXT, CORNER3RAD, CORNER3
```

Fonte: Autor (2024).

Figura 67 – Continuação da definição do estudo do SHAPER

```
ER, DEFECTCORNER, DEFECTRADPEQ, DEFECTRADMED, DEFECTRADG, DEFECTLONGIINT, DEFECTCIRCINT, DEFECTLONGICENTER, DEF
EXT, CORNER1RADINT, CORNER1RADEXT, = SHAPERSTUDY.shape(model.featureStringId(Remove_SubShapes_2))
 CORNER2RAD, = SHAPERSTUDY.shape(model.featureStringId(Remove_SubShapes_3))
3CIRCEXT, CORNER3RAD, CORNER3CIRC, LONG3LONGI, = SHAPERSTUDY.shape(model.featureStringId(Remove_SubShapes_4))
```

Fonte: Autor (2024).

Assim, inicia-se o módulo MESH e a definição das malhas, algoritmos e hipóteses. As Figuras 68 e 69 mostram a definição da malha da RD, nomeada como "DEFECT" no *software*. Todas as outras malhas seguem estruturas similares. Essa seção apresentou problemas no código durante o desenvolvimento e as soluções envolveram deixar na estrutura apresentada nas Figuras abaixo.

Figura 68 – Início do módulo MESH e da definição da malha da RD

```
import  SMESH, SALOMEDS
from salome.smesh import smeshBuilder

smesh = smeshBuilder.New()
#smesh.SetEnablePublish( False ) # Set to False to avoid publish in study if not needed or in some particular situations:
                                 # multiples meshes built in parallel, complex and numerous mesh edition (performance)

DEFECT_1 = smesh.Mesh(DEFECT,'DEFECT')
Regular_1D = DEFECT_1.Segment()
a8 = Regular_1D.NumberOfSegments(8)
NETGEN_2D = DEFECT_1.Triangle(algo=smeshBuilder.NETGEN_2D)
NETGEN_3D = DEFECT_1.Tetrahedron()
NETGEN_3D_Parameters_1 = NETGEN_3D.Parameters()
NETGEN_3D_Parameters_1.SetMaxSize( 1.12908 )
NETGEN_3D_Parameters_1.SetMinSize( "RCC" )
NETGEN_3D_Parameters_1.SetOptimize( 1 )
NETGEN_3D_Parameters_1.SetFineness( 2 )
NETGEN_3D_Parameters_1.SetCheckOverlapping( 0 )
NETGEN_3D_Parameters_1.SetElemSizeWeight( 1.31043e-311 )
NETGEN_3D_Parameters_1.SetCheckChartBoundary( 192 )
```

Fonte: Autor (2024).

Figura 69 – Finalização da malha da RD

```
Regular_1D_1 = DEFECT_1.Segment(geom=DEFECTFILLETRAD)
a4 = Regular_1D_1.NumberOfSegments(4)
Regular_1D_2 = DEFECT_1.Segment(geom=DEFECTFILLETLONG)
FILLET = Regular_1D_2.LocalLength("RCC",None,1e-07)
Regular_1D_3 = DEFECT_1.Segment(geom=DEFECTFILLETCIRC)
status = DEFECT_1.AddHypothesis(FILLET,DEFECTFILLETCIRC)
Regular_1D_4 = DEFECT_1.Segment(geom=DEFECTFILLETCORNER)
status = DEFECT_1.AddHypothesis(FILLET,DEFECTFILLETCORNER)
Regular_1D_5 = DEFECT_1.Segment(geom=DEFECTCORNER)
a12 = Regular_1D_5.NumberOfSegments(12)
Regular_1D_6 = DEFECT_1.Segment(geom=DEFECTRADPEQ)
RAD = Regular_1D_6.LocalLength("RCC",None,1e-07)
Regular_1D_7 = DEFECT_1.Segment(geom=DEFECTRADMED)
status = DEFECT_1.AddHypothesis(a4,DEFECTRADMED)
Regular_1D_8 = DEFECT_1.Segment(geom=DEFECTRADG)
Regular_1D_9 = DEFECT_1.Segment(geom=DEFECTLONGIINT)
a16 = Regular_1D_9.NumberOfSegments(16)
Regular_1D_10 = DEFECT_1.Segment(geom=DEFECTCIRCINT)
status = DEFECT_1.AddHypothesis(a16,DEFECTCIRCINT)
Regular_1D_11 = DEFECT_1.Segment(geom=DEFECTLONGICENTER)
status = DEFECT_1.AddHypothesis(FILLET,DEFECTLONGICENTER)
Regular_1D_13 = DEFECT_1.Segment(geom=DEFECTCIRCCENTER)
status = DEFECT_1.AddHypothesis(FILLET,DEFECTCIRCCENTER)
Regular_1D_14 = DEFECT_1.Segment(geom=DEFECTLONGIPEQ)
status = DEFECT_1.AddHypothesis(a4,DEFECTLONGIPEQ)
Regular_1D_15 = DEFECT_1.Segment(geom=DEFECTCIRCPEQ)
status = DEFECT_1.AddHypothesis(a4,DEFECTCIRCPEQ)
Quadrangle_2D = DEFECT_1.Quadrangle(algo=smeshBuilder.QUADRANGLE,geom=DEFECTHEXFACES)
status = DEFECT_1.AddHypothesis(a4,DEFECTRADG)
isDone = DEFECT_1.Compute()
```

Fonte: Autor (2024).

Por fim, com todas as malhas definidas, para o código da geometria de um quarto do duto, concatena-se as malhas, executa mais alguns comandos do *software*, Figura 70, executa-se comandos de nomenclatura, Figura 71, e, por fim, as linhas de comando para a malha concatenada ser exportada no formato .dat é executado e finaliza-se o código, Figura 72. Já para o arquivo do duto completo, há a etapa de espelhar as malhas em relação aos planos de simetria antes de concatená-las, Figura 73.

Figura 70 – Junção das malhas numa única malha e comandos para as malhas 1D

```
744  Compound_Mesh_1 = smesh.Concatenate( [ DEFECT_1.GetMesh(), CIRCULAR1_1.GetMesh(), LO
745  DEFECTFILLETRAD_1 = Regular_1D_1.GetSubMesh()
746  DEFECTFILLETLONG_1 = Regular_1D_2.GetSubMesh()
747  DEFECTFILLETCIRC_1 = Regular_1D_3.GetSubMesh()
748  DEFECTFILLETCORNER_1 = Regular_1D_4.GetSubMesh()
749  DEFECTCORNER_1 = Regular_1D_5.GetSubMesh()
750  DEFECTRADPEQ_1 = Regular_1D_6.GetSubMesh()
751  DEFECTRADMED_1 = Regular_1D_7.GetSubMesh()
752  DEFECTRADG_1 = Regular_1D_8.GetSubMesh()
753  DEFECTLONGIINT_1 = Regular_1D_9.GetSubMesh()
754  DEFECTCIRCINT_1 = Regular_1D_10.GetSubMesh()
755  DEFECTLONGICENTER_1 = Regular_1D_11.GetSubMesh()
756  DEFECTCIRCCENTER_1 = Regular_1D_13.GetSubMesh()
```

Fonte: Autor (2024).

Figura 71 – Comandos de nomenclatura

```
781  ## Set names of Mesh objects
782  smesh.SetName(DEFECTRADG_1, 'DEFECTRADG')
783  smesh.SetName(DEFECTLONGIINT_1, 'DEFECTLONGIINT')
784  smesh.SetName(DEFECTFILLETLONG_1, 'DEFECTFILLETLONG')
785  smesh.SetName(DEFECTFILLETCIRC_1, 'DEFECTFILLETCIRC')
786  smesh.SetName(DEFECTFILLETRAD_1, 'DEFECTFILLETRAD')
787  smesh.SetName(DEFECTRADPEQ_1, 'DEFECTRADPEQ')
788  smesh.SetName(DEFECTRADMED_1, 'DEFECTRADMED')
789  smesh.SetName(DEFECTFILLETCORNER_1, 'DEFECTFILLETCORNER')
790  smesh.SetName(DEFECTCORNER_1, 'DEFECTCORNER')
791  smesh.SetName(CORNER2RAD_1, 'CORNER2RAD')
792  smesh.SetName(FILLET, 'FILLET')
```

Fonte: Autor (2024).

Figura 72 – Comando para exportar a malha e finalização do código

```
842  try:
843      Compound_Mesh_1.ExportDAT( r'C:/Users/heito/OneDrive/Documentos/UFPE/TCC1/SCRIPTS/MESH.dat' )
844      pass
845  except:
846      print('ExportDAT() failed. Invalid file name?')
847
848  if salome.sg.hasDesktop():
849      salome.sg.updateObjBrowser()
```

Fonte: Autor (2024).

Figura 73 – Linhas do código para espelhar as malhas

```
DEFECT_mirrored1 = DEFECT_1.MirrorObjectMakeMesh( DEFECT_1, SMESH.AxisStruct( 0, 0, 0, 0, 1, 0 ), SMESH.SMESH_MeshEditor.PLANE, 0, 'DEFECT_mirror
CIRCULAR1_mirrored1 = CIRCULAR1_1.MirrorObjectMakeMesh( CIRCULAR1_1, SMESH.AxisStruct( 0, 0, 0, 0, 1, 0 ), SMESH.SMESH_MeshEditor.PLANE, 0, 'CIRC
LONGI1_mirrored1 = LONGI1_1.MirrorObjectMakeMesh( LONGI1_1, SMESH.AxisStruct( 0, 0, 0, 0, 1, 0 ), SMESH.SMESH_MeshEditor.PLANE, 0, 'LONGI1_mirror
```

Fonte: Autor (2024).

4.2 *SCRIPT* PARA CONVERSÃO DA MALHA EM ARQUIVO .DAT PARA .INP

Para a comunicação do *software* CAD e CAE, precisou-se do código desenvolvido de conversão do arquivo .dat para o arquivo .inp, como já descrito. Nesta seção, as funções e linhas de comando serão ilustradas de forma a entender como as necessidades da conversão foram supridas. Serão descritas primeiro as linhas de comando do corpo do código e, em seguida, as funções.

Inicialmente, o módulo os é importado para renomear o arquivo mesh.dat para mesh.txt, a função de leitura da primeira linha é chamada e o número de nós e de elementos são armazenados em nb_no e nb_elem, respectivamente, Figura 74.

Figura 74 – Mudança no arquivo e obtenção do número de nós e de elementos

```
89  import os
90
91  src = 'MESH.dat'
92  dst = 'MESH.txt'
93
94  os.rename(src, dst)
95  file_path = 'MESH.txt'
96  nb_no, nb_elem = read_first_number_from_file(file_path)
```

Fonte: Autor (2024).

Depois, é definido o *array* dos nós ao chamar a função de leitura e armazenamento das linhas referentes aos nós. Como a primeira linha do arquivo .dat é a informação da quantidade de nós e elementos, então a linha número de nós + 1 corresponde ao último nó, Figura 75. Importante notar a substituição da informação da primeira linha que muda da quantidade de nós e elementos para *NODE, que é necessário no arquivo .inp.

Figura 75 – Criação do *array* dos nós

```
98  num_rows_no = nb_no + 1
99  no = read_rows_from_file(file_path, num_rows_no)
100 first_row = '*NODE'
101 no[0] = first_row
```

Fonte: Autor (2024).

De forma similar, define-se o *array* de elementos, Figura 76, mas a função difere pois precisa-se que a primeira linha a ser lida seja a linha “número de nós + 2”.

Figura 76 – Criação do *array* dos elementos

```
103 num_rows_elem = nb_elem
104 start_row = num_rows_no + 1
105 elem = read_rows_from_file_elem(file_path, start_row)
```

Fonte: Autor (2024).

Em seguida, filtra-se os elementos tridimensionais pela informação de cada linha indicando o elemento, por exemplo o 308 que indica um elemento tridimensional de 8 nós,

repete-se então o quinto nó dos elementos piramidais três vezes, insere-se a informação que precisa anteceder as linhas dos elementos e, por fim, o *array* dos elementos piramidais é inserido no fim do *array* dos hexaedros, Figura 77.

Figura 77 – Filtragem e criação dos *arrays* dos elementos tridimensionais

```
107  hex = filtered_array = [row for row in elem if row[1] == '308']
108  prism = filtered_array = [row for row in elem if row[1] == '306']
109  pyr = filtered_array = [row for row in elem if row[1] == '305']
110  for row in pyr:
111      row.extend([row[6], row[6], row[6]])
112  tet = filtered_array = [row for row in elem if row[1] == '304']
113
114  hex.insert(0, '*ELEMENT, TYPE=C3D8')
115  prism.insert(0, '*ELEMENT, TYPE=C3D6')
116  tet.insert(0, '*ELEMENT, TYPE=C3D4')
117
118  hex.extend(pyr)
```

Fonte: Autor (2024).

Assim, finalizando, numera-se os elementos deixando-os em ordem e a informação do tipo do elemento, proveniente do arquivo .dat, é apagada, escrevendo-se, então, o arquivo final no formato .inp, Figura 78.

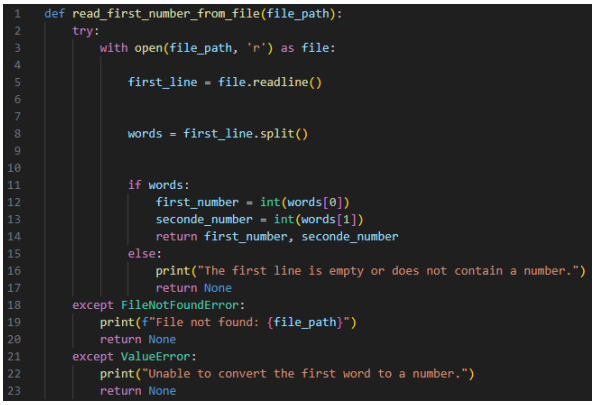
Figura 78 – Numeração dos elementos e escrita do arquivo final

```
120  for i in range(1, len(hex)):
121      hex[i][0] = i
122      del hex[i][1]
123
124  for i in range(1, len(prism)):
125      prism[i][0] = hex[-1][0] + i
126      del prism[i][1]
127
128  for i in range(1, len(tet)):
129      tet[i][0] = prism[-1][0] + i
130      del tet[i][1]
131
132  no_file = 'mesh.inp'
133  write_to_file(no_file, no, hex, prism, tet)
```

Fonte: Autor (2024).

Para as funções, primeiramente, tem-se a função de leitura e armazenamento da primeira linha do arquivo .dat que retorna o número de nós e elementos, Figura 79. Depois, tem- se a função para ler e armazenar “x” linhas do arquivo .dat e servirá para ler a primeira linha e as linhas dos nós, ler e armazenar número de nós + 1 linhas, Figura 80. De forma similar, a função da Figura 81 lê e armazena “x” linhas, mas começa de uma linha diferente da primeira, sendo usada para ler e armazenar as linhas dos elementos. Por fim, tem-se a função de escrever o arquivo, obedecendo cada tipo de espaçamento descrito na seção de metodologia, Figura 82.

Figura 79 – Função para leitura e armazenamento da primeira linha do arquivo .dat



```python
def read_first_number_from_file(file_path):
    try:
        with open(file_path, 'r') as file:

            first_line = file.readline()


            words = first_line.split()


            if words:
                first_number = int(words[0])
                seconde_number = int(words[1])
                return first_number, seconde_number
            else:
                print("The first line is empty or does not contain a number.")
                return None
    except FileNotFoundError:
        print(f"File not found: {file_path}")
        return None
    except ValueError:
        print("Unable to convert the first word to a number.")
        return None
```

Fonte: Autor (2024).

Figura 80 – Função para leitura e armazenamento de “x” linhas do arquivo .dat

```python
def read_rows_from_file(file_path, num_rows):
    data = []

    try:
        with open(file_path, 'r') as file:
            for _ in range(num_rows):
                line = file.readline()
                if not line:
                    break

                values = line.strip().split()
                data.append(values)

    except FileNotFoundError:
        print(f"File not found: {file_path}")
    except Exception as e:
        print(f"Error reading file: {e}")

    return data
```

Fonte: Autor (2024).

Figura 81 – Função para leitura e armazenamento de “x” linhas do arquivo .dat começando de uma linha específica

```python
def read_rows_from_file_elem(file_path, start_row):
    data = []

    try:
        with open(file_path, 'r') as file:
            for _ in range(start_row - 1):
                file.readline()

            while True:
                line = file.readline()
                if not line:
                    break

                values = line.strip().split()
                data.append(values)

    except FileNotFoundError:
        print(f"File not found: {file_path}")
    except Exception as e:
        print(f"Error reading file: {e}")

    return data
```

Fonte: Autor (2024).

Figura 82 – Função para escrita do arquivo .inp

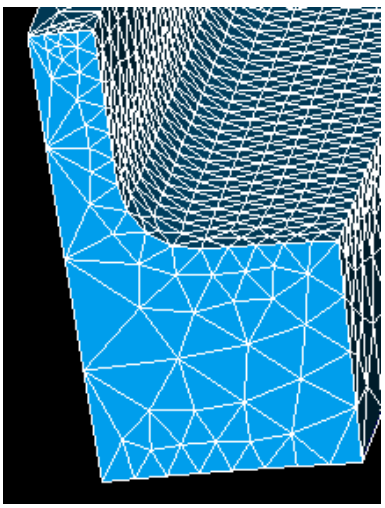
```
def write_to_file(file_path, data_no, data_hex, data_prism, data_tet):
    try:
        with open(file_path, 'w') as file:
            file.write(data_no[0] + '\n')
            for row in data_no[1:]:
                file.write(', '.join(map(str, row)) + '\n')
            file.write(data_hex[0] + '\n')
            for row in data_hex[1:]:
                file.write(','.join(map(str, row)) + '\n')
            file.write(data_prism[0] + '\n')
            for row in data_prism[1:]:
                file.write(','.join(map(str, row)) + '\n')
            file.write(data_tet[0] + '\n')
            for row in data_tet[1:]:
                file.write(','.join(map(str, row)) + '\n')
        print("File written successfully.")
    except Exception as e:
        print(f"Error writing to file: {e}")
```

Fonte: Autor (2024).

4.3 TESTES DE MODELOS VARIADOS

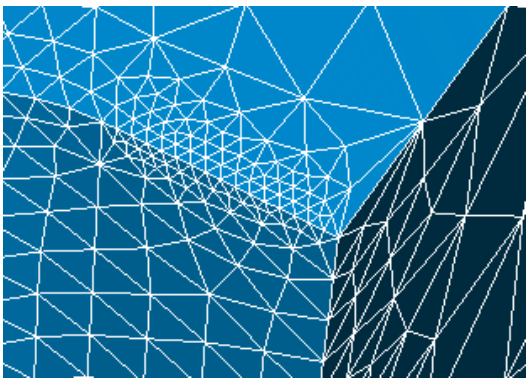
Tendo definido os parâmetros das amostras, utilizou-se do *script* desenvolvido para gerar cada duto teste, possibilitando perceber as limitações e aplicações do código. Como limitações da ferramenta desenvolvida, é possível perceber que, em geral, as regiões com malha muito distorcida acontecem em geometrias com locais em que uma dimensão é muito maior que as outras adjacentes, como quando a espessura é pequena e o comprimento linear ou angular é muito maior. As Figuras 83 a 87 ilustram alguns casos dessas distorções.

Figura 83 – Face da RD da amostra 3



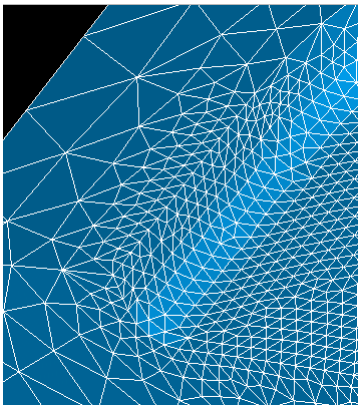
Fonte: Autor (2024).

Figura 84 – Malha excessivamente refinada na RD da amostra 4



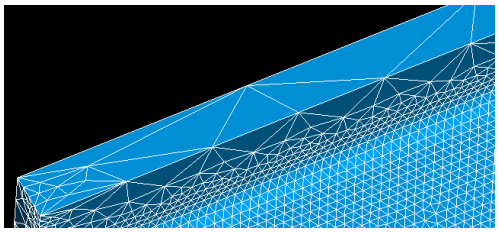
Fonte: Autor (2024).

Figura 85 – Transição brusca na amostra 8

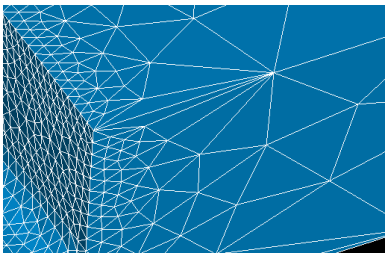


Fonte: Autor (2024).

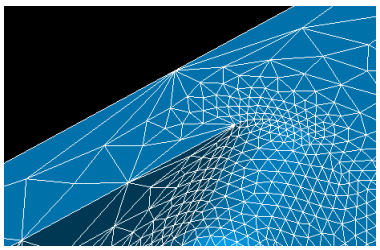
Figura 86 – Regiões de atenção nas malhas da amostra 19



(a) Elementos ruins na malha na RD.



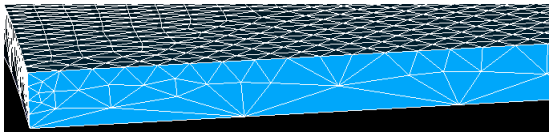
(b) Elementos muito distorcidos.



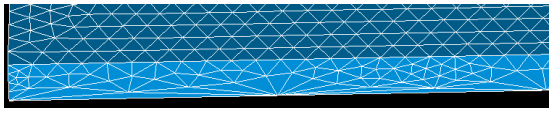
(c) Região de comportamento ruim.

Fonte: Autor (2024).

Figura 87 – Transição brusca na região da espessura remanescente



(a) Amostra 18.

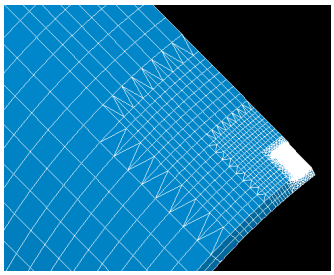


(b) Amostra 21.

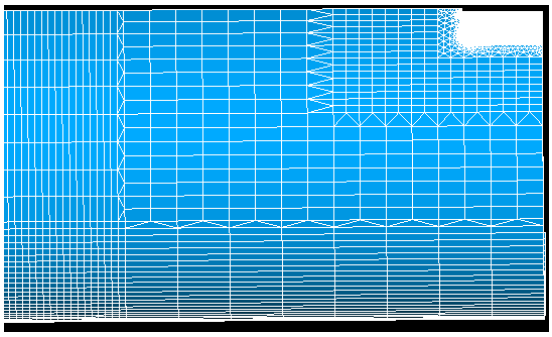
Outra limitação que foi percebida durante o desenvolvimento é que RA não pode ser maior que RC, pois acaba acarretando em nomenclaturas diferentes, no *software*, para as arestas dos filetes, fazendo com que se perca a referência da malha nesses locais. Por isso, não foi possível manter a proporção desejada do RA ser metade da profundidade do defeito

em todas as amostras, como mencionado na seção anterior, acarretando nas RD's com malha demasiadamente refinada, nas geometrias com ângulo defeito pequeno, visto que o RA é um dos parâmetros para a malha e RC é pequeno nesses casos devido ao comprimento angular ser pequeno, restringindo RA. Uma outra perda de referência foi notada quando o ângulo do defeito é grande o suficiente, por exemplo 40° de comprimento angular total, para fazer com que a região excluída fique maior que a região RT3 A, fazendo com que o *software* dê outra nomenclatura para a região e as referências de malha são perdidas. Além disso, podem ocorrer problemas quando o comprimento linear é excessivamente pequeno em relação às outras dimensões, acarretando no mal funcionamento do *software* e não gerando a geometria. Assim, o observado é que o uso ideal do *script* é para defeitos com comprimento angular de 5° a 15°, com comprimentos até 150 mm e profundidade menor que 75% da espessura do tubo. Importante notar que muitas dessas limitações são casos hipotéticos, ou seja, o comportamento do código é melhor em casos reais de defeitos, principalmente quando são mais "quadrados", com o comprimento linear e angular mais equivalentes. Além disso, a Figura 88 mostra como a malha das RT's se comportou nos casos dos defeitos de menor ângulo e menor comprimento longitudinal e maior ângulo e comprimento longitudinal, não apresentando problemas. Os resultados de cada amostra dos testes podem ser vistos no apêndice A.

Figura 88 – Malha das regiões de transição



(a) Malha das RT's da amostra 11.



(b) Malha das RT's da amostra 21.

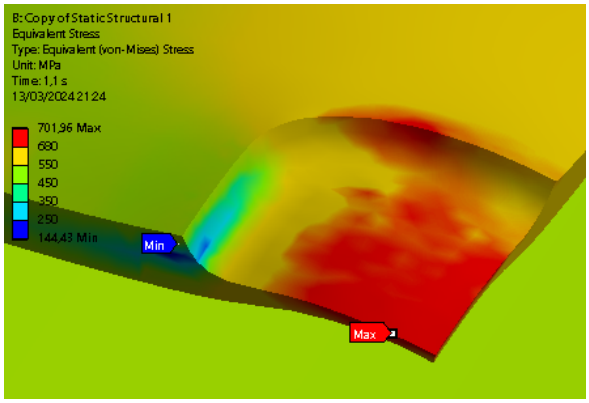
Fonte: Autor (2024).

4.4 MODELOS ANALISADOS

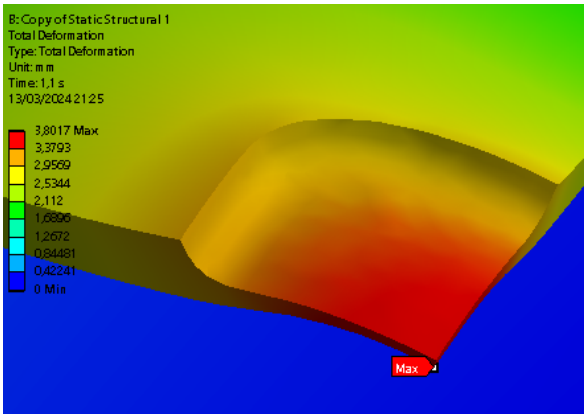
Tendo definido todos os parâmetros das análises descritos em metodologia, realizou-se a análise para o MD1 e o MD2. No MD1, a solução convergiu até o quinto sub-passo entre o passo 1 e 2, correspondente à pressão interna de 22,05 MPa, obtendo-se uma tensão máxima de 701,96 MPa, abaixo da tensão de ruptura do material de 718,2 MPa, e uma deformação máxima de 3,8 mm, Figura 89. De forma similar, em Cabral (2007), o modelo também não atinge a pressão de ruptura e a pressão interna é de 22,09 MPa, próxima à obtida no MD1. A grande diferença está na distribuição de tensões, em que as tensões

se desenvolvem próximas da parede no modelo de Cabral (2007), enquanto as tensões se desenvolvem a partir do meio do defeito no modelo do presente trabalho. O fato da pressão inicial ser aplicada num passo único pode ser uma razão da diferença. O resultado final obtido por Cabral (2007) está na Figura 90, sendo o modelo IDTS 2 equivalente ao MD1, como mencionado na metodologia.

Figura 89 – Tensão e deformação do MD1



(a) Distribuição de tensões para o MD1.



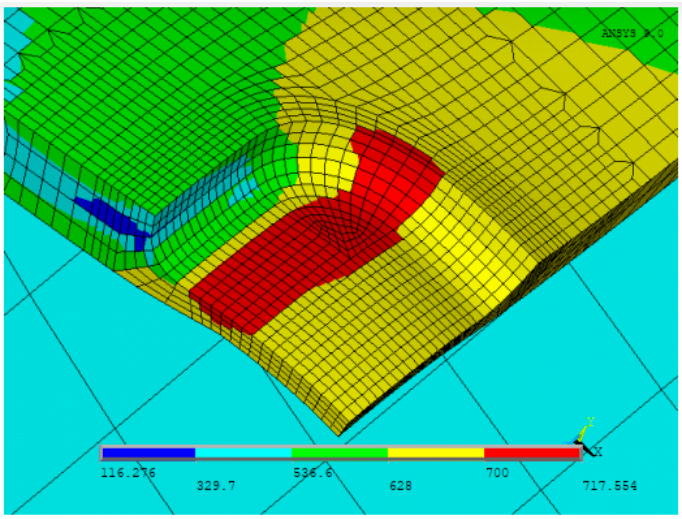
(b) Distribuição de deformação para o MD1

Fonte: Autor (2024).

Figura 90 – Resultados obtidos no trabalho base 1

	Pressões de Falha [MPa]				Erro[1] (%)		
	Experimental	Numérico (MEF)		Empírico	Numérico		Empírico
Espécime	$Pf_{(EXP)}$	Pf_1 (Andrade et al, 2006)	Pf_2 (este trabalho)	Pf_3 (BS7910)	(Andrade et al, 2006)	(este trabalho)	(BS7910)
IDTS2	22,679	22,710	22,0894	21,253	+0,14	-2,60	-6,29

(a) Comparação dos valores obtidos por Cabral (2007), Benjamin *et al.* (2005) e Andrade *et al.* (2006) para o IDTS 2.



(b) Distribuição das tensões obtida por (CABRAL, 2007).

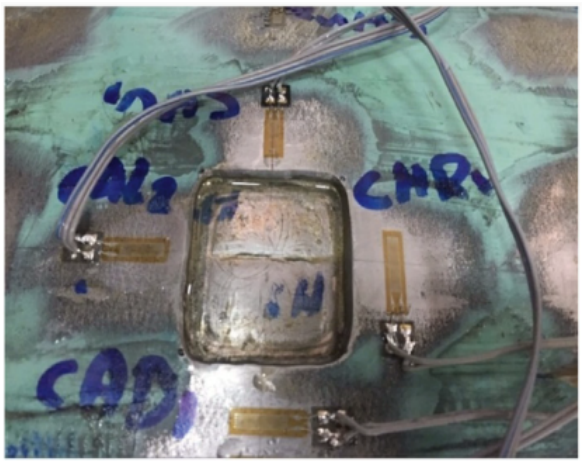
Fonte: Cabral (2007).

Apesar do modelo de Cabral (2007) concordar com os experimentos de Benjamin *et al.* (2005), Figura 91, faz sentido as tensões se desenvolverem no meio, principalmente por conta da simetria do trabalho, sem preferência para nenhum dos lados, e esse comportamento é corroborado por S. Al-Owaisi *et al.* (2018), que realizou experimentos e os defeitos quadriláteros falharam no meio, Figura 92, ou na base do raio de adoçamento, sendo a causa do último, de acordo com o autor, possíveis imperfeições causadas durante a usinagem ou concentração de tensões.

Figura 91 – Falha do IDTS 2

Fonte: Benjamin *et al.* (2005).

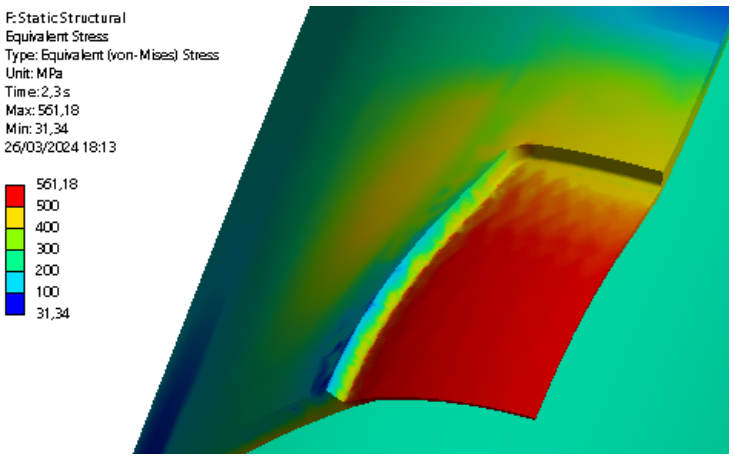
Figura 92 – Falha com defeito único



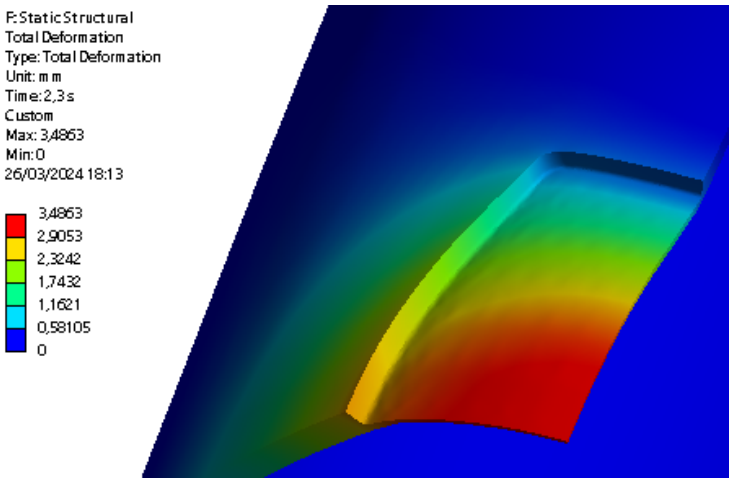
Fonte: S. Al-Owaisi *et al.* (2018).

Já para o MD2, obteve-se uma tensão máxima de 561,18 MPa e uma deformação máxima de aproximadamente 3,49 mm, Figura 93, para uma pressão de 13,65 MPa, pressão de ruptura obtida por Tee e Wordu (2020). A geometria do defeito o coloca nas limitações descritas do código e sua malha possui elementos muitos distorcidos, como já ilustrado na Figura 50. Consequentemente, espera-se que os valores sejam discordantes, mas, comparando os valores de tensão máxima na pressão de 13,65 MPa, os valores estão muito próximos. Apesar dos valores, a distribuição das tensões em si é muito diferente, mesmo que o comportamento da deformação pareça similar. Além da malha, como já comentado no MD1, o fato da pressão inicial ser aplicada num passo único pode ser uma razão da diferença, assim como a falta de informação dos raios de adoçamento, por poderem gerar concentrações de tensão, e a utilização de um *software* diferente podem ter contribuído nas divergências. O resultado obtido por Tee e Wordu (2020), para a pressão de 13,65 MPa, está na Figura 94.

Figura 93 – Tensão e deformação do MD2



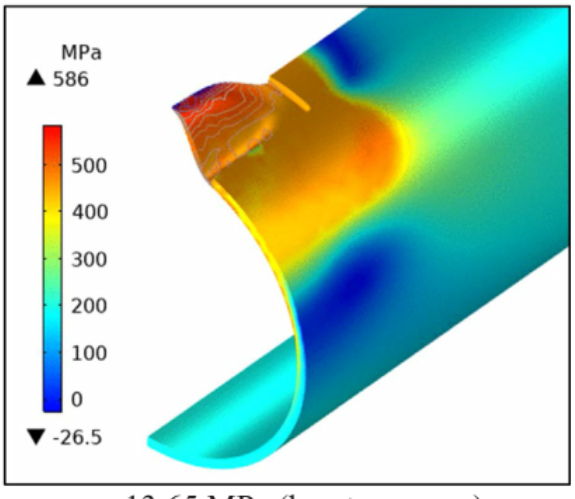
(a) Distribuição de tensões para o MD1.



(b) Distribuição da deformação para o MD2.

Fonte: Autor (2024).

Figura 94 – Distribuição de tensões obtidas no trabalho base 2



13.65 MPa (burst pressure)

Fonte: Tee e Wordu (2020).

5 CONCLUSÃO

Assim, diante do contexto da grande importância das linhas de duto, principalmente no transporte e distribuição de combustíveis como petróleo e gás natural, a utilização do MEF ganha destaque quando os métodos semi-empíricos de avaliação de defeitos não são suficientes.

No entanto, é importante reconhecer que a aplicação bem-sucedida do MEF muitas vezes requer um nível de conhecimento técnico específico que nem todos os engenheiros de tubulações possuem. Além disso, outro desafio reside nos custos associados aos softwares especializados necessários para conduzir análises por meio do MEF. Esses custos podem representar uma barreira significativa para muitas empresas, profissionais e na área de ensino, limitando o acesso a essa importante ferramenta de engenharia. Assim, em vista das problemáticas apresentadas, o presente trabalho visou a diminuição dessas barreiras ao ter desenvolvido ferramentas para a geração automatizada de modelos computacionais de dutos com defeito de corrosão idealizado.

Para isso, modelou-se uma geometria arbitrária de duto com defeito de corrosão idealizado de forma parametrizada, foi, então, gerada uma malha visualmente adequada e gerou-se um código, por meio do *software* SALOME, que automatiza essas etapas e permite sua utilização para outras geometrias. Com o código devidamente ajustado e funcional, realizou-se testes com diferentes geometrias para entender os limites dele, resultando na obtenção do escopo para o qual o código gera uma boa malha.

Para comprovar a qualidade dos modelos, foram realizadas análises estruturais não lineares de modelos baseados em outros trabalhos da área que foram gerados com a ferramenta desenvolvida. Para fazer a comunicação do SALOME com o programa das análises, ANSYS, foi desenvolvido um código em Python que converte o arquivo da malha exportado pelo SALOME para um formato que o ANSYS consegue importar. Assim, fez-se as análises e os resultados obtidos foram comparados com os trabalhos base, obtendo-se boa correspondência e corroborando para os limites de aplicação do código

Com isso, sendo necessárias apenas modificações nos parâmetros, diminuiu-se a necessidade de intervenção manual do engenheiro nas etapas de geração da geometria e da malha, juntamente com uma economia no tempo gasto durante essas etapas. Além disso, a utilização do SALOME, software gratuito, e da linguagem Python permite o acesso facilitado das ferramentas por estudantes e profissionais, possibilitando a evolução delas e dá um primeiro passo na diminuição do impacto financeiro que o método geralmente impõe.

Por fim, com a aplicação da ferramenta testada e documentada, espera-se que ela seja utilizada e difundida e que os resultados deste trabalho auxiliem futuras pesquisas e estudos da área, voltados, por exemplo, a tornar a ferramenta mais versátil e robusta.

5.1 TRABALHOS FUTUROS

Como sugestões para futuros desenvolvimentos e pesquisas, tem-se:

- Verificar se outras formas de modelagem diminuem os problemas de perda de referência por parte do *software*;
- Verificar outros parâmetros para as malhas de forma a deixar o código mais versátil, por exemplo utilizando os comprimentos linear ou circunferencial;
- Evoluir a ferramenta para gerar defeitos em posições arbitrárias do duto;
- Evoluir o código para gerar a geometria e malha de múltiplos defeitos;
- Contemplar a automatização da análise utilizando recursos gratuitos.

REFERÊNCIAS

ALVES FILHO, A. **ELEMENTOS FINITOS A BASE DA TECNOLOGIA CAE**. São Paulo: Érica, 2000.

ANDRADE, E. Q.; BENJAMIN, A. C.; MACHADO JR., P. R. S.; PEREIRA, L. C.; JACOB, B. P.; CARNEIRO, E. G.; GUERREIRO, J. N. C.; SILVA, R. C. C.; NORONHA JR., D. B. FINITE ELEMENT MODELING OF THE FAILURE BEHAVIOR OF PIPELINES CONTAINING INTERACTING CORROSION DEFECTS. **25th International Conference on Offshore Mechanics and Arctic Engineering**, Hamburg, Germany, 2006.

ANSYS. **Ansys Mechanical - Finite Element Analysis (FEA) Software for Structural Engineering**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.ansys.com/products/structures/ansys-mechanical. Acesso em: 05 mar. 2024.

ANSYS. **Mechanical User's Guide**. Canonsburg, PA: ANSYS, Inc, 2023.

API. API RP 571 - Damage Mechanisms Affecting Fixed Equipment in the Refining Industry. **AMERICAN PETROLEUM INSTITUTE**, 2011.

ASME. **ASME B31G - Manual for Determining the Remaining Strength of Corroded Pipelines**. New York, USA: The American Society of Mechanical Engineers, 1991.

ASME. **ASME B36.10M - WELDED AND SEAMLESS WROGHT STEEL PIPE**. New York, NY: The American Society of Mechanical Engineers, 2004.

BEDAIRI, B.; CRONIN, D.; HOSSEINI, A.; PLUMTREE, A. Failure prediction for Crack-in-Corrosion defects in natural gas transmission pipelines. **International Journal of Pressure Vessels and Piping**, v. 96-97, p. 90–99, 2012.

BEER, F. P.; JOHNSTON, JR., E. R.; DEWOLF, J. T.; MAZUREK, D. F. **MECHANICS OF MATERIALS - 6th edition**. New York, NY: The McGraw-Hill Companies, Inc, 2012.

BENJAMIN, A. C.; FREIRE, J. L. F.; VIEIRA, R. D.; DINIZ, J. L. C.; DE ANDRADE, E. Q. BURST TESTS ON PIPELINE CONTAINING INTERACTING

CORROSION DEFECTS. **24th International Conference on Offshore Mechanics and Arctic Engineering**, Halkidiki, Greece, 2005.

BRUÈRE, V. M.; BOUCHONNEAU, N.; MOTTA, R. S. Failure pressure prediction of corroded pipes under combined internal pressure and axial compressive force. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, v. 42, n. 172, 2019.

BUDYNAS, R. G.; NISBETT, J. K. **Shigley's Mechanical Engineering Design - Ninth edition**. New York, NY: The McGraw-Hill Companies, Inc, 2011.

CABRAL, H. L. D. **Desenvolvimento de ferramentas computacionais para modelagem e análise automática de defeitos de corrosão em dutos**. Recife: Universidade Federal de Pernambuco, 2007.

CABRAL, H. L. D.; MOTTA, R. S.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; LYRA, P. R. M.; ANDRADE, E. Q. de. The development of a computational tool for generation of high quality FE models of pipelines with corrosion defects. **J Braz. Soc. Mech. Sci. Eng.**, v. 39, p. 3137–3150, 2017.

CABRAL, H. L. D.; WILLMERSDORF1, R. B.; AFONSO, S. M. B.; LYRA, P. R. M.; ANDRADE, E. Q. de. Development of Computational Tools for Automatic Modeling and FE Analysis of Corroded Pipelines. **INTERNATIONAL JOURNAL OF MODELING AND SIMULATION FOR THE PETROLEUM INDUSTRY**, v. 1, n. 1, 2007.

CHANDRUPATLA, T. R.; BELENGUNDU, A. D. **Elementos Finitos - 4^a Edição**. São Paulo, Brasil: Pearson Education do Brasil, Ltda, 2015.

CHEN, Y.; ZHANG, H.; ZHANG, J.; LIU, X.; LI, X.; ZHOU, J. Failure assessment of X80 pipeline with interacting corrosion defects. **Engineering Failure Analysis**, v. 47, p. 67–76, 2015. ISSN 1350-6307.

COOK, R. D.; MALKUS, D. S.; PLESHA, M. E. **CONCEPTS AND APPLICATIONS OF FINITE ELEMENT ANALYSIS - THIRD EDITION**. New York, USA: John Wiley & Sons, Inc, 1989.

COSHAM, A.; KIRKWOOD, Mike. BEST PRACTICE IN PIPELINE DEFECT ASSESSMENT. **IPC00-0205 - International Pipeline Conference**, Alberta, Canada, 2000.

DASSAULT SYSTEMES. **Abaqus - Finite Element Analysis for Mechanical Engineering and Civil Engineering**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.3ds.com/products/simulia/abaqus. Acesso em: 09 mar. 2024.

DE SOUZA NETO, E. A.; PERIC, D.; OWEN, D. R. J. **COMPUTATIONAL METHODS FOR PLASTICITY - THEORIES AND APPLICATIONS**. West Sussex, UK: John Wiley & Sons Ltd, 2008.

DNV, Corroded Pipelines. Recommended Practice DNV-RP-F101. DNV Oslo, Norway, 2017.

EPE, Empresa de Pesquisa Energética. **Matriz Energética e Elétrica**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.epe.gov.br/pt/abcdenergia/matriz-energetica-e-eletrica. Acesso em: 04 mar. 2024.

FERREIRA, A. D. M.; MOTTA, R. S.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; LYRA, P. R. M.; ANDRADE, E. Q. de; CUNHA, D. J. S. Stochastic assessment of burst pressure for corroded pipelines. **J Braz. Soc. Mech. Sci. Eng.**, v. 43, n. 193, 2021.

FERREIRA, A. D. M.; MOTTAA, R. S.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; LYRAA, P. R. M.; BENJAMIN, A. C.; ANDRADE, E. Q. de. ESTUDO DE CONVERGÊNCIA DE MALHAS PARA A ANÁLISE DE DUTOS COM DEFEITOS OBTIDOS ATRAVÉS DA INSPEÇÃO DE DADOS DE CAMPO. **Mecánica Computacional**, v. XXIX, p. 7725–7739, 2010.

FERREIRA, A. D. M.; PIMENTEL, J. T.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; SILVA, L. M. T. da. ESTUDO DA INFLUÊNCIA DO RAIO DE ADOÇAMENTO NA OBTENÇÃO DA PRESSÃO DE FALHA VIA MEF PARA DUTOS COM DEFEITOS DE CORROSÃO. **XXXVII Iberian Latin-American Congress on Computational Methods in Engineering**, Brasília, Brasil, 2016.

FERREIRA, A. D. M.; SOUZA, A. H. T. de; WILLMERSDORF, R. B.; AFONSO, S. M. B.; LYRA, P. R. M.; MOTTA, R. S.; ANDRADE, E. Q. de. A GENERAL PROCEDURE TO MODEL AND TO ANALYZE PIPELINES WITH

REAL DEFECTS CAUSED BY CORROSION MEASURED IN SITU. **Brazilian Petroleum, Gas and Biofuels Institute - IBP**, 2009.

FREIRE, J.L.F.; VIEIRA, R.D.; CASTRO, J.T.P.; BENJAMIN, A.C. PART 3: BURST TESTS OF PIPELINE WITH EXTENSIVE LONGITUDINAL METAL LOSS. **Experimental Techniques**, v. 30, n. 6, p. 60–65, 2006.

HAN, C.J.; ZHANG, H.; ZHANG, J. Failure Pressure Analysis of the Pipe with Inner Corrosion Defects by FEM. **International Journal of Electrochemical Science**, v. 11, n. 6, p. 5046–5062, 2016. ISSN 1452-3981.

HEXAGON. **Patran - Complete FEA modeling solution**. [*S.l.*: *s.n.*], 2024. Disponível em: https://hexagon.com/products/patran. Acesso em: 04 mar. 2024.

KIEFNER, J F; VIETH, P H. A modified criterion for evaluating the remaining strength of corroded pipe. United States, dez. 1989.

LIU, H.; KHAN, F.; THODI, P. Revised burst model for pipeline integrity assessment. **Engineering Failure Analysis**, v. 80, p. 24–38, 2017. ISSN 1350-6307.

MESHIO. **meshio 5.3.5**. [*S.l.*: *s.n.*], 2024. Disponível em: https://pypi.org/project/meshio/#description. Acesso em: 09 mar. 2024.

MMS. **CONTRACT NO. 1435-01-CT-99-50001 APPRAISAL AND DEVELOPMENT OF PIPELINE DEFECT ASSESSMENT METHODOLOGIES**. Ascot, Berkshire: MSL Engineering Limited, 2000.

MOTTA, R. S.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; LYRA, P. R. M.; ANDRADE, E. Q. de. AUTOMATIC MODELING AND ANALYSIS OF PIPELINES WITH COLONIES OF CORROSION DEFECTS. **Mecánica Computacional**, v. XXIX, p. 7871–7890, 2010.

MOTTA, R. S.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; LYRA, P. R. M.; CABRAL, H. L. D.; ANDRADE, E. Q. AUTOMATIC GEOMETRIC MODELING, MESH GENERATION AND FE ANALYSIS FOR PIPELINES WITH IDEALIZED DEFECTS AND ARBITRARY LOCATION. **Brazilian Petroleum, Gas and Biofuels Institute - IBP**, 2009.

MOTTA, R. S.; CABRAL, H. L.D.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; BOUCHONNEAU, N.; LYRA, P. R. M.; DE ANDRADE, E. Q. Comparative studies for failure pressure prediction of corroded pipelines. **Engineering Failure Analysis**, v. 81, p. 178–192, 2017. ISSN 1350-6307.

OPENAI. **Introducing ChatGPT**. [*S.l.*: *s.n.*], 2024. Disponível em: https://openai.com/blog/chatgpt. Acesso em: 07 mar. 2024.

AL-OWAISI, S.; BECKER, A. A.; SUN, W.; AL-SHABIBI, A.; AL-MAHARBI, M.; P., T.; AL-SALMI, H. An experimental investigation of the effect of defect shape and orientation on the burst pressure of pressurised pipes. **Engineering Failure Analysis**, v. 93, p. 200–213, 2018.

AL-OWAISI, S. S.; BECKER, A. A.; SUN, W. Analysis of shape and location effects of closely spaced metal loss defects in pressurised pipes. **Engineering Failure Analysis**, v. 68, p. 172–186, 2016.

PIMENTEL, J. T.; FERREIRA, A. D. M.; MOTTA, R. S.; CABRAL, M. A. F. S.; AFONSO, S. M. B.; WILLMERSDORF, R. B.; LYRA, P. R. M.; DE ANDRADE, E. Q.; CUNHA, D. J. S. New procedure of automatic modeling of pipelines with realistic shaped corrosion defects. **Engineering Structures**, v. 221, p. 111030, 2020. ISSN 0141-0296.

PYTHON. **Python**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.python.org. Acesso em: 05 mar. 2024.

RAMBERG, W.; OSGOOD, W. R. TECHNICAL NOTE NO. 902 - DESCRIPTION OF STRESS-STRAIN CURVES BY THREE PARAMETERS. NASA, 1943.

SALOME. **Functionalities**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.salome-platform.org/?page_id=23. Acesso em: 05 mar. 2024.

SIMSCALE. **Mesh Quality**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.simscale.com/docs/simulation-setup/meshing/mesh-quality/#:~:text=It%20is%20recommended%20to%20keep%20the%20maximum%20skewness%20below%200.5,maximum%20skewness%20is%20above%200.85.. Acesso em: 14 mar. 2024.

SOARES, E.; BRUÈRE, V. M.; AFONSO, S. M.B.; WILLMERSDORF, R. B.; LYRA, P. R.M.; BOUCHONNEAU, N. Structural integrity analysis of pipelines with

interacting corrosion defects by multiphysics modeling. **Engineering Failure Analysis**, v. 97, p. 91–102, 2019. ISSN 1350-6307.

SUN, J.; CHENG, Y. F. Assessment by finite element modeling of the interaction of multiple corrosion defects and the effect on failure pressure of corroded pipelines. **Engineering Structures**, v. 165, p. 278–286, 2018. ISSN 0141-0296.

TEE, K. F.; WORDU, A. H. Burst strength analysis of pressurized steel pipelines with corrosion and gouge defects. **Engineering Failure Analysis**, v. 108, p. 104347, 2020. ISSN 1350-6307.

TRANSPETRO. **Dutos e Terminais**. [*S.l.*: *s.n.*], 2024. Disponível em: https://transpetro.com.br/transpetro-institucional/nossas-atividades/dutos-e-terminais.htm. Acesso em: 04 mar. 2024.

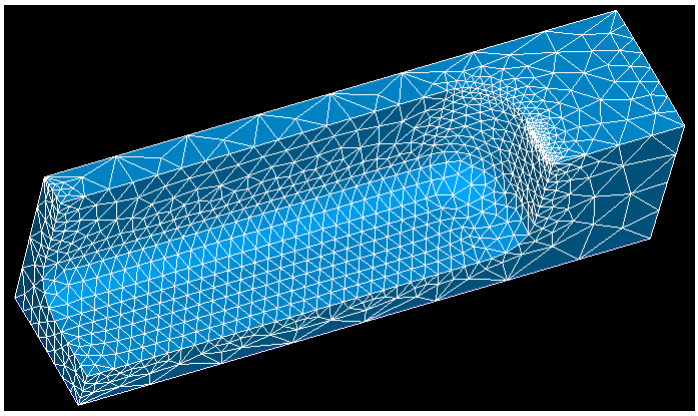
U.S. EIA. **Natural gas explained - Natural gas pipelines**. [*S.l.*: *s.n.*], 2024. Disponível em: https://www.eia.gov/energyexplained/natural-gas/natural-gas-pipelines.php#:~:text=The%20pipeline%20network%20has%20about,to%20about%2077.7%20million%20consumers.. Acesso em: 04 mar. 2024.

XU, L.Y.; CHENG, Y.F. Reliability and failure pressure prediction of various grades of pipeline steel in the presence of corrosion defects and pre-strain. **International Journal of Pressure Vessels and Piping**, v. 89, p. 75–84, 2012. ISSN 0308-0161.

ZEID, I. F. Investigating and Comparing Two Different CAD Methodologies to Create Top-down Assemblies. **ASEE Annual Conference & Exposition**, 2016.

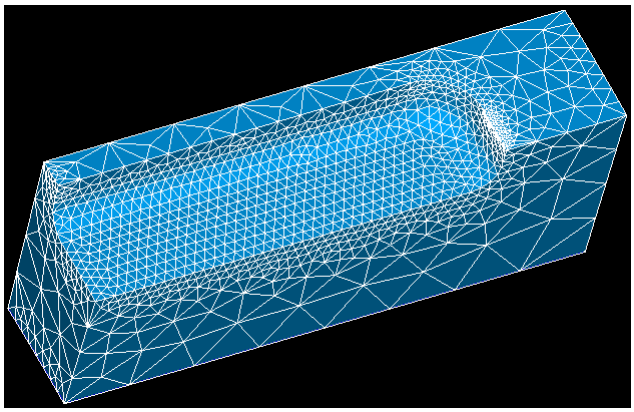
APÊNDICE A – Resultados dos testes do código de geração dos modelos

Figura 95 – Malha da RD da amostra 1



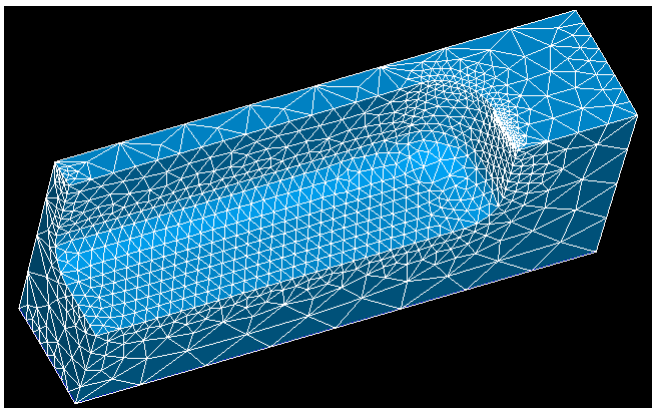
Fonte: Autor (2024).

Figura 96 – Malha da RD da amostra 2



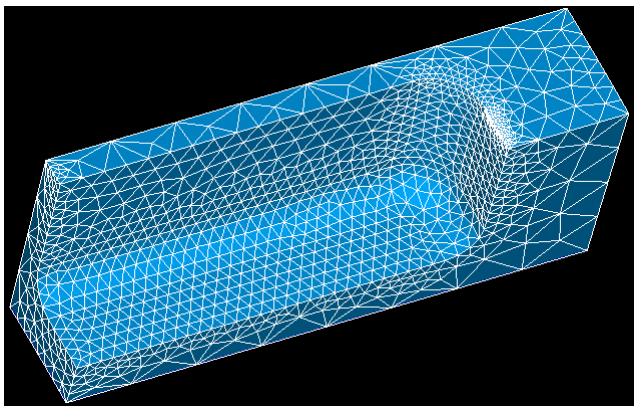
Fonte: Autor (2024).

Figura 97 – Malha da RD da amostra 3



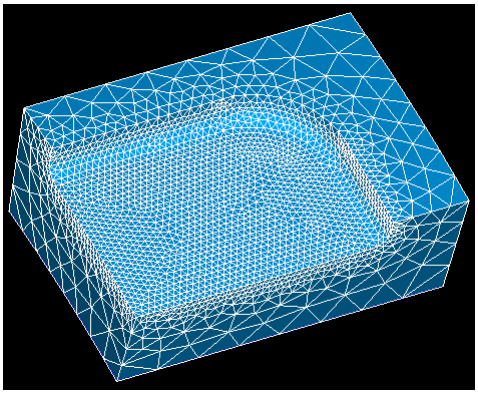
Fonte: Autor (2024).

Figura 98 – Malha da RD da amostra 4



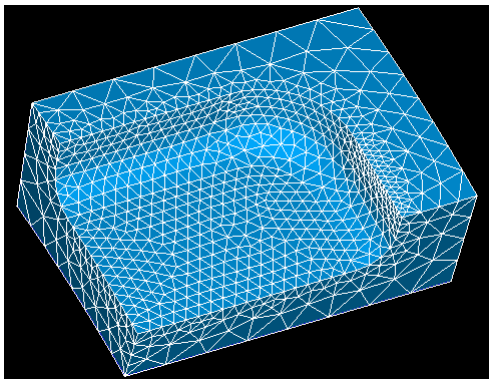
Fonte: Autor (2024).

Figura 99 – Malha da RD da amostra 5



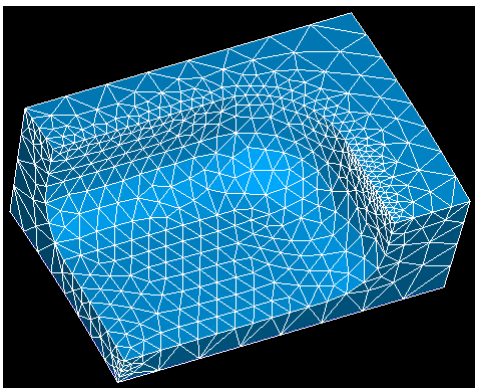
Fonte: Autor (2024).

Figura 100 – Malha da RD da amostra 6



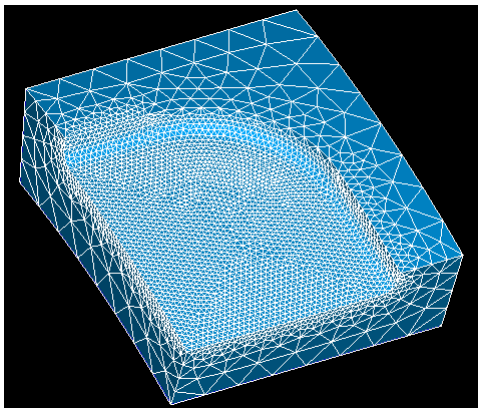
Fonte: Autor (2024).

Figura 101 – Malha da RD da amostra 7



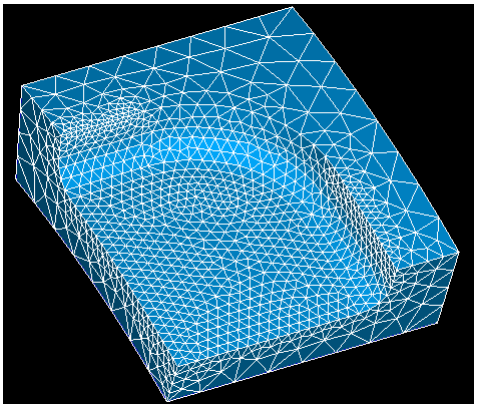
Fonte: Autor (2024).

Figura 102 – Malha da RD da amostra 8



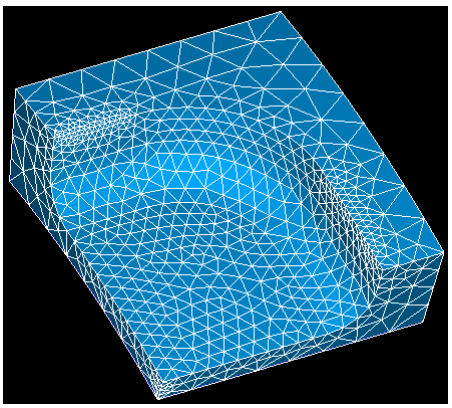
Fonte: Autor (2024).

Figura 103 – Malha da RD da amostra 9



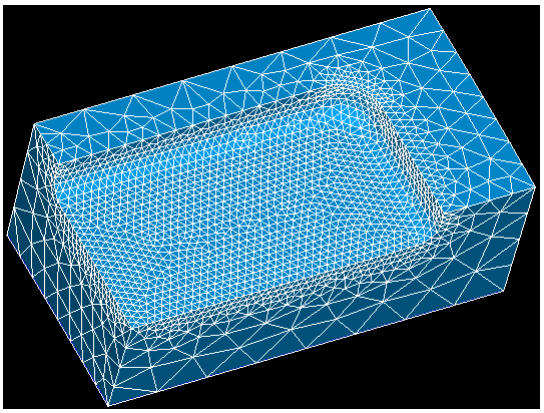
Fonte: Autor (2024).

Figura 104 – Malha da RD da amostra 10



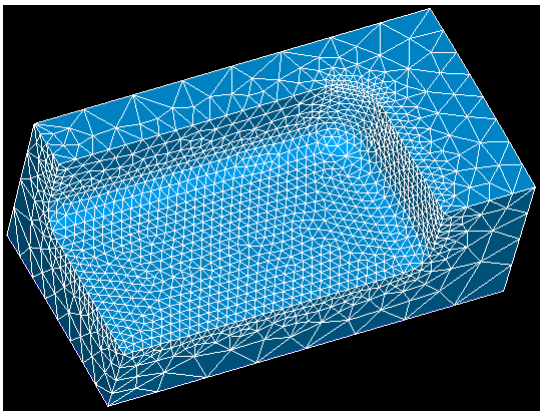
Fonte: Autor (2024).

Figura 105 – Malha da RD da amostra 11



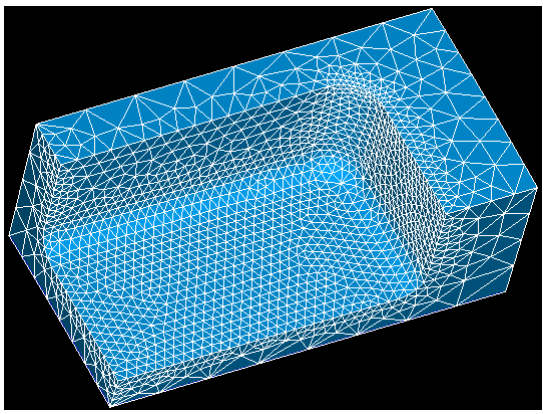
Fonte: Autor (2024).

Figura 106 – Malha da RD da amostra 12



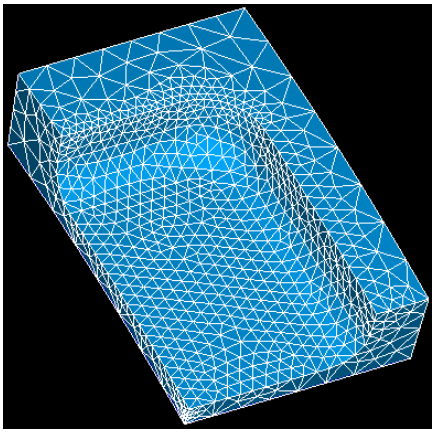
Fonte: Autor (2024).

Figura 107 – Malha da RD da amostra 13



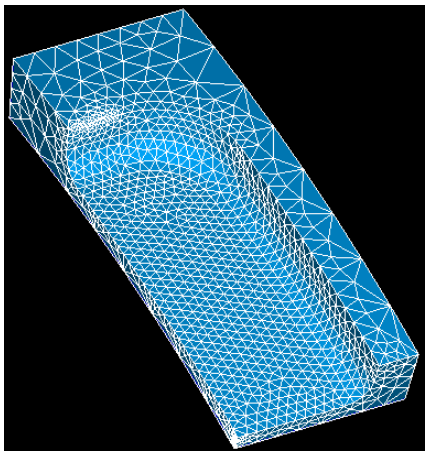
Fonte: Autor (2024).

Figura 108 – Malha da RD da amostra 14



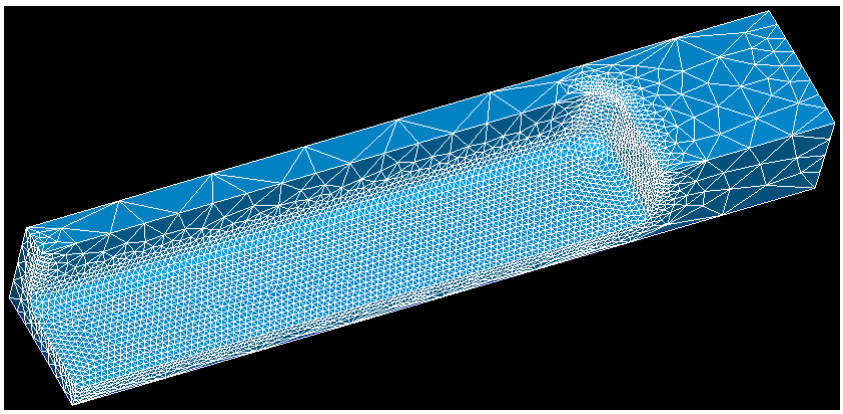
Fonte: Autor (2024).

Figura 109 – Malha da RD da amostra 15



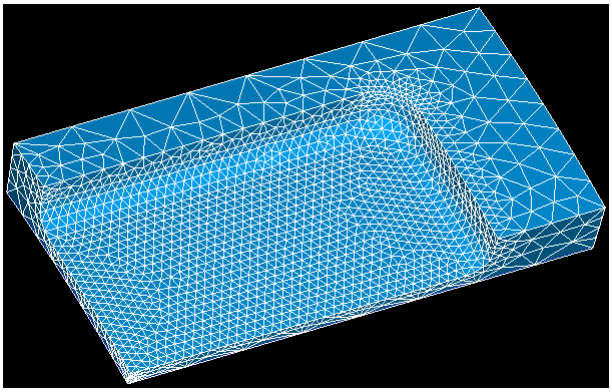
Fonte: Autor (2024).

Figura 110 – Malha da RD da amostra 16



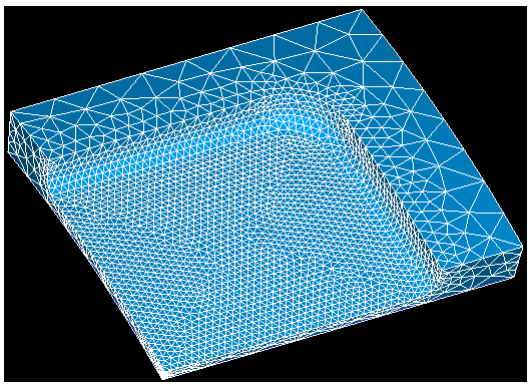
Fonte: Autor (2024).

Figura 111 – Malha da RD da amostra 17



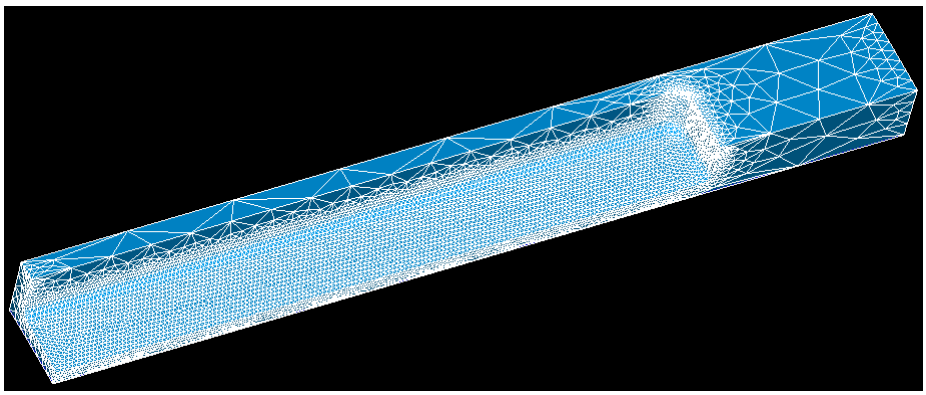
Fonte: Autor (2024).

Figura 112 – Malha da RD da amostra 18



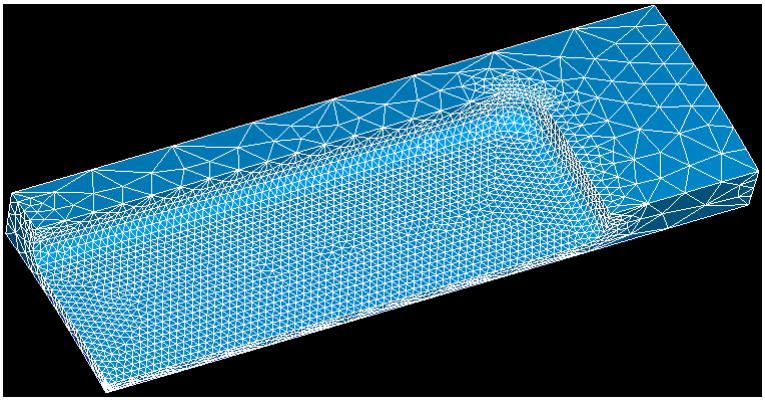
Fonte: Autor (2024).

Figura 113 – Malha da RD da amostra 19



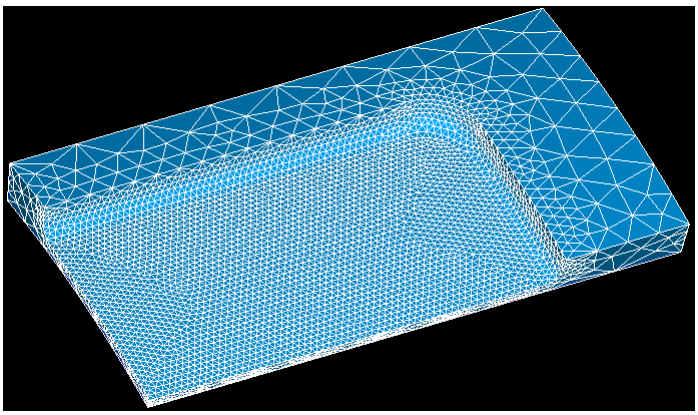
Fonte: Autor (2024).

Figura 114 – Malha da RD da amostra 20



Fonte: Autor (2024).

Figura 115 – Malha da RD da amostra 21



Fonte: Autor (2024).