1 Introdução

O presente trabalho busca desenvolver um código numérico utilizando linguagem C# nos moldes de uma API capaz de realizar cálculos de viscoelasticidade aplicada a tecidos moles, baseando-se nos modelos linear de Maxwell, quase-linear de Fung e não-linear de Schapery, além de gerar um arquivo externo em formato CSV com os resultados para comparar com os dados experimentais. Busca-se com este projeto auxiliar a pesquisa sobre viscoelasticidade em tecidos moles desenvolvida pelo professor Paulo Pedro Kenedi, sendo extraída desta os cálculos, as rotinas e as operações secundárias necessárias.

1.1 Revisão Bibliográfica

No ramo da indústria civil, pode-se ver a aplicação da viscoelasticidade em modelos de fluência de estruturas mistas de aço, conforme apresentado por Tareco [19] que usa os modelos lineares de Maxwell e de Kelvin para modelar uma estrutura de aço-betão, analisando a relaxação e a fluência somente do betão na resposta da estrutura mista e dos materiais que a compõem. Além disso, conforme apresentado Souza [20] materiais viscoelásticos também são utilizados para atenuar vibrações e ruídos em estruturas, tendo aplicações tanto no setor automotivo quanto aeroespacial. Com o objetivo de melhor compreender os modelos, R. F. Navarro [7] aborda as combinações possíveis entre as analogias mecânicas mola e amortecedor que caracterizam o comportamento viscoelásticos de materiais reais de uso no ramo industrial.

Pode ver também a viscoelasticidade aplicada a outros materiais, como visto na obra de Weinerowska-Bords [29] que analisou os aspectos relativos ao comportamento viscoelástico de paredes de tubos de polímero durante um fluxo instável, também apresentou e comparou a expressão alternativa linear com convolução do termo viscoelástico com o correspondente que se refere ao atrito instável.

A importância do estudo da viscoelasticidade não se restringe apenas à metais e polímeros, tendo também importâncias no estudo de tendões e ligamentos, conhecidos como tecidos moles, como demonstrado por Rossetto [8] que demonstra que esse conhecimento é de suma importância para melhor compreensão do comportamento dos tendões, permitindo que melhores análises possam ser feitas para determinar o treinamento físico, como nos casos de terapêuticos das tendinopatias. Além disso, também é visto grandes preocupações em se saber as forças aplicadas em tendões, como abordado por Bernardes et. all [9], que buscou determinar os parâmetros biomecânicos para o modelamento da articulação do joelho humano através de exercícios extensivos, junto de imagens raio-X obtidas por videofluoroscopia, e Zheng, et al., que, também em cima do joelho humano, que buscou modelar analiticamente das forças internas durante um exercício físico, além disso, usaram um programa para otimizar os resultados e minimizar os erros.

Devido a sua simplicidade, algumas pesquisas foram feitas com o objetivo de caracterizar as propriedades mecânicas de tecidos moles por meio de modelos lineares de viscoelasticidade. Isto foi feito em Jamison, et al., [30] em que é apresentado um estudo caracterizando os tecidos moles e o modelo supracitado é implementado, apesar do modelo teórico não ser tão compatível com o comportamento real, visto pela investigação experimental dos autores com pele de porco. Na obra de Miller [31], também é usado o modelo linear com o objetivo de aplicar em um software de elementos finitos para caracterizar as propriedades mecânicas de tecidos de cérebro.

Como os modelos lineares são mais simples e não refletem o comportamento real de um material viscoelástico, alguns autores preferiram utilizar o modelo de viscoelasticidade quase-linear apresentado por Fung [3] ou modelos não-lineares por melhor corresponder à realidade. Em Piazza, et al., [24] desenvolveram um modelo dinâmico tridimensional da articulação tibiofemoral e patelofemoral para prever o movimento de implantes no joelho durante atividades de intensidade se baseando no modelo de Fung, em que se utilizou as equações dinâmicas de movimento sujeitas às forças geradas pelos músculos, ligamentos e contato nas articulações tibiofemoral e patelofemoral. Além disso, Abramowitch, et al., [33] usaram o modelo quase-linear para obter uma função de relaxamente que melhor se adequasse aos tecidos moles. Qinwu Xu, et al, [38] propuseram um modelo matemático para o módulo de relaxação e sua solução numérica, se baseando em modelos não-lineares de viscoelasticidade. Este modelo apresentou-se capaz de simular fluência, deformação sinusoidal e dissipação de energia, além de apresentar melhor precisão no ajuste de dados experimentais e estabilidade de predição fora da faixa experimental quando comparado a outros modelos amplamente utilizados, como a série de Prony, presente no modelo de Maxwell generalizado, que será mais bem explicado em próximos capítulos deste trabalho.

Com o objetivo de analisar o comportamento viscoelástico de tendões ao longo do tempo, diversos autores fizeram variados experimentos de modo a observar o comportamento tanto da tensão e quanto da deformação. Bonifasi-Lista, et al., [23] quantificaram a deformação e frequência do ligamento medial colateral (MCL) de um humano aplicando tensões nas direções longitudinais e transversais, e sobre cisalhamento ao longo da direção das fibras, também verificaram se o comportamento viscoelástico depende da direção de aplicação. Duenwald, et al., [22] utilizaram o ligamento medial colateral (MCL) impondo deformações de 1, 2, 3, 4, 5 e 6% com o objetivo de explorar a dependência com o tempo da recuperação deste tendão, já que, de acordo com estes, é uma característica muito importante do ligamento, porém pouco estudada, além disso, para que a o efeito da recuperação fosse observado, mantiveram a deformação em 6% e diminuíram a níveis diferentes de zero. Eles observaram que o relaxamento encontrado contrasta com o modelo de viscoelasticidade quase-linear (QLV), que prevê dependência com o tempo igual para várias deformações, e que os tendões não se recuperaram até os níveis previstos pelos modelos não linear e quase linear, embora tenham tido recuperação parcial.

A preocupação em utilizar recursos computacionais para modelos de viscoelasticidade pode ser visto em diversos artigos, já que faz-se necessário executar cálculos complexos que, em alguns casos, não possuem respostas analíticas. No estudo desenvolvido por Stephanie Salles [6], é abordado os conceitos primordiais para implementar matematicamente a abordagem viscoelástica em tecidos moles para descrever seu comportamento, além de uma revisão bibliográfica detalhada para atingir esse objetivo. Weiss, et al., [21] fizeram uma revisão sobre técnicas antigas e atuais para modelar computacionalmente ligamentos e tendões, demonstrando conceitos relevantes sobre mecânica do contínuo e elementos finitos, além de dar ênfase na influência microestrutural dos tecidos moles. Woo, et al., [25] buscaram obter as cinco constantes para o modelo teórico quase-linear de viscoelasticidade que são para descrever a resposta elástica (constantes A e B) e a função relaxação reduzida (constantes C, e ) através de uma abordagem improvisada que converge para uma única solução com mínima variação. Submeteram seis ligamentos medial colateral de cabra a testes de tensão uniaxial com tempo de rampa na faixa de 18 segundos seguido de uma hora de relaxação, por fim, usaram as equações constitutivas do modelo QLV com convolução para os ajustes de curva. Dortmans, et al. [26] investigaram a divergência entre os parâmetros estimados e reais de um material hipotético com propriedades viscoelásticas quase-lineares, além de perceberem que a constante é a que possui os maiores erros e usarem métodos computacionais para obter estimativas mais precisas. Duenwald, et al. [28] utilizaram métodos numéricos para expandir a janela experimental de tempo para o relaxamento e fluência e demonstraram as vantagens associadas a isso, também apresentaram um estudado sobre o papel do tempo de rampa, relaxação e recuperação no contexto das formulações para os modelos de viscoelasticidade quase-linear e não-linear. Behzad Babaei, et al, [37] desenvolveram um algoritmo eficiente que permite a identificação de espectros de relaxação viscoelástica, sendo testado contra dados experimentais para caracterizar sua robustez e identificar suas limitações e pontos fortes, dessa forma, aplicaram para identificar a resposta viscoelástica do colágeno reconstituído. Parnell, et al., [27] reavaliaram o modelo de Fung para viscoelasticidade quase-linear expondo uma abordagem diferente que produz um comportamento melhorado e oferece um esquema simples para resolver uma ampla gama de modelos por meio de soluções numéricas para a equação de Volterra, além disso, apresentaram que uma série de características negativas encontradas em outras obras, que são atribuídas à Fung, são somente uma consequência da forma como o modelo é aplicado.

1.2 Conceitos fundamentais de viscoelasticidade

Entende-se como viscoelasticidade a propriedade dos materiais que apresentam comportamento viscoso e elástico ao mesmo tempo, sendo um conceito amplamente utilizado em diversos setores da indústria. Elasticidade é a propriedade de um corpo de sofrer deformação quando submetido à tração e retornar ao seu estado de equilíbrio, por outro lado viscosidade é a resistência de um fluido quanto ao escoamento, causando uma velocidade de fluxo maior quanto mais afastado da superfície de contato. Atualmente há diversos modelos que descrevem o comportamento viscoelástico de diferentes materiais, podendo serem divididos em linear, quase-linear e não-linear.

Os primeiros conceitos de viscoelasticidade que podem ser abordados são fluência e relaxação. A fluência ocorre quando aplicada uma tensão constante ao corpo e observa-se uma contínua deformação ao longo do tempo, conforme pode ser visto na figura 1.1(a). Por outro lado, a relaxação ocorre quando aplicada uma deformação constante ao corpo e observa-se a tensão diminuir ao longo do tempo até um determinado patamar e se mantendo constante depois, conforme demonstrado na figura 1.1(b). (Fung, página 41 [3])

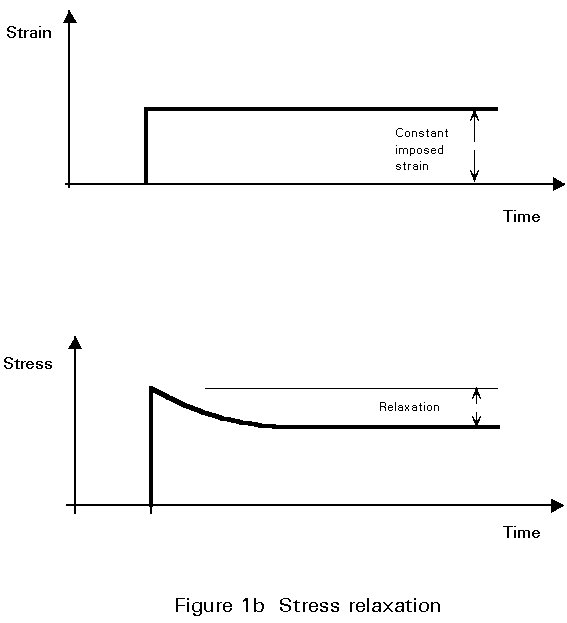
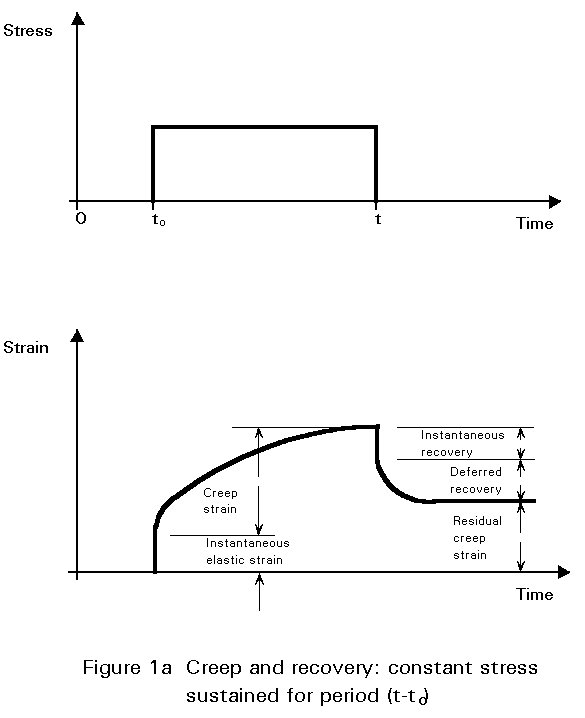


Figura 1.1 – (a) Tensão e deformação em uma situação de fluência e (b) tensão e deformação em uma situação de relaxação [34]

O comportamento linear da deformação no caso de fluência pode ser expresso de acordo com a equação 1.1, que indica que a tensão é constante e a fluência, expressa por J(t), depende somente do tempo.

|  |  |
| --- | --- |
|  | 1.1 |

Já para a tensão, seu comportamento linear no caso de relaxação pode ser expresso de acordo com a equação 1.2, demonstrando que a deformação atua de maneira constante e a função relaxação, expressa por G(t), depende somente do tempo.

|  |  |
| --- | --- |
|  | 1.2 |

Outro conceito importante é o princípio da superposição de Boltzmann que é utilizado em diversos modelos de viscoelasticidade. Segundo este princípio, para o caso unidimensional é considerado uma força F(t) e um alongamento u(t), que é causada por todo o histórico do carregamento até o tempo t. Se a função F(t) for contínua e diferencial, então, em um pequeno intervalo de tempo dτ no tempo t, o incremento do carregamento pode ser expresso por (dF/dτ)dτ. Este incremento continua atuando no material e contribui um elemento du(t) para o alongamento no tempo t, com uma constante de proporcionalidade c dependente do intervalo de tempo t – τ. Isto está descrito abaixo na equação 1.3.

|  |  |
| --- | --- |
|  | 1.3 |

Algo similar pode ser escrita relacionando o comportamento do carregamento a partir da ação de um alongamento, conforme apresentado na equação 1.4, contudo, para esta, será usado uma constante de proporcionalidade k.

|  |  |
| --- | --- |
|  | 1.4 |

Essas formulações também podem ser aplicadas para se obter uma relação entre a deformação e tensão, já que isto possui uma importância maior para os estudos de viscoelasticidade em tecidos moles, sendo estas descritas abaixo. Contudo, vale lembrar que as constantes de proporcionalidades serão diferentes, sendo estas J e G que são, respectivamente, a fluência e função relaxação.

|  |  |
| --- | --- |
|  | 1.5 |
|  | 1.6 |

De acordo com a hipótese de Boltzmann, para o cálculo do efeito total pode-se somar cada diferencial, aplicando uma integral do tempo inicial igual a 0 até o tempo t, conforme descrito abaixo.

|  |  |
| --- | --- |
|  | 1.7 |
|  | 1.8 |

Essas equações são lineares, já que estas dependem somente do tempo, entretanto, também são vistas em modelos quase-lineares e não-lineares aplicadas de maneiras diferentes.

Fung – subida elástica não linear e descida viscoelástica linear

**NÃO ESQUECER DAS EQUAÇÕES DE VISCO (MAXWELL MAIS IMPORTANTE) – DEPOIS DA REVISÃO BIBLIOGRÁFICA**

**REVISÃO BIBLIOGRAFICA**

**MAIOR PARA O MENOR, AFUNILIANDO OS CONCEITOS PARA O QUE VAI SER VISTO NO TCC (20 ou 30 referencias)**

**Um capitulo só de noções fundamentais de visco**

**Mostrar a parte linear que não aplica a tecido mole (FOCAR QUE NÃO SERVE PARA TECIDO MOLE)**

**Maxwell generalizado**

**Explicar cada variável que atua em viscoelasticidade ANTES de falar dos modelos.**

**No final da introdução, explicar as partes de programação usados.**

**Descrever modelo como está na literatura e depois como foi colocado em computação**

1.3 Conceitos fundamentais de programação

Neste estudo, diversos conceitos da área de ciência da computação serão visitados devido ao objetivo deste trabalho, sendo assim, vale citar cada um deles e explicá-los. Sendo o primeiro, a linguagem de programação escolhida, a linguagem C# que, segundo a Microsoft [3], “é uma linguagem de programação moderna, orientada a objeto e de tipo seguro”, sendo comumente utilizada no desenvolvimento de sistemas por empresas por seu poder computacional, além de possuir uma vasta documentação ofertada gratuitamente pela Microsoft e por estar atrelada à metodologia ágil, que também vem sendo utilizado com frequência atualmente.

O próximo conceito é a API, sigla para Application Programming Interface, que traz consigo mais diversos outros conceitos. Segundo a Microsoft, a API [1] “especifica como os componentes e sistemas de software devem interagir uns com os outros” e, segundo TechTudo [2], corresponde a um conjunto de normas que possibilitam a comunicação entre sistemas através de uma série de padrões, como SOLID, que expõe princípios de boas práticas de programação orientada a objeto (POO) de modo a tornar o código limpo, de fácil manutenção e escalável, e protocolos, como HTTP, que determina a estrutura básica de comunicação a ser usada por aplicações WEB. Vale salientar que esse protocolo e esse padrão foram utilizados no desenvolvimento do código para este trabalho com o objetivo de criar um código com grande poder computacional, de fácil compreensão e manutenibilidade.

Por utilizar os protocolos HTTP, pode-se chamar a aplicação desenvolvida para este estudo de API RESTful. [12][14] A sigla REST significa *Representational State Transfer*, que traduzido para o português significa Transferência Representacional de Estado, e, por sua utilização por completo na API, recebe o sufixo “ful”. Devido a isso, foram utilizados alguns recursos essenciais para o correto funcionamento da API RESTful, como entrada e saída de dados usando formatação JSON, chamadas HTTP através de *endpoint*, que possuem verbos HTTP específicos para cada tipo de tarefa, status HTTP na resposta da operação e estrutura de criação de URLs estáticas, sendo escolhido o *framework* ASP.NET MVC. Dessa forma, diversos conceitos são injetados neste contexto, sendo necessário uma melhor explicação sobre cada.

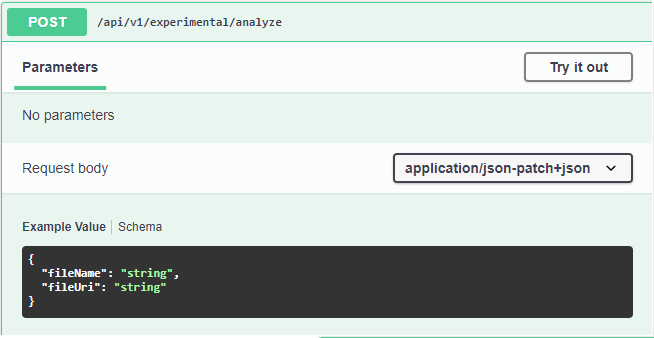
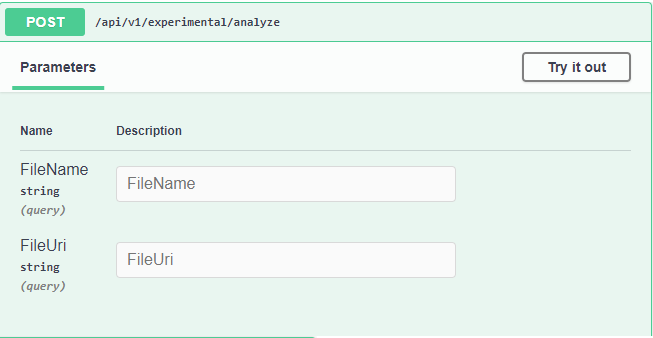
JSON [13], sigla para *JavaScript Object Notation* ou, em português, Notação de Objetos JavaScript, é uma formatação leve de troca de dados, simples de ler e escrever para o ser humano e de fácil interpretação e geração para máquinas, é independente de linguagem, por usar convenções que são familiares a diversas linguagens de programação, conforme pode ser visto na figura 1.1. No presente trabalho, será amplamente utilizada na serialização de objetos, isto é, converter uma cadeia de texto em um objeto utilizado internamente pelo código, e desserialização de objetos que é o caminho inverso da serialização. Ambos são usados, principalmente, pelo Swagger na entrada e saída de dados, respectivamente, conceito que será melhor apresentado mais a diante.

Texto

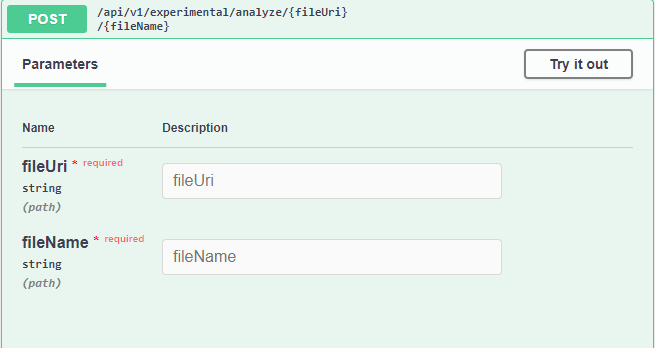
Descrição gerada automaticamente

Figura 1.1 – Exemplo de formatação JSON

O próximo conceito a ser descrito é o *endpoint* [11][12], esse é um termo em inglês que pode ser traduzido para pontos de extremidade, que, no caso abordado neste documento, possui de fato esta característica, sendo o ponto de extremidade da aplicação. Nesse contexto, o *endpoint* corresponderá a uma URL por onde são feitas as requisições possibilitando que as operações em uma API sejam acessadas. Para efetuar essa requisição, é necessário definir dois parâmetros: o verbo HTTP definido pelo programador, que pode ser GET, para uma busca de dados, POST, para criar um dado, PUT, para atualizar algum dado, e DELETE, para excluir um dado; e o conjunto de dados da requisição passado pelo usuário. Este último pode preenchido de três maneiras diferentes: na URL, escrita juntamente à rota do *enpoint*; por *query*, em que no Swagger permite escrever cada parâmetro separado; e no corpo da requisição, usando formatação JSON. Vale salientar que um *endpoint* está atrelado a uma operação que é uma classe que possui as lógicas para a execução de uma determinada rotina e/ou tarefa.

(a) Corpo da requisição no formato JSON (b) Dados de requisição por *query*



(c) Dados de requisição a partir da URL

Figura 1.2 – Formas de passar o conjunto de dados da requisição no Swagger

O próximo conceito é o Swagger [10] que é uma ferramenta *open source* para desenvolvimento de APIs, facilitando no design, construção, documentação, teste e padronização, além disso permite ao usuário acessar os *endpoints* disponíveis através de uma interface simples e de fácil compreensão com documentações, exemplos e explicações sobre os recursos, conforme pode ser visto na figura 1.3.

COLOCAR AQUI UMA IMAGEM DO SWAGGER DA API DEPOIS DE TER AS DOCUMENTAÇÕES NELE

Figura 1.3 – Swagger da API desenvolvida para este estudo

Além disso, este recurso permite expor ao usuário a resposta detalhada da operação, que nada mais é do que uma cópia da estrutura da classe cuja responsabilidade única é conter essas informações. No caso deste estudo, todas as classes de resposta possuem a mesma estrutura com as seguintes propriedades: *Success*, que é do tipo booleano, ou seja, pode ser verdadeiro, quando a operação executar corretamente, e falso, caso falhe; *HttpStatusCode*, que contém o status HTTP da operação; *Errors*, que é uma lista onde são escritos os erros que ocorrem durante a aplicação, juntamente a um código pré-definido para facilitar a compreensão do motivo do erro; e *Data*, que contém os dados principais da resposta, variando através da necessidade de cada operação, já que em cada caso há informações diferentes que são importantes de retornar para o usuário.

Devido a sua grande gama de informações, vale informar que somente alguns status HTTP foram utilizados neste estudo, já que este pode informar diversos acontecimentos com a API, conforme demonstrado pela empresa Runscope [17] em seu site dedicado a isso. Os valores utilizados serão descritos abaixo na tabela 1.1 com sua nomenclatura, significado e dados de resposta obrigatórios.

Tabela 1.1 – Status HTTP utilizados no código

|  |  |  |  |
| --- | --- | --- | --- |
| Status HTTP | Nomenclatura | Significado | Dados de resposta obrigatórios |
| 200 | OK | Operação executada com sucesso. | Não tem. |
| 201 | Created | Arquivo gerado com sucesso. | Caminho do arquivo. |
| 202 | Accepted | Dados de requisição válidos e aceitos para executar a operação. | Não tem. |
| 400 | Bad Request | Dados de requisição inválidos. | Motivo do erro descrevendo quais dados estão inválidos. |
| 401 | Unauthorized | Usuário não tem autorização para acessar o *endpoint*. | Não tem. |
| 404 | Not found | Nenhum dado foi encontrado de acordo com o filtro passado. | Não tem. |
| 500 | Internal Server Error | Erro interno. | Descrição detalhada do motivo do erro. |
| 501 | Not Implemented | Recurso não implementado. | Descrição do recurso que não foi implementado. |

Para que a aplicação possa expor na WEB seus recursos, faz-se necessário a utilização de alguma ferramenta que possibilita isso, sendo escolhida para este trabalho o *framework* ASP.NET MVC, que possibilita criar sites estáticos, sendo interpretado com facilidade pelo Swagger. [15][16] ASP.NET é uma plataforma da Microsoft criada para desenvolvimento WEB e a sigla MVC vem de *Model View Controller*, sendo este conjunto responsável pela apresentação da aplicação, visando criar um código que não possua conexão forte entre as partes, pois facilita a manutenção e adição de funcionalidades. Dentro deste padrão, há três componentes principais: *model*, responsável por representar a entidade da lógica de negócio da aplicação, ou seja, a operação; *view*, responsável por apresentar uma interface para o usuário, que neste caso estará em conjunto com o Swagger; e *controller*, que é responsável pelo controle dos elementos, propiciando uma ligação entre eles. Um exemplo de código escrito no *controller* pode ser visto na figura 1.4, estando destaca em vermelho a parte que representa a operação e em azul a parte que representa os dados de entrada passados pelo usuário através do Swagger.

Texto

Descrição gerada automaticamente

Figura 1.4 – Exemplo de código presente no *controller*

Com o objetivo de simplificar e padronizar o código, foi desenvolvido um conjunto de classes que servem de base para todas as operações, de modo que cada operação da API só precisa implementar seus parâmetros e lógicas específicas, evitando repetições de código. Dessa forma, todas as operações seguem o mesmo fluxograma apresentado na figura 1.5, em que, após a entrada dos dados a partir do Swagger, esses são validados no método *ValidateOperationAsync*, caso sejam válidos, segue para o método *ProcessOperationAsync*, onde a operação será de fato executada e, caso não sejam, encerra o processo retornando mensagens de erro e status HTTP 400. Além disso, foram criados contratos específicos que servem como base para os demais. Entende-se como contratos as classes que contêm os dados da requisição e a resposta e seus dados principais de uma determinada operação, sendo, para o presente trabalho, as classes com os sufixos *Request*, *Response* e *ResponseData*.

Diagrama

Descrição gerada automaticamente

Figura 1.5 – Fluxograma básico das operações

Para tornar a aplicação mais otimizada e rápida, foi utilizado o conceito de paralelismo e assincronismo, otimizando o tempo gasto. Paralelismo diz respeito a executar tarefas em paralelo, ou seja, mais de uma tarefa será executada ao mesmo tempo, enquanto assincronismo representa executar *threads* separadas, isto é, rodar tarefas em ordens de processamento diferentes. Esses conceitos foram aplicados usando a classe *Task* [18], nativa do C#, que possui todos os recursos necessários para esse gerenciamento. Isso foi utilizado em 3 partes do código, sendo estas: na chamada de algumas operações; na iteração dos dados de requisição; e no cálculo de cada parâmetro dos modelos. Estas duas últimas serão mais explicadas no decorrer deste trabalho.

Assincronismo e paralelismo devem ser utilizados com cautela, visto que há um maior consumo de recursos computacionais, logo, deve ser feito uma análise de custo-benefício levando em consideração, principalmente, a quantidade de tarefas em paralela ou assíncronas que são executadas.

EXPLICAR COMO FICA O FLUXO DA OPERAÇÃO QUANDO EXECUTADA DE MANEIRA ASSÍNCRONA E COLOCAR UMA FIGURA DO FLUXOGRAMA.

Figura 1.6 – Fluxograma de uma operação assíncrona

Por último, o conceito de herança precisa ser visitado, já que isto é utilizado por todo o código, além de ser um tópico intrínseco à programação orientada a objetos. Segundo DevMedia [39], “a herança possibilita que as classes compartilhem seus atributos, métodos e outros membros da classe entre si”, isto é, ao definir que uma classe herda da outra, a classe filho ou classe derivada recebe as características da classe pai ou classe base, tendo acesso a estas e podendo manipulá-las. Isso é extremamente utilizado com o intuito de evitar replicações de código, além de impor que, em determinados contextos, alguns comportamentos sejam implementados, como será melhor apresentado mais a frente no capítulo sobre a implementação numérica dos modelos de viscoelasticidade.

2 Recursos implementados

Para que fosse possível atingir o objetivo deste trabalho, diversos recursos foram implementados, sendo estes separados por tópicos, conforme serão demonstrados a seguir.

2.1 Modelos de viscoelasticidade

Para cada modelo de viscoelasticidade, foi necessário implementar duas operações e as equações constitutivas, sendo estas últimas implementadas em classes separadas com o nome de cada modelo. A primeira operação é responsável pelo cálculo das propriedades mecânicas do tecido mole, em que o código recebe e retorna parâmetros específicos de cada modelo, funcionando somente como um orquestrador validando se os valores passados e chamando o método responsável por efetuar os cálculos da classe que contém as equações do modelo, além de escrever esses resultados em um arquivo. A segunda operação é responsável por fazer uma análise de sensibilidade para cada parâmetro do modelo, se baseando na funcionalidade citada anteriormente, diferindo somente nos dados de entrada, em que é possível passar um intervalo de valores para serem testados e comparados os resultados.

2.2 Métodos numéricos

Conforme será demonstrado mais a seguir neste trabalho, as equações para os modelos quase-linear e não linear possuem integrais e derivadas que precisam ser resolvidas numericamente. Posto isto, foi implementado uma classe que contém os métodos para calcular uma determinada integral e outra que contém as lógicas para efetuar a derivada.

A derivada numérica utilizada foi o método de diferenças centrais por possuir uma boa precisão numérica e ser fácil de se implementar, cuja formulação está demonstrada pela equação 2.1.

|  |  |
| --- | --- |
|  | 2.1 |

A integração numérica utilizada foi o método de Simpson composto [35][36] por apresentar uma grande precisão ao permitir que o intervalo seja dividido em espaços muito pequenos. Esse método é aplicado considerando o intervalo [a, b], sendo a e b quaisquer valores reais e que a função f(x) a ser integrada seja contínua dentro deste intervalo, e é adotado N divisões, com e N/2 subintervalos do tipo , com k = 0, 1,..., N/2. Assim temos, considerando os subintervalos, temos:

|  |  |
| --- | --- |
|  | 2.2 |

Aplicando a regra de Simpson simples na integral da direita do subintervalo, podemos reescrever a equação 2.2 conforme abaixo.

|  |  |
| --- | --- |
|  | 2.3 |

Simplificando a equação 2.3, é possível obter a equação final para a Regra de Simpson Composta.

|  |  |
| --- | --- |
|  | 2.4 |

Conforme pode ser observado, para calcular a integral e derivada de uma função no código seria necessário calcular previamente todos os pontos que o método utilizaria. Para evitar que isto fosse feito, foi utilizado a classe Func, nativa do C#, que permite encapsular um método do código do código em uma variável, sem perder suas características.

2.3 Dados experimentais

As operações implementadas para os dados experimentais foram duas baseando-se em duas necessidades: remover pontos indesejáveis e extrapolar os dados.

A primeira é responsável por somente analisar os dados experimentais removendo os pontos considerados inválidos. Isso foi feito se baseando no comportamento esperado das curvas, em que, para o tempo de rampa, é esperado que a tensão aumente, logo, primeira derivada positiva e a concavidade da curva seja para cima, por tanto, segunda derivada positiva, e para a relaxação, a tensão deve diminuir, primeira derivada negativa, e, assim como na anterior, concavidade da curva para cima.

A principal função da segunda operação é extrapolar os dados através do menor valor encontrado para a segunda derivada, que, com isto, prevê a primeira derivada e, em seguida, prevê o valor da tensão para aquele ponto usando, em ambos os casos, método de integração trapezoidal. Contudo, para que isso seja possível, antes da etapa principal, os pontos são analisados utilizando a primeira operação supracitada.

2.4 Operações auxiliares

Além das operações supracitadas, algumas outras foram necessárias para automatizar algumas etapas feitas manualmente, sendo estas: uma para diminuir a quantidade de pontos em um arquivo; uma para buscar arquivos com os resultados de alguma operação nas pastas onde estes são gravados; e outra para baixar um determinado arquivo desejado. Por se tratar de uma Web API, essas duas últimas operações foram desenvolvidas para permitir que o usuário da API possa acessar e baixar quaisquer arquivos previamente gerados por este ou por outrem.

3 Modelo Linear de Viscoelasticidade

Uma função linear é caracterizada por depender somente de uma variável. Posto isto, o modelo de viscoelasticidade linear precisa ter uma função relaxação e uma complacência de fluência lineares, em que ambas dependam somente do tempo. Modelos lineares são mais comumente utilizados para metais, entretanto, devido a sua simplicidade, será utilizado neste trabalho para iniciar os estudos sobre implementação numérica de modelos de viscoelasticidade.

3.1 Maxwell

O modelo de viscoelasticidade linear de Maxwell considera que o tecido mole é representado por um amortecedor e uma mola em série, conforme demonstrado na figura 3.1.

Diagrama

Descrição gerada automaticamente

Figura 3.1 – Representação de mola e amortecedor em série [3]

Observando a figura 3.1, temos que F é a força, que dividida por uma área considerada constante, se torna tensão, µ é o módulo de elasticidade e η é a viscosidade. O deslocamento do amortecedor e da mola são representados, respectivamente, por e . Usando as expressões de elasticidade, viscosidade e equilíbrio de forças e assumindo que as áreas são iguais, temos:

|  |  |
| --- | --- |
|  | 3.1 |
|  | 3.2 |
|  | 3.3 |
|  | 3.4 |

Derivando as equações 3.1 e 3.4, onde o ponto representa a derivada no tempo, temos:

|  |  |
| --- | --- |
|  | 3.5 |
|  | 3.6 |

Substituindo as equações 3.2, 3.3 e 3.5 em 3.6, temos:

|  |  |
| --- | --- |
|  | 3.7 |

Além disso, para este modelo, é considerado como deformação inicial o valor da tensão inicial dividida pelo módulo de elasticidade, conforme visto na equação 3.8.

|  |  |
| --- | --- |
|  | 3.7 |

Sabendo que a partir da equação 3.1, temos μ que é a rigidez da mola, e integrando a equação 3.2, obtemos a rigidez de amortecimento de η / t, a rigidez à fluência pode ser avaliada pela soma da rigidez da mola e do amortecimento. Por fim, a complacência à fluência pode ser avaliada invertendo o resultado de rigidez à fluência, conforme expresso a seguir:

|  |  |
| --- | --- |
|  | 3.8 |

Para a relaxação, o modelo de Maxwell assume a taxa de deformação igual a zero na equação 3.7, possuindo uma equação diferencial de primeira ordem que possui solução, considerando e . Sua resolução está descrita abaixo.

|  |  |
| --- | --- |
|  | 3.9 |

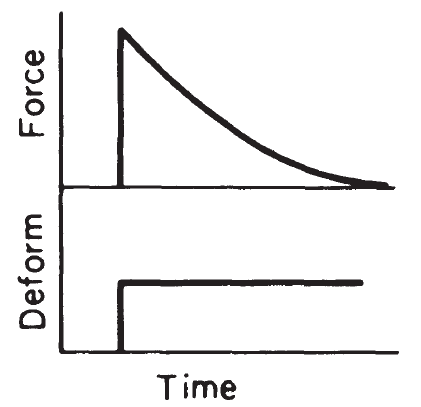
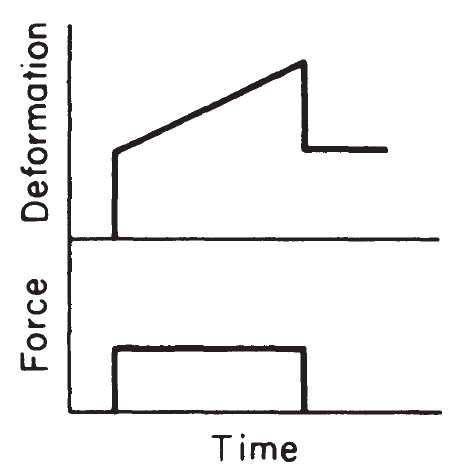
Isolando a tensão inicial da equação 3.7 e a substituindo em 3.9, temos:

|  |  |
| --- | --- |
|  | 3.9 |

Comparando a equação 3.9 com a equação 1.2, podemos definir a função relaxação como:

|  |  |
| --- | --- |
|  | 3.10 |

As curvas de relaxação e fluência para o modelo de Maxwell podem ser melhor observados nas figuras 3.2a e 3.2b, onde pode-se observar o comportamento da tensão e deformação ao longo do tempo. Vale salientar que as imagens foram extraídas de [3] em que é considerado força e alongamento, ao invés de tensão e deformação, como neste estudo, porém, seu comportamento é análogo.



(a) (b)

Figura 3.2 – Força e alongamento ao longo do tempo para (a) relaxação e para (b) fluência

3.2 Maxwell Generalizado

O modelo generalizado de Maxwell propõe que o tecido mole pode ser representado como um conjunto de várias molas e amortecedores em série e todas estas em paralelo com uma mola, conforme pode ser visto na figura 3.3.

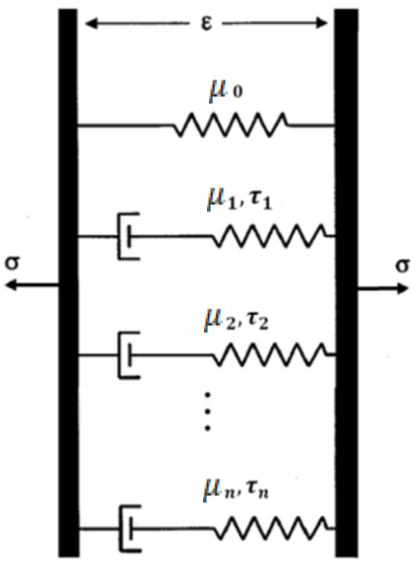


Figura 3.3 – Modelo generalizado de Maxwell [37] com letras alteradas

Este modelo generalizado é capaz de expressar o comportamento de um tecido mole com uma grande quantidade de parâmetros, já que possui mais elementos do que o modelo simples de Maxwell. Mediante isso, a função relaxação para este modelo pode ser expressa conforme a equação abaixo.

|  |  |
| --- | --- |
|  | 3.11 |

Neste caso, a função relaxação também é chamada de série de Prony para relaxação, onde é o módulo de relaxação ou módulo de Young para a situação de equilíbrio, são os múltiplos módulos de Young para cada elemento de mola e são os tempos de relaxação de cada elemento. Conforme demonstrado por Behzad Babaei, et al [37], na prática, essa equação apresenta boa aproximação utilizando somente 3 elementos no somatório, não tendo diferenças físicas e matemáticas consideráveis.

4 Modelo Quase-Linear de Viscoelasticidade de Fung

O modelo de Viscoelasticidade Quase-Linear de Fung aplica a não-linearidade da relação entre tensão e deformação expressando a tensão em duas partes: a função relaxação reduzida, que depende somente do tempo, e a resposta elástica, que depende da deformação, em que as equações utilizadas para cada parâmetro serão demostradas a seguir. Para facilitar os cálculos, este último pode ser expresso dependendo somente do tempo, já que a deformação depende deste parâmetro. Além disso, no modelo de Fung é utilizado uma formulação baseada na equação 1.8 para o cálculo da tensão, fazendo uma consideração análoga, porém usando as duas parcelas citadas acima.

Este é um modelo comumente utilizado para tecidos moles, por se conseguir representar a realidade com boa proximidade conforme será demonstrado mais a diante nas comparações entre os resultados experimentais e numéricos, além de as constantes necessárias para as equações não apresentam grandes dificuldades de serem obtidas, já que não é necessário um grande número de experimentos para obtê-las e possuindo os dados experimentais, basta aplicar ajustes de curva através de softwares como o Origin para calculá-las. Entretanto, assim como qualquer modelo, este apresenta limitações, uma vez que para diferentes relaxações e patamares de deformação são encontrados valores diferentes para estas constantes. Mediante a isto, busca-se, também, analisar se estas constantes apresentam algum padrão em relação aos níveis de deformação impostos ao ligamento.

Ademais, para este trabalho, serão feitas duas ponderações relacionadas ao tempo de rampa: considerar e desconsiderar. Ao considerar o tempo de rampa, é possível considerar diversas relaxações, pela presença das variáveis A e B demonstradas nos cálculos da resposta elástica que só são possíveis de serem obtidas ao considerar este tempo, para este projeto foi considerado somente duas relaxações. Enquanto isso, ao desconsiderar o tempo de rampa, só será possível considerar uma relaxação devido à ausência das variáveis supracitadas.

4.1 DEFORMAÇÃO

Para a deformação, é utilizado a equação 4.1 quando é, de modo a se assemelhar a como é feito experimentalmente no laboratório, e a equação 4.2 quando o tempo de rampa não é considerado. Os parâmetros e representam, respectivamente, a taxa de deformação aplicada durante o experimento, sendo para o caso de aumento na deformação e para o caso de diminuição da deformação, e o tempo de rampa, além disso e representam os tempos em que a deformação é alterada, seja aumentando, diminuindo ou ficando constante.

|  |  |
| --- | --- |
|  | 4.1 |
|  | 4.2 |

Desse modo, pode-se calcular a derivada da deformação, que mais a frente será demonstrada sua importância ao calcular a tensão. Derivando a equação 4.2, temos:

|  |  |
| --- | --- |
|  | 4.3 |

Sendo assim, a derivada somente será diferente de zero quando a deformação não for constante.

4.2 RESPOSTA ELÁSTICA

A resposta elástica pode ser calculada através da equação 4.4, sendo em função da deformação, porém, como supracitado, esta é em função do tempo, a resposta elástica pode ser reescrita também em função somente do tempo. Os parâmetros A e B apresentados na equação representam, respectivamente, *elastic stress constant* e *elastic stress power*. Vale salientar que mesmo que essa propriedade dependa da deformação, que para diferentes intervalos de tempo são usadas diferentes equações, pode-se usar somente a equação 4.4, além disso, observando a equação 4.1, pode-se perceber que em alguns intervalos de tempo a resposta elástica será constante e não dependerá do tempo.

|  |  |
| --- | --- |
|  | 4.4 |

Além disso, para o caso em que o tempo de rampa é desconsiderado, é usado o valor de tensão inicial aplicado ao experimento, .

|  |  |
| --- | --- |
|  | 4.5 |

Assim como feito para a deformação, também será calculado a derivada da resposta elástica, devido ao mesmo motivo, sua importância no cálculo da tensão, sendo assim, derivando a equação 4.4.

|  |  |
| --- | --- |
|  | 4.6 |

Para esta equação, vale a mesma consideração feita para a 4.4, de que, mesmo tendo grandes oscilações das equações usadas para a deformação e sua derivada apresentando, é válido utilizar para fins computacionais a equação 4.6. Porém, vale relembrar que só é diferente de zero para os casos em que a deformação não é constante, sendo assim, é somente nesses momentos em que a derivada da resposta elástica é diferente de zero.

4.3 FUNÇÃO RELAXAÇÃO REDUZIDA

A função relaxação reduzida corresponde a parcela viscosa das equações de viscoelasticidade, podendo ser escrita de duas maneiras segundo Fung [3], a primeira, equação 4.7, é chamada de simplificada por apresentar maior facilidade de ser obtida em experimentos por ser representada por uma séria de somas de exponenciais, enquanto a segunda, equação 4.8, foi obtida a partir de diversos cálculos matemáticos mais complexos baseados no modelo de Kelvin (*standard linear solid*). Vale ressaltar que para o presente trabalho será considerado que a equação 4.7 é composta por somente três somas de exponenciais, conforme citado anteriormente para o modelo de Maxwell generalizado. Ademais, vale lembrar que para o tempo igual a zero, a função relaxação reduzida é igual a 1, .

Os parâmetros C e correspondem, respectivamente, a amplitude de relaxação e o tempo de relaxação, para a equação 4.7, os índices são somente para diferenciar cara constante, enquanto para a equação 4.8, representa o tempo de relaxação rápido e , o tempo de relaxação lendo, sendo sempre maior do que .

|  |  |
| --- | --- |
|  | 4.7 |
|  | 4.8 |

Como , pode-se afirmar que , sendo assim, pode ser reescrito como:

Além disso, temos que:

Logo:

Por fim, temos que a equação 4.8, pode ser reescrita como:

Simplificando, temos:

|  |  |
| --- | --- |
|  | 4.9 |

Como feito nas propriedades anteriores, será calculada a derivada da função relaxação reduzida para as equações 4.7 e 4.9.

Derivando a equação 4.7:

|  |  |
| --- | --- |
|  | 4.10 |

Derivando a equação 4.7:

Aplicando a definição de cálculo para derivada de uma integral determinada:

Em que:

Temos que:

Logo:

Portanto:

|  |  |
| --- | --- |
|  | 4.11 |

4.4 TENSÃO

Para se calcular a tensão existem 3 diferentes formas de acordo com Fung [3], sendo estas expressas abaixo.

|  |  |
| --- | --- |
|  | 4.12 |
|  | 4.13 |
|  | 4.14 |

Conforme demonstrado anteriormente, a resposta elástica e a função relaxação reduzida dependem somente do tempo, o que permite reescrever as derivadas parciais como derivadas totais. Ademais, temos que e .

|  |  |
| --- | --- |
|  | 4.15 |
|  | 4.16 |
|  | 4.17 |

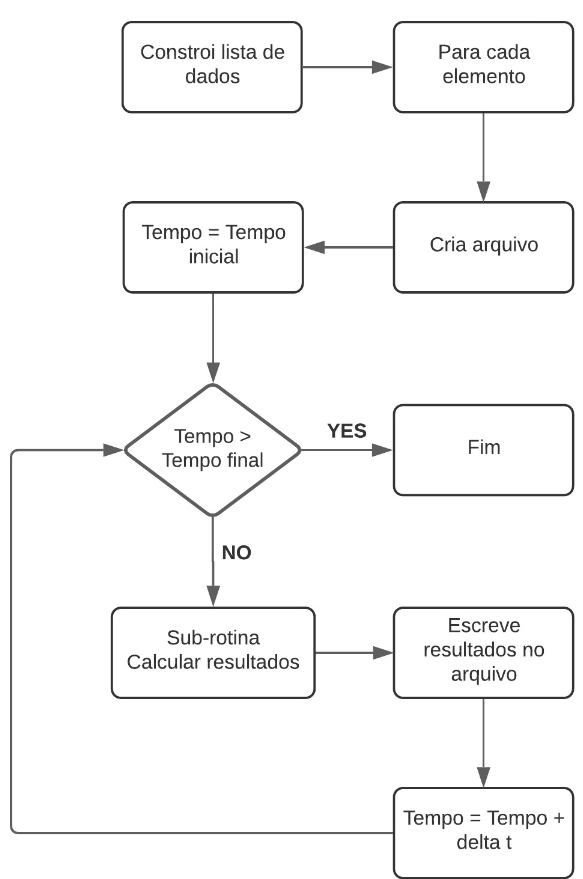
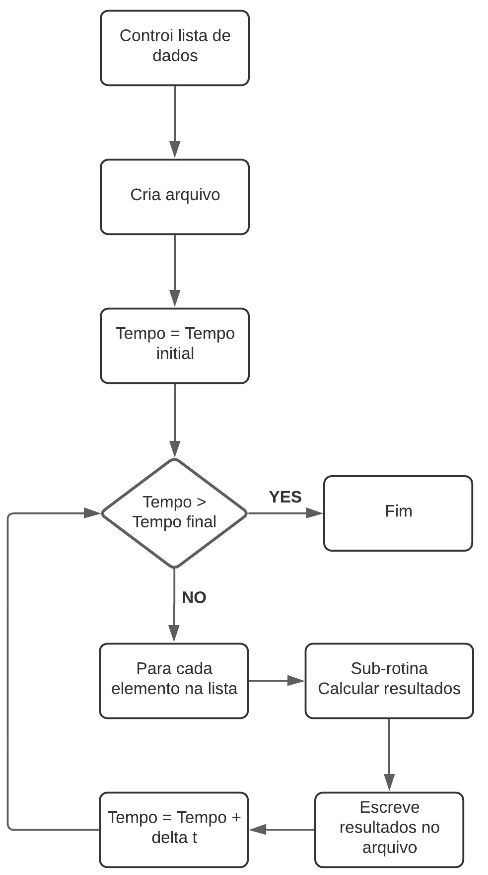
Segundo Fung [3], essas equações são matematicamente equivalentes, logo, em teoria, devem retornar os mesmos valores ou resultados muito próximos. Sendo assim, optou-se por implementar essas três equações retornando seus valores no mesmo arquivo de resposta. Entretanto, vale ressaltar que a equação 4.15 é amplamente utilizada em diversas outras teses, sendo considerada a principal para este estudo e utilizada como base para comparação dos resultados obtidos nas demais equações. Conforme será apresentado a seguir, a utilização destas equações sem quaisquer tratamentos não apresenta viabilidade por impactar negativamente no tempo de execução do código, sendo necessário reescrevê-las com o objetivo de diminuir otimizar o intervalo de tempo da integral. Isso está descrito com melhores detalhes do Anexo 1 deste documento.

5 IMPLEMENTAÇÃO NUMÉRICA

A implementação numérica de todos os modelos apresentados neste trabalho foi feita de maneira similar, sendo feita em duas partes: criação de uma classe que contém as equações constitutivas do modelo, sendo chamada de modelo neste capítulo para facilitar sua citação, e criação das operações para efetuar os cálculos necessários. Isso foi com o objetivo de evitar repetições de código e torná-lo mais legível e de fácil manutenção, de modo a atender os padrões de desenvolvimento de software citados na introdução deste documento, além de se basear no conceito de responsabilidade única, um dos princípios SOLID para programação orientada a objetos, em que cada classe deve ter somente uma responsabilidade.

Durante o desenvolvimento, percebeu-se que há parâmetros em comum que precisam ser calculados, dessa forma, para implementar as equações constitutivas dos modelos, foram criadas abstrações que definem o que cada modelo precisa implementar. Posto isto, no código, pode-se observar separação entre os modelos linear e quase-linear, em que cada um possui uma classe abstrata própria, além de ter sido criada uma classe abstrata que estas herdam que define os métodos que todos os modelos de viscoelasticidade devem possuir, sendo um para calcular cada um dos seguintes parâmetros a deformação, a tensão, as condições iniciais e o resultado.

Além disso, as operações que calculam os resultados e fazem a análise de sensibilidade funcionam da mesma maneira para cada modelo, com fluxograma demonstrado nas figuras 5.1a e 5.1b. As únicas mudanças nestas são para definir quais dados de entrada devem ser utilizados, alterando a forma como é construída a lista de dados, e qual interface para o modelo deve ser utilizada, alterando as contas realizadas dentro da sub-rotina “Calcular resultados”.

(a) (b)

Figura 5.1 – Fluxograma das operações de (a) calcular os resultados do modelo e (b) fazer análise de sensibilidade das variáveis

Vale lembrar, que, conforme pode ser observado nos fluxogramas, os dados de resposta são retornados da mesma forma, em um arquivo de formato CSV, para que este possa ser utilizado em algum outro software de geração gráfica, como Origin.

Conforme pode ser visto nos fluxogramas acima, é necessário construir a lista de dados de entrada a ser utilizada para efetuar os cálculos na sub-rotina, também vista nos fluxogramas. Isso foi feito para seguir o princípio de responsabilidade única apresentado pelo padrão SOLID, já que os contratos das operações devem ser manipulados somente por esta e classes externas não devem ter conhecimento sobre estes contratos. Posto isso, fez-se necessário criar classes que atuariam como contratos para os modelos, em que existe uma responsável por conter os dados de entrada e outra, os resultados dos cálculos. Desse jeito, pode-se observar o motivo da etapa de construir a lista de dados, em que os dados de requisição da operação são mapeados para o contrato do modelo. Mesmo isso conferindo maior complexidade ao código, essa decisão facilitou as manutenções necessárias no código, além de ter permitido escalabilidade do código e adição de recursos com grande facilidade. Vale salientar que a iteração dentro desta lista é feita de maneira assíncrona usando a classe *Task*, nativa do C#, isso para reduzir o tempo de execução, visto que, antes dessa alteração, em casos extremos chegou a durar horas e o tempo médio era na faixa de minutos, enquanto, com essa mudança, os tempos máximos de execução são na ordem de segundos e o tempo médio na ordem de décimos de segundo.

Em cada operação os dados de saída são escritos de maneiras diferentes no arquivo. Para a operação que gera os resultados de um determinado modelo de viscoelasticidade, os resultados são escritos em arquivos separados para cada elemento dentro da lista criada inicialmente, já para a operação que faz análise de sensibilidade das variáveis, todos os resultados são escritos no mesmo arquivo, sendo escrito em colunas diferentes para dados de entrada diferentes. Isso é feito pois o objetivo principal desta última é a comparação de resultados.

A sub-rotina “Calcula resultados” está implementada na operação em uma única linha em que é feita uma chamada ao método do modelo que calcula os resultados, deixando todas as lógicas de cálculos dentro do modelo, conforme citado anteriormente.

5.1 Modelo Linear

Para os modelos lineares de viscoelasticidade, os dados de entrada possuem a viscosidade e a rigidez do material, além da tensão inicial, quando deseja calcular a deformação, e da deformação inicial, quando quer calcular a tensão. O fluxograma para as sub-rotinas para esses cálculos pode ser visto na figura 5.2.

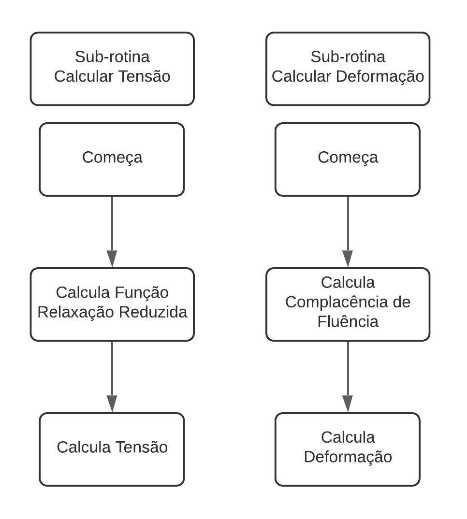


Figura 5.2 – Fluxograma para as sub-rotinas que calculam a tensão e deformação.

Os modelos lineares possuem dois métodos específicos que precisam ser implementados, que são para calcular função relaxação e outro para calcular a complacência de fluência. Vale salientar que, como é utilizado a herança entre as classes, um determinado modelo linear só precisa implementar o que for específico a este, logo, pode-se utilizar as lógicas presentes na classe mais genérica. Os dados de resposta para este modelo são deformação, tensão, função relaxação e complacência de fluência.

5.2 Modelo Quase-Linear

Os modelos quase-linear apresentam maior complexidade na estruturação dos dados de entrada, uma vez que, para cada consideração feita, é passado parâmetros diferentes, logo, a etapa de construir a lista de dados, visto nas figuras 5.1a e 5.1b, é alterada para cada consideração para que seja possível mapear esses valores corretamente.

Ao considerar o tempo de rampa, é necessário passar na requisição da operação as constantes A e B para que seja possível calcular a resposta elástica, entretanto, ao desconsiderar, é necessário informar somente a tensão inicial, em que será considerado que este é o valor da resposta elástica e esta é constante por todo o domínio de tempo. Como para calcular a função relaxação reduzida há duas equações (4.7 e 4.9) e cada uma apresenta constantes distintas que são necessárias, foi feito contratos distintos para cada uma. Dessa forma, os possíveis contratos são uma combinação dessas considerações.

Os modelos quase-lineares possuem diversos métodos específicos, sendo um para calcular cada um dos seguintes parâmetros: derivada da deformação; resposta elástica e sua derivada; função relaxação reduzida e sua derivada; e tensão, sendo um método para cada equação apresentada anteriormente (4.15, 4.16 e 4.17). No decorrer desta implementação, percebeu-se que as equações utilizadas para a tensão precisavam ser otimizadas com o objetivo de reduzir o tempo de execução. A primeira solução adotada foi utilizar o conceito de paralelismo, executando em paralelo os cálculos dos parâmetros que compõem os resultados do modelo, sendo estes: deformação, função relaxação reduzida, resposta elástica e as 3 formas de calcular a tensão. Isso gerou uma grande redução no tempo de execução aliado à outra alteração que aplica assincronismo ao código citada anteriormente. Sendo assim, o fluxograma para a sub-rotina que calcula os resultados pode ser visto na figura 5.3.

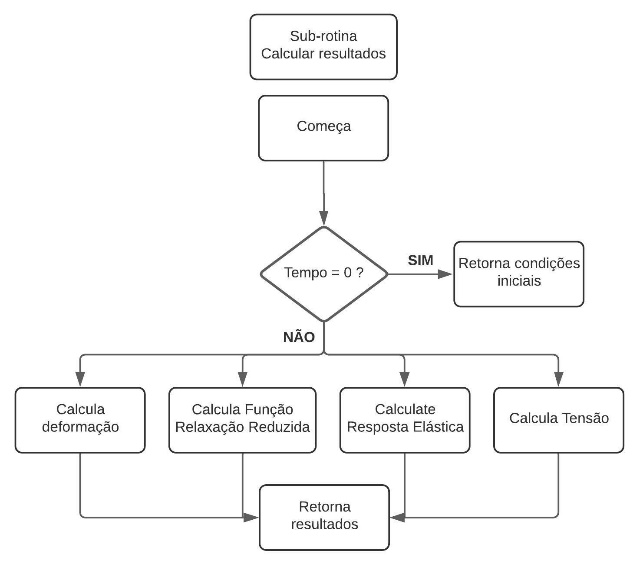


Figura 5.3 – Fluxograma da sub-rotina que calcula os resultados para modelo Quase-Linear

A segunda solução adotada foi reescrever as equações utilizadas com objetivo de encontrar equações específicas para cada intervalo de tempo, conforme demonstrado no Anexo 1 deste documento. Estas propostas de fato auxiliaram e atingiram seu objetivo, entretanto, para a segunda solução, inicialmente eram utilizadas vários *ifs* no código o que o tornava difícil de manutenção e prejudicava a compreensão. Para sanar este problema, aplicou-se o *design pattern* *Strategy*, que, segundo Refactoring Guru [4], “é um padrão de projeto comportamental que permite que você defina uma família de algoritmos, coloque-os em classes separadas, e faça os objetos deles intercambiáveis”, isto é, é um padrão de projeto que permite a segmentação do código de modo a ter partes separadas com lógicas diferentes e que são escolhidas baseadas no contexto. Sendo assim, foi desenvolvido uma classe para cada intervalo de tempo contendo as equações específicas deste período, além de um orquestrador, que baseado no tempo decide qual classe deve ser utilizada. Vale lembrar que todo o código utilizado para este trabalho se encontra no GitHub, no repositório referenciado neste documento [5].

Bibliografia

[1] (27/04/2021) <https://azure.microsoft.com/mediahandler/files/resourcefiles/apis-microservices-ebook/Azure_API-Microservices_eBook.pdf>

[2] (27/04/2021) <https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml>

[3] Livro do Fung

[4] (27/04/2021) <https://refactoring.guru/pt-br/design-patterns/strategy>

[5] <https://github.com/M3110/SoftTissue>

[6] RESEARCH ON VISCOELASTICITY MODELS FOR HUMAN KNEE LIGAMENTS

[7] Modelos Viscoelásticos Aplicáveis a Materiais Reais: uma Revisão

[8] A viscosidade no alongamento de tendões (IMPRESSO)

[9] Determinação de parâmetros biomecânicos para o modelamento da articulação do joelho (IMPRESSO).

[10] <https://swagger.io/about/> (29/04/2021)

[11] <https://milvus.com.br/o-que-e-endpoint/> (29/04/2021)

[12] <https://www.hostgator.com.br/blog/api-restful/> (29/04/2021)

[13] <https://www.json.org/json-pt.html#:~:text=JSON%20(JavaScript%20Object%20Notation%20%2D%20Nota%C3%A7%C3%A3o,3a%20Edi%C3%A7%C3%A3o%20%2DDezembro%20%2D%201999>. (29/04/2021)

[14] <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api> (29/04/2021)

[15] <https://docs.microsoft.com/pt-br/aspnet/overview> (29/04/2021)

[16] devmedia.com.br/introducao-ao-asp-net-mvc/31878 (29/04/2021)

[17] <https://httpstatuses.com/> (30/04/2021)

[18] <https://docs.microsoft.com/pt-br/dotnet/api/system.threading.tasks.task?view=net-5.0> (04/05/2021)

[19] Conceitos de viscoelasticidade – Tareco

Site: <https://run.unl.pt/bitstream/10362/12481/1/Tareco_2014.pdf> (04/05/2021)

[20] QUEIROZ, José Aparecido Silva de. Análise de estruturas flexíveis com aplicação de materiais viscoelásticos. 2008. 101 f. Dissertação (mestrado) - Universidade Estadual Paulista, Faculdade de Engenharia de Bauru, 2008.

Site: https://repositorioslatinoamericanos.uchile.cl/handle/2250/2568006

[21] Computational Modeling of Ligament Mechanics, Jeffrey A. Weiss\* and John C. Gardiner, University of Utah, Department of Bioengineering, Salt Lake City, Utah 84112. 2001

[22] Viscoelastic Relaxation and Recovery of Tendon - SARAH E. DUENWALD

[23] Viscoelastic properties of the human medial collateral ligament under longitudinal, transverse and shear loading - Carlos Bonifasi-Lista

[24] Three-Dimensional Dynamic Simulation of Total Knee Replacement Motion During a Step-Up Task - Stephen J. Piazza

[25] An Improved Method to Analyze the Stress Relaxation of Ligaments Following a Finite Ramp Time Based on the Quasi-Linear Viscoelastic Theory - Steven D. Abramowitch e Savio L.-Y. Woo - DOI: 10.1115/1.1645528

[26] Parameter estimation using the Quase-Linear Viscoelastic Model Proposed by Fung – Dortmans, et al.

[27] On nonlinear viscoelastic deformations: a reappraisal of Fung’s quasi-linear viscoelastic model - Riccardo De Pascalis, I. David Abrahams and William J. Parnell

[28] Constitutive equations for ligament and Other soft tissue: evaluation by experimente – Duenwald, et al.

[29] Alternative Approach to Convolution Term of Viscoelasticity in Equations of Unsteady Pipe Flow - Katarzyna Weinerowska-Bords

[30] Jamison, C. E., Marangoni, R. D., & Glaser, A. A. (1968). Viscoelastic properties of soft tissue by discrete model characterization. Journal of Engineering for Industry.

[31] Constitutive model of brain tissue suitable for finite element analysis of surgical procedures – Miller

[33] Abramowitch, S. D., Woo, S. L.-Y., Clineff, T. D., & Debski, R. E. (2004). An evaluation of the quasi-linear viscoelastic properties of the healing medial collateral ligament in a goat model. Annals of Biomedical Engineering.

[34] COMPOSITE CONSTRUCTION. (2020). Fonte: <http://fgg-web.fgg.uni-lj.si/~/pmoze/ESDEP/master/wg10/l0520.htm>

[35] <https://www.math.tecnico.ulisboa.pt/~calves/courses/integra/capiii33.html#:~:text=Regra%20de%20Simpson%20aplicada%20a%20dois%20sub%2Dintervalos.&text=Assim%2C%20podemos%20considerar%20tr%C3%AAs%20n%C3%B3s,cada%20um%20destes%20sub%2Dintervalos>. (18/05/2021)

[36] <https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/in-regras_compostas.html> (18/05/2021)

[37] Efficient and optimized identification of generalized Maxwell viscoelastic relaxation spectra - Behzad Babaei, et al

[38] A mathematical model for fitting and predicting relaxation modulus and simulating viscoelastic responses - Qinwu Xu and Björn Engquist

[39] <https://www.devmedia.com.br/conceitos-e-exemplos-heranca-programacao-orientada-a-objetos-parte-1/18579> (23/05/2021)

ANEXO 1 – Descrição das equações usadas para modelo de viscoelasticidade Quase-Linear

Equações usadas para deformação e resposta elástica:

A1.1 Considerando tempo de rampa

A1.1.1

A1.1.1.1

|  |  |
| --- | --- |
|  | A1.1 |

A1.1.1.2

|  |  |
| --- | --- |
|  | A1.2 |

A1.1.1.3

|  |  |
| --- | --- |
|  | A1.3 |

A1.1.1.4

|  |  |
| --- | --- |
|  | A1.4 |

A1.1.1.5

|  |  |
| --- | --- |
|  | A1.5 |

A1.1.1.6

|  |  |
| --- | --- |
|  | A1.6 |

A1.1.2 e

Para estas equações, não foram encontradas vantagens em reescrevê-las por terem a parcela , que dificultou drasticamente, já que a equação utilizada para a resposta elástica não dependeria somente do limite de integração ou somente do tempo, fazendo com que, em um limite de integração arbitrário, mais de uma equação para a resposta elástica poderia ser utilizada, diferente do que ocorria a equação .

A1.2 Desconsiderando tempo de rampa

Ao desconsiderar o tempo de rampa, temos que a resposta elástica sempre será constante, logo, sua derivada é igual a zero.

A1.2.1

Para esta equação, não será possível utilizá-la já que, como a resposta elástica é constante, sua derivada será igual a zero, logo, para todo o domínio de tempo, essa equação será igual a zero.

A1.2.2

A1.2.3