ANÁLISE DINÂMICA DE EDIFÍCIOS EQUIPADOS COM AMORTECEDORES DE LÍQUIDO SINTONIZADO ASSISTIDA PELO SOFTWARE DYNAPY

Mario Raul Freitas

MONOGRAFIA DE PROJETO FINAL EM ESTRUTURAS E CONSTRUÇÃO CIVIL

DEPARTAMENTO DE ENGENHARIA CIVIL

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

ANÁLISE DINÂMICA DE EDIFÍCIOS EQUIPADOS COM AMORTECEDORES DE LÍQUIDO SINTONIZADO ASSISTIDA PELO SOFTWARE DYNAPY

MARIO RAUL FREITAS

ORIENTADOR: LINEU JOSÉ PEDROSO

MONOGRAFIA DE PROJETO FINAL EM ESTRUTURAS E CONSTRUÇÃO CIVIL

BRASÍLIA / DF: JULHO/2017

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

ANÁLISE DINÂMICA DE EDIFÍCIOS EQUIPADOS COM AMORTECEDORES DE LÍQUIDO SINTONIZADO ASSISTIDA PELO SOFTWARE DYNAPY

MARIO RAUL FREITAS

MONOGRAFIA DE PROJETO FINAL SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM ENGENHARIA CIVIL.

APROVADA POR:

_____
Prof. Lineu José Pedroso, PhD (ENC/UnB)
(ORIENTADOR)

_____
Luciano Mendes Bezerra, PhD (ENC/UnB)
(EXAMINADOR INTERNO)

_____
William Taylor Matias Silva, PhD (ENC/UnB)
(EXAMINADOR INTERNO)

DATA: BRASÍLIA/DF, 11 DE JULHO DE 2017.

FICHA CATALOGRÁFICA

FREITAS, MARIO RAUL
Análise Dinâmica de Edifícios Equipados Com Amortecedores De Líquido Sintonizado Assistida Pelo Software Dynapy [Distrito Federal] 2016.
xii, 104 p., 297 mm (ENC/FT/UnB, Bacharel, Engenharia Civil, 2016)
Monografia de Projeto Final - Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Civil e Ambiental.
1. Amortecedor de Líquido Sintonizado 2. Dinâmica das Estruturas
3. Sismos 4. Python
I. ENC/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

FREITAS, M.R. (2016). Análise Dinâmica de Edifícios Equipados Com Amortecedores De Líquido Sintonizado Assistida Pelo Software Dynapy. Monografia de Projeto Final, Publicação G.PF-001/90, Departamento de Engenharia Civil e Ambiental, Universidade de Brasília, Brasília, DF, 104 p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Mario Raul Freitas
TÍTULO DA MONOGRAFIA DE PROJETO FINAL: Análise Dinâmica de Edifícios Equipados Com Amortecedores De Líquido Sintonizado Assistida Pelo Software Dynapy
GRAU / ANO: Bacharel em Engenharia Civil / 2017

_____

Mario Raul Freitas
Rua das Paineiras Lote 05 Apt. 604
71929-180 – Águas Claras/DF - Brasil

AGRADECIMENTOS

Agradeço aos meus professores e colegas que me incentivaram e me apresentaram assuntos correlacionados a esse projeto. Agradeço em especial também ao professor Michael Tait, da McMaster University, que me introduziu ao conceito de Amortecedores de Líquido Sintonizados e me acolheu em suas pesquisas, permitindo que eu tivesse contado direto e ajudasse na realização dos experimentos. Agradeço também à minha família, que sempre me apoiou a continuar estudando e alcançar todos os meus objetivos.

“No problem can be solved from the same level of consciousness that created it.”
(Albert Einstein)

RESUMO

Esse trabalho tem por objetivo estudar a atenuação de vibrações em estruturas do tipo *shear building* equipadas com um Amortecedor de Coluna de Líquido Sintonizado e os efeitos da variação paramétrica nessa atenuação. O modelo de amortecedor utilizado possui formato de tubo em U e é montado sobre uma estrutura hipotética com o objetivo de contrabalancear o movimento desta. São estudados diferentes tipos de amortecedor, como um com as extremidades do tubo abertas e outro com uma câmara de ar comprimido em cada uma delas. O sistema estrutura-amortecedor é excitado por um sismo de projeto simplificado. Estuda-se, portanto, o efeito do amortecimento em diferentes frequências de excitação e para diferentes dimensões do amortecedor. A modelagem é feita considerando-se que as lajes do *shear building* são rígidas e concentram toda a massa da estrutura. Dessa forma, cada andar da estrutura possui apenas um grau de liberdade e o amortecedor em si representa mais um grau de liberdade. As equações de movimento são obtidas por meio da análise estrutural e da mecânica dos fluidos. Essas equações são resolvidas utilizando-se o Método das Diferenças Finitas implementado no software DynaPy, desenvolvido nessa pesquisa. Esse se trata de um método bastante simples, eficiente e de fácil implementação computacional para a resolução de equações diferenciais. Os resultados do software são validados pela solução analítica de problemas mais simples e pela comparação com os resultados do SAP 2000 para casos mais complexos. O pós-processamento é feito através da biblioteca Matplotlib do Python. Por fim, busca-se o dimensionamento ideal do amortecedor para a dada situação de projeto

Palavras-chave: Amortecedor de Coluna de Líquido Sintonizado, Dinâmica das Estruturas, Sismos, Método das Diferenças Finitas, DynaPy.

ABSTRACT

This work aims to study the attenuation of vibrations in shear building structures equipped with a Tuned Liquid Column Damper and the effects of parametric variation on this attenuation. The damper model used in this study is U-shaped. It is mounted on top of a hypothetical structure to counterbalance the structure movement. Different models of damper are studied, like one with open ends and another with a compressed air chamber on each end. The structure-damper system is vibrated by a simplified design quake. The effects of damping are studied for different excitation frequencies and different damper dimensions. Modelling is made considering that the shear building slabs are rigid and contain the entire mass of the structures. This way, each story contains only one degree of freedom and the damper itself contains another one. The equations of motion are obtained by structural analysis and fluid dynamics. These equations are solved using the Finite Differences Method implemented in the software DynaPy, which was developed in the research process. This is a very simple, efficient and easy to implement on a script method that solves differential equations. Results from the software are validated by solving simpler problems analytically and by comparison with results from SAP 2000 for more complex cases. Post-processing is done through Python's library Matplotlib. At last, the ideal design of the damper is obtained for both damper models.

Keywords: Tuned Liquid Column Damper, Structural Dynamics, Earthquake, Finite Differences Method, DynaPy.

SUMÁRIO

ÍNDICE DE FIGURAS

SIMBOLOGIA

A – Área da seção transversal do TLCD
A_c – Área do diafragma do TLCD
b – Comprimento do TLCD
c – Coeficiente de amortecimento do sistema
c_{dk} - Parte constante do coeficiente de amortecimento do fluido devido ao diafragma
$c_{d\dot{x}}$ – Fator de correção do coeficiente de amortecimento do fluido devido ao diafragma
c_k - Parte constante do coeficiente de amortecimento do fluido devido às perdas distribuídas
$c_{\dot{x}}$ – Fator de correção do coeficiente de amortecimento do fluido devido às perdas distribuídas
D – Diâmetro do TLCD (modelo simples)
DMF – Fator de Amplificação Dinâmica (Dynamic Magnification Factor)
E – Módulo de elasticidade do material
f – Coeficiente de atrito do escoamento
F_I – Forças de inércia
F_d – Forças de amortecimento
F_S – Forças elásticas
g – Aceleração da gravidade
h – Altura de lâmina d'água no TLCD em repouso
H – Altura do pilar da estrutura
I – Momento de inércia da seção transversal
k – Coeficiente de rigidez do sistema
k_{ar} – Coeficiente de rigidez da câmara de ar
k_f – Coeficiente de rigidez do fluido
k_s – Coeficiente de rigidez da estrutura
L – Comprimento total de líquido no TLCD
m – Massa do sistema
m_f – Massa do fluido
m_s – Massa da estrutura
P – Forças externas
P_0 – Amplitude da forças externas
P_{eq} – Força equivalente de um sismo
r – Relação de frequências
Re – Número de Rynolds
x – Deslocamento do sistema
$\dot{x}$ – Velocidade do sistema
$\ddot{x}$ – Aceleração do sistema
x_h – Deslocamento do sistema – parcela transiente
x_p – Deslocamento do sistema – parcela permanente
x_f – Deslocamento vertical do fluido
$\dot{x}_f$ – Velocidade vertical do fluido
$\ddot{x}_f$ – Aceleração vertical do fluido

$\ddot{x}_q$ – Aceleração horizontal do sismo
x_s – Deslocamento horizontal da estrutura
$\dot{x}_s$ – Velocidade horizontal da estrutura
$\ddot{x}_s$ – Aceleração horizontal da estrutura
X_d – Máximo deslocamento da estrutura sob carga dinâmica
X_e – Máximo deslocamento da estrutura sob carga estática
Δh – Perda de carga do fluido
Δh_d – Perda de carga do fluido devido ao diafragma
Δt – Passo de tempo
ε – Rugosidade absoluta do tubo do TLCD
ζ – Razão de amortecimento crítico da estrutura
ρ_f – Massa específica da água
ν_f – Viscosidade cinemática da água
ω – Frequência natural de vibração da estrutura
ω_D – Frequência natural de vibração amortecida da estrutura
Ω – Frequência de excitação da força externa

$[A]$ – Representação de matriz
$\{A\}$ – Representação de vetor

$[C_f]$ – Submatriz genéria de amortecimento do fluido
$[C_s]$ – Submatriz genéria de amortecimento da estrutura
$[K_f]$ – Submatriz genéria de rigidez do fluido
$[K_s]$ – Submatriz genérica de rigidez da estrutura
$[M_f]$ – Submatriz genérica de massa do fluido
$[M_s]$ – Submatriz genérica de massa da estrutura
$[O^*]$ – Submatriz genéria de acoplamento de massas
$[0]$ – Submatriz nula

$\{\bar{P}_{eq}(t)\}$ – Subvetor genérico de força
$\{X_f(t)\}$ – Subvetor genérico de deslocamento do fluido
$\{X_s(t)\}$ – Subvetor genérico de deslocamento da estrutura
$\{\dot{X}_f(t)\}$ – Subvetor genérico de velocidade do fluido
$\{\dot{X}_s(t)\}$ – Subvetor genérico de velocidade da estrutura
$\{\ddot{X}_f(t)\}$ – Subvetor genérico de aceleração do fluido
$\{\ddot{X}_s(t)\}$ – Subvetor genérico de aceleração da estrutura
$\{0\}$ – Subvetor nulo

TLD – Amortecedor de Líquido Sintonizado (*Tuned Liquid Damper*)
TLCD – Amortecedor de Coluna Líquido Sintonizado (*Tuned Liquid Column Damper*)
CTLCD – Amortecedor de Coluna Líquido Sintonizado Circular (*Circular Tuned Liquid Column Damper*)
DTLCD – Amortecedor de Coluna Líquido Sintonizado Duplo (*Duble Tuned Liquid Column Damper*)

PTLCD – Amortecedor de Coluna Líquido Sintonizado Pressurizado (*Pressurized Tuned Liquid Column Damper*)
LCVA – Amortecedor de Líquido Absorsor de Vibrações (*Liquid Column Vibration Absorber*)

1. Introdução

1.1. Generalidades

Desde meados do século XIX, o avanço da urbanização e a crescente limitação de espaço nas principais cidades do mundo criou a necessidade de se construir prédios mais altos para comportar mais residências, comércios e escritórios. O aperfeiçoamento da segurança dos elevadores por Elisha Otis e o desenvolvimento do concreto armado e das estruturas metálicas permitiu que isso acontecesse, iniciando assim o movimento de construção de "arranha-céus". Dessa forma, as estruturas foram se tornado cada vez mais altas e esbeltas, o que trouxe a elas problemas de vibração, principalmente por sismos e ventos.

A vibração de uma estrutura pode provocar mal estar e desconforto nas pessoas assim como, em casos mais extremos, danos estruturais e até mesmo o colapso total da estrutura. Projetos executados em regiões com grande incidência de sismos e ventos devem considerar o efeito de cargas dinâmicas, pois essas, em muitas circunstâncias, podem se tornar o critério crítico de dimensionamento. Edifícios construídos em regiões onde não há a ocorrência de sismos ou de vento de grande intensidade ainda assim podem precisar de uma análise dinâmica, pois certos tipos de utilização também podem provocar vibrações consideráveis. Esse é o caso de estádios de futebol, academias de dança e ginásios, por exemplo.

Frente a isso, engenheiros ao redor do mundo passaram a buscar soluções como o Amortecedor de Massa Sintonizado (TMD – *Tuned Mass Damper*) e o Amortecedor de Líquido Sintonizado (TLD – *Tuned Liquid Damper*). Essas são soluções similares entre si, sendo que o princípio de funcionamento é contrabalancear o movimento da estrutura com o peso de uma massa sólida ou líquida instalada no topo do edifício. Ambas soluções são ditas sintonizadas porque são dimensionadas para trabalhar com máxima eficiência na frequência natural de vibração da estrutura, o que garante máxima proteção contra a vibração ao edifício no pior caso possível. Edifícios como o Taipei 101, em Taiwan e o Citigroup Center na cidade de Nova Iorque utilizam

TMDs, enquanto edifícios como o One Wall Centre em Vancouver e o Comcast Center na Filadélfia usam TLDs.

1.2. Motivação

A atual tendência mundial de se construir prédios cada vez mais altos e esbeltos vem tornando as estruturas cada vez mais sensíveis aos efeitos de cargas dinâmicas, como sismos e ventos. No contexto brasileiro, as cargas de sismos não costumam ser significativas, mas o efeito do vento, principalmente na região Sul do país, vem causando vários danos a estruturas que não foram projetadas com esse tipo ou nível de cargas. Além disso, outros países da América Latina e do mundo têm enfrentado terremotos de grande intensidade, como no Peru em 2007, no Chile e Haiti, em 2010, ou até mesmo na Itália, em 2016.

Uma forma de proteger as estruturas contra a ação de sismos e ventos é a implantação de sistemas de controle de vibração, como o Amortecedor de Líquido Sintonizado. Essa solução foi apresentada pela primeira vez ao autor desse trabalho pelo professor Dr. Michael Tait, da McMaster University, em sua experiência de intercâmbio. Após trabalhar com o estudo experimental de um TLD, o autor teve seu interesse despertado para o tema.

Tendo em vista o fenômeno da globalização e grande facilidade de se trabalhar no exterior, torna-se importante para um engenheiro brasileiro que ele também conheça a área de dinâmica das estruturas, assim como os sistemas de controle de vibração existentes, suas vantagens e desvantagens e que ele saiba dimensionar estruturas solicitadas a cargas dinâmicas. Dentro desse contexto, o presente trabalho visa estudar o comportamento de estruturas amortecidas com TLCD, um tipo específico de TLD, para contribuir com as pesquisas relacionadas a esse tema. Busca-se ainda encontrar um dimensionamento adequado de um TLCD para amenizar o efeito da vibração na estrutura e desenvolver um *software* capaz de realizar todos os cálculos necessários para alcançar esses objetivos.

1.3. Colocação do Problema

O estudo da dinâmica das estruturas e dos sistemas de controle de vibrações são muito importantes para o dimensionamento de edifícios altos em regiões de grande incidência de sismos

e ventos fortes. Para realizar esses estudos é comum utilizar-se estruturas modeladas como *shear building*. Quanto aos sistemas de controle de vibrações, existem diversos modelos diferentes. Nesse trabalho, será estudado o Amortecedor de Coluna de Líquido Sintonizado (TLCD).

Estruturas do tipo *shear building* são pórticos em que as lajes são consideradas infinitamente rígidas e concentram toda a massa da estrutura. Nesse tipo de modelo, o único movimento possível é o deslocamento horizontal das lajes. Portanto, um *shear building* possui tantos graus de liberdade quanto forem o número de pavimentos da estrutura. Esse tipo de modelo é bastante utilizado no estudo de dinâmica, pois ele simplifica bastante a análise do problema e fornece resultados bastante próximos à realidade.

A análise de um *shear building* equipado com TLCD sob ação sísmica deve ser iniciada com a determinação dos parâmetros da estrutura, tais como as dimensões do edifício, rigidez dos pilares e massa das lajes. Essa deve ser seguida da determinação dos parâmetros do TLCD, como massa de fluido, rigidez equivalente e amortecimento equivalente. Feito isso, deve-se escolher um sismo de projeto e encontrar as equações que descrevem a força equivalente que esse sismo gera na estrutura. De posse de todas essas informações é possível determinar a equação de movimento desse sistema.

A resolução da equação de movimento pode ser feita por diversas técnicas analíticas ou numéricas. Para esse trabalho escolheu-se utilizar o Método das Diferenças Finitas como forma de resolver tal equação. Ao obter a solução, deve-se proceder com a análise dos resultados, que pode ser feita pelo fator de amplificação dinâmica (DMF). Esse é um parâmetro que relaciona o deslocamento da estrutura provocado pela carga dinâmica com o deslocamento que seria obtido caso a carga fosse estática. Pode-se analisar o DMF para diferentes frequências de excitação: maiores, menores ou iguais à frequência natural da estrutura.

A Figura 1 mostra os modelo de *shear building* que serão estudado e os parâmetros geométricos da estrutura e do TLCD. Pode-se observar que esse se trata de um sistema com dois graus de liberdade, sendo x_f o grau de liberdade do fluido e x_s o grau de liberdade da estrutura.

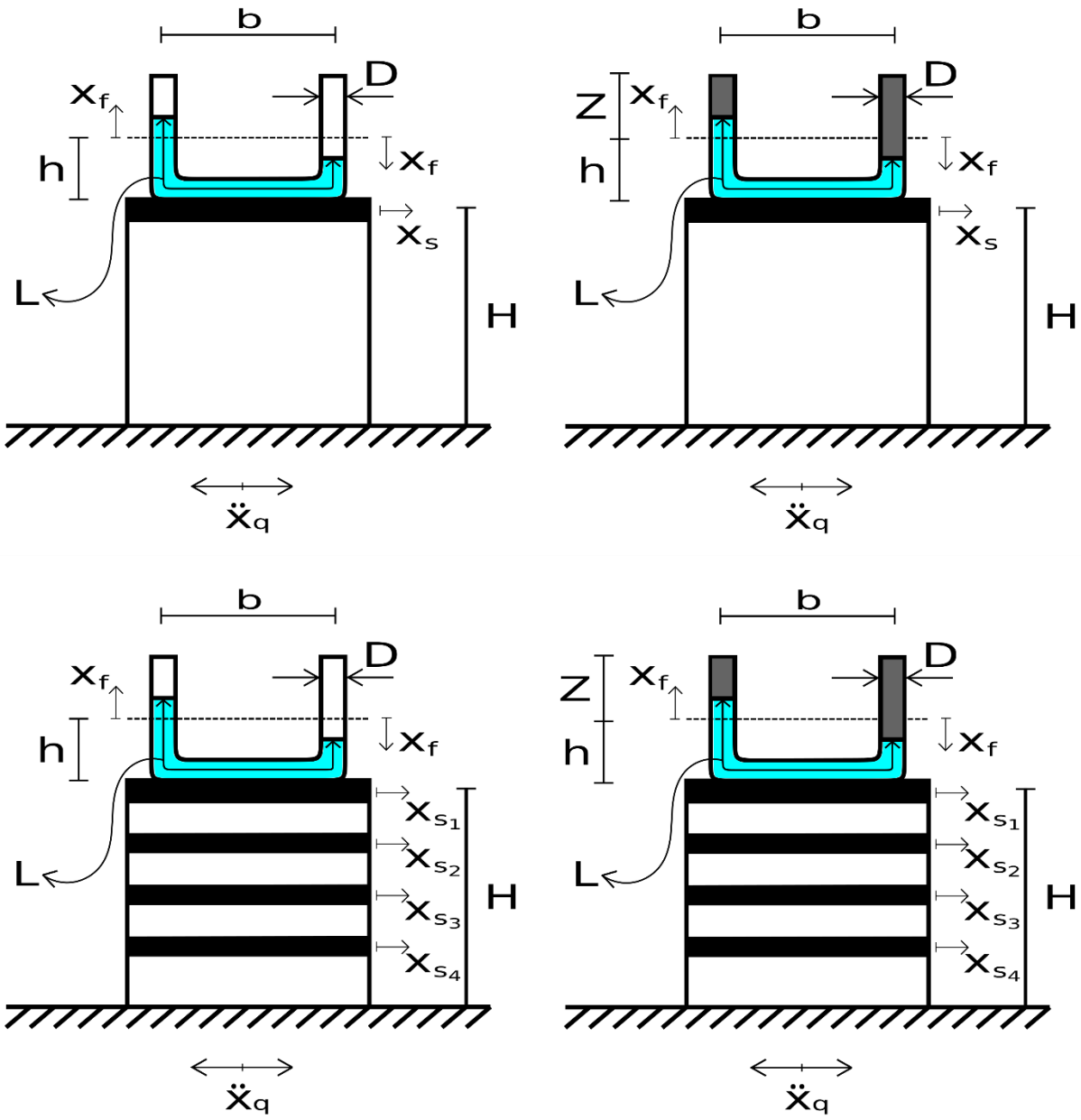

Figura 1 - Modelos de shear building equipados com TLCD simples (esquerda) e com ar comprimido (direita). Estruturas com um grau de liberdade (acima) e com vários graus de liberdade (abaixo)

1.4. Objetivos

O objetivo geral desse trabalho é analisar e entender o comportamento de edifícios equipados com TLCD através da utilização do *software* DynaPy, desenvolvido na pesquisa, assim como comparar diferentes modelos de TLCD em diferentes estruturas.

Os objetivos específicos desse trabalho incluem:

- Determinar os parâmetros da estrutura e do amortecedor;
- Determinar as equações que descrevem a força efetiva de um sismo sobre a estrutura;
- Determinar as equações de movimento que descrevem um *shear building* equipado com cada modelo de TLCD;
- Desenvolver em Python o *software* DynaPy para resolver numericamente as equações de movimento;
- Validar o programa através da comparação de resultados analíticos para problemas de um grau de liberdade e da comparação de resultados numéricos obtidos pelo SAP 2000 para problemas de vários graus de liberdade;
- Obter o fator de amplificação dinâmica do sistema para um dado conjunto de parâmetros;
- Analisar o efeito da variação paramétrica dos TLCDs sobre o fator de amplificação dinâmica;
- Encontrar um dimensionamento adequado para cada TLCD para a atenuação da vibração da estrutura;
- Comparar os resultados para os diferentes modelos de TLCD;

1.5. Metodologia

Inicialmente foi realizada uma revisão bibliográfica sobre os estudos realizados a respeito de TLDs para situar esse trabalho em relação ao que já existe e uma revisão sobre os conceitos básicos de dinâmica das estruturas. Além disso, pesquisou-se os métodos computacionais que poderiam ser empregados nesse trabalho para realizar as operações necessárias e obter-se os resultados pretendidos.

Em seguida, implementou-se uma rotina computacional em Python para o armazenamento de dados, obtenção da equação de movimento, análise numérica e pós-processamento. O método

numérico utilizado foi o Método das Diferenças Finitas. Com esse *software* foi possível realizar diversas análises de diferentes estruturas e amortecedores e obter dados comparativos para cada análise.

Os modelos de estrutura e amortecedor estudados foram modelados um a um e suas respectivas equações de movimento foram obtidas pelo programa. Para fazer essa modelagem utilizou-se os conceitos da Teoria das Estruturas e da Mecânica dos Fluidos, além dos conceitos específicos da Dinâmica das Estruturas. Os resultados obtidos na análise de cada modelo foram comparados e discutidos, verificando-se o efeito da variação paramétrica do TLD sobre a atenuação da vibração da estrutura. Por fim, buscou-se encontrar um dimensionamento ideal para os modelos estudados.

1.6. Abrangências e Limitações

Esse trabalho aborda modelos de TLCD de seção circular, podendo ser aplicado a estruturas do tipo *shear building*. Os resultados aqui encontrados podem ser utilizados como material de apoio na análise dinâmica de estruturas amortecidas por TLCD.

As limitações do trabalho decorrem da não adoção de tubos com seções não circulares. Além disso, adota-se que cada pavimento da estrutura possui um único grau de liberdade e que a massa se concentra exclusivamente nas lajes de cada pavimento.

1.7. Organização do Trabalho

Esse trabalho é dividido em seis capítulos. No capítulo 1, faz-se a introdução do trabalho, apresentando-se as motivações, colocação do problema estudado e objetivos. Esse capítulo possui ainda uma breve explicação dos principais conceitos utilizados.

No capítulo 2, aborda-se a revisão bibliográfica realizada pelo autor, que inclui uma explicação sobre como funciona o Amortecedor de Líquido Sintonizado, suas principais

características, seus diferentes tipos, vantagens e desvantagens e alguns estudos sobre TLDs realizados ao redor do mundo.

No capítulo 3, explica-se os fundamentos teóricos desse trabalho, incluindo uma breve introdução à dinâmica das estruturas, como se analisa a vibração de sistemas com um ou mais graus de liberdade e como se calcula a força equivalente de um sismo sobre a estrutura. Além disso, aborda-se o método de solução empregado e explica-se como foi feita toda a modelagem do problema, quais equações e parâmetros de análise foram utilizados e como se fez o dimensionamento do TLCD.

No capítulo 4, trata-se dos aspectos computacionais envolvidos na solução do problema, como a arquitetura do *software* DynaPy, desenvolvido pelo autor, as variáveis de entrada, o processo de solução numérica e o pós-processamento. Além disso, comenta-se a respeito da validação do programa.

No capítulo 5, apresenta-se os resultados obtidos com auxílio do *software*, assim como os efeitos da variação paramétrica do TLCD.

No capítulo 6, expõem-se as conclusões desse trabalho, assim como as perspectivas para os futuros trabalhos nessa área.

2. Revisão Bibliográfica

Nesse capítulo, explica-se o funcionamento do Amortecedor de Líquido Sintonizado, suas principais características, seus diferentes tipos, vantagens e desvantagens e alguns estudos sobre TLDs realizados ao redor do mundo.

2.1. Como Funcionam os Amortecedores de Líquido Sintonizado

Toda e qualquer estrutura possui um certo nível de amortecimento intrínseco, que se origina de suas deformações plásticas, atrito, geração de calor, som, entre outras fontes. No entanto, tal amortecimento pode ser bastante reduzido em edifícios esbeltos. Em função disso, tais estruturas necessitam de sistemas de controle de vibrações para resistir a cargas dinâmicas e minimizar amplitudes dos deslocamentos.

Sistemas de controle de vibrações podem ser de três tipos, ativos, passivos ou híbridos. Sistemas de controle ativo são aqueles que utilizam energia externa para dissipar energia da estrutura, através de um atuador hidráulico ou pneumático, por exemplo. Já os sistemas de controle passivo são aqueles que não requerem energia externa. Por fim, os sistemas de controle híbridos são aqueles que utilizam tanto um sistema ativo como um passivo (HOUSNER, CHASSIAKOS, *et al.*, 1997).

Amortecedores de Líquido Sintonizado (TLD) são um sistemas de controle passivo e têm como objetivo amortecer a vibração da estrutura. O princípio básico de funcionamento de um TLD é bastante simples e pode ser explicado de forma intuitiva. Se um dado edifício alto com um tanque de água no topo é acelerado bruscamente para a direita, por exemplo, por inércia, a água dentro do tanque será jogada para o lado oposto, contrabalanceando assim o movimento da estrutura. Além disso, o próprio movimento da água dentro de um tanque gera perdas de pressão que irão dissipar

a energia transmitida a ela e, consequentemente, a energia do sistema como um todo. Dessa forma, a instalação do TLD reduz as amplitudes de vibração da estrutura.

No entanto, tal princípio de funcionamento se trata de um caso de vibração tanto da estrutura quanto do tanque d'água. Com isso, a água e a estrutura possuirão frequências naturais de vibração, coeficientes de amortecimento, rigidezes e massas distintas. Dessa forma, é possível encontrar valores de rigidez e amortecimento equivalentes do fluido e analisar esse sistema como um sistema massa-mola-amortecedor de dois graus de liberdade, como é mostrado na Figura 2.

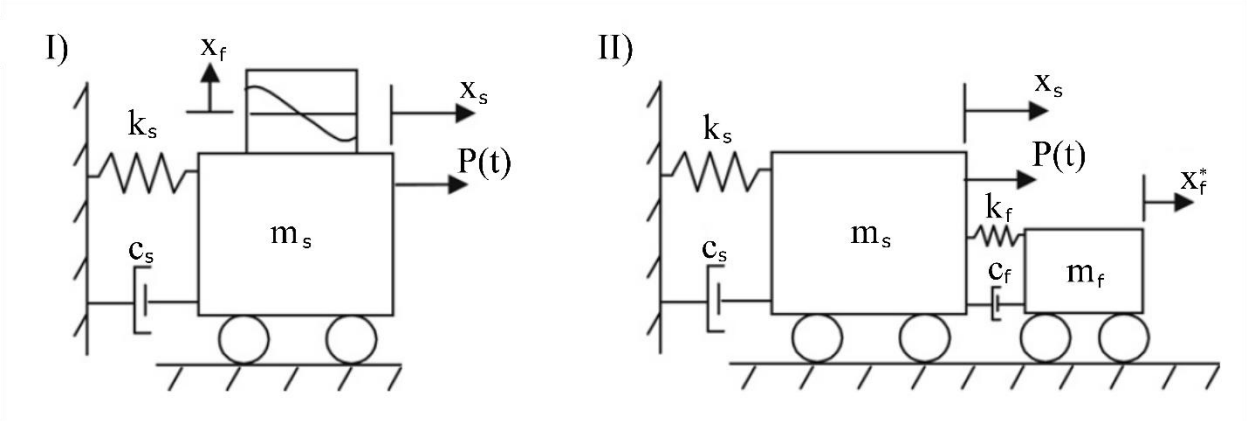

Figura 2 - (I) Modelo mecânico de uma estrutura equipada com TLCD - (TAIT, 2008) - Adaptada.

Existem dois principais tipos principais de TLDs: *Tuned Sloshing Damper* (TSD) e *Tuned Liquid Column Damper* (TLCD). O primeiro possui o formato de um tanque e funciona dissipando energia através da ação da viscosidade e do movimento de quebra das ondas. Já o segundo possui a forma de um tubo em U e a energia é dissipada pela perda de pressão devido ao contato com as paredes do tubo e pela passagem da água por um orifício.

Apesar de o princípio de funcionamento ser simples de se entender, as formulações envolvidas no estudo de um TLD, seja ele um TSD ou um TLCD, são não lineares e bastante complexas. Isso dificulta consideravelmente a pesquisa a respeito desse sistema de controle de vibração se comparado ao TMD, por exemplo. No entanto, os resultados obtidos com TLDs são

muito bons e é possível alcançar uma boa performance estrutural com um baixo custo de manutenção e uma pequena massa adicionada. Uma das primeiras estruturas a terem um TLD instalado foi uma torre no aeroporto de Nagasaki. A instalação do TLD proporcionou um aumento de aproximadamente cinco vezes na razão de amortecimento crítico da estrutura, sendo que o TLD pesava apenas 0,59% da massa da estrutura (HOUSNER, CHASSIAKOS, *et al.*, 1997).

O uso de TLDs, comparado com outras soluções, possui várias vantagens. A primeira é que o TLD não necessita de nenhum sistema de ativação, como ocorre no TMD. Isso significa que o sistema funciona por conta própria, não havendo nenhum gasto de energia e ou risco de o sistema não funcionar por falta de ativação. Uma outra vantagem é que mesmo os cálculos para o dimensionamento de um TLD sendo complexos, o sistema em si é muito simples, fácil e barato de ser instalado. Além disso, por se tratar essencialmente de um tanque de água, o TLD pode ser utilizado como reservatório para o sistema de *sprinklers* do edifício, como é feito no One Wall Centre (Vancouver, Canadá).

2.2. Tipos de TLCDs

Existem diversos modelos distintos de TLCDs, que são utilizados para diversas aplicações diferentes. Junto aos diversos modelos, surgem diversas siglas, já consagradas na literatura. Alguns desses modelos e suas principais características são (PESTANA, 2012):

- Amotecedor de Coluna de Liquido Sintonizado (TLCD) – Modelo padrão;
- Amortecedor de Líquido Absorsor de Vibrações (LCVA) – TLCD com área da seção transversal variável;
- Amortecetor de Coluna de Líquido Sintonizado Duplo (DTLCD) – Combinação de dois TLCDs em direções ortogonais;
- Amortecedor de Líquido Sintonizado Circular (CTLCD) – TLCD com formato circular (visto em planta);
- Amortecedor de Coluna de Líquido Sintonizado Pressurizado (PTLCD) – TLCD pressurizado.

O modelo padrão de TLCDs costuma ser utilizado para combater a vibração de edifícios devido à carga de vento e, em menor escala, de sismos. Já o LCVA surge mais recentemente como um aprimoramento do TLCD. Os DTLCDs, por sua vez, podem ser utilizados quando se deseja aumentar o amortecimento da estrutura nas duas direções e os CTLCDs são ideais para o controle de torção. Por fim, os PTLCDs surgem como uma alternativa inteligente de ajustar a frequência natural de vibração do fluido variando-se a pressão aplicada sobre ele, ao invés de se alterar as dimensões do reservatório.

2.3. Estudos sobre TLDs ao Redor do Mundo

O estudo de Dinâmica das Estruturas já se encontra em um estado bastante consolidado, podendo se encontrar vários textos clássicos sobre o assunto, como Blevins (2001), Chopra (1995), Clough e Penzien (2003), French (2001), Naudascher e Rockwell (1994) e Tedesco (1999). Em todos esses textos é possível encontrar com formulações para sistemas de um ou mais graus de liberdade, assim como as matrizes modais de vibração e soluções analíticas para diversos casos distintos.

Também é possível encontrar uma série de textos e pesquisas desenvolvidos pelo Grupo de Dinâmica e Fluido-Estrutura da Universidade de Brasília (GDFE). Entre eles, incluem-se Pedroso (1992, 2000, 2003, 2005), Bomtempo (2014). Nesses textos são abordados diversos temas, desde uma introdução à dinâmica das estruturas e a derivação de equações de movimentos a métodos de amortecimento específicos e análise dinâmica de plataformas *offshore*.

Da mesma forma, existe uma série de pesquisas sobre TLD publicadas e disponíveis. Cassolato (2007) estuda a instalação de *smart screens* em um TLD em forma de tanque retangular. Seu objetivo é analisar diferentes mecanismos de telas que se adaptam de forma a mudar a perda de pressão provocada pela passagem do fluido pelas telas, mantendo a característica passiva do sistema de controle de vibrações em questão.

Kenny (2013) estuda a otimização de diversos parâmetros de um TLCD, como a taxa de sintonia, coeficiente de perda de pressão, taxa de massa e taxa de comprimento. Os resultados experimentais sem amortecimento são comparados àqueles com um TLCD bem otimizado e de

um TLCD mau otimizado, mostrando a grande importância da otimização desse tipo de amortecedor.

De maneira similar, Gao *et al* (1996) estuda a otimização de TLCDs, mas também propõe a utilização de um TLCD em forma de tubo em "V", ao invés dos tradicionais tubos em "U". Em seus resultados, ele observou que esse novo modelo proposto possui a capacidade de suprimir vibrações mais intensas, o que possibilita a utilização de TLCDs em ambientes de ventos fortes.

Já Tait (2008) propõe um método de pré-dimensionamento de TLDs com tanque retangular em que se lineariza as equações de movimento e faz-se a análise do sistema como sendo um simples modelo massa-mola-amortecedor equivalente. O método proposto por esse autor foi testado experimentalmente e concluiu-se que ele é capaz de representar bem o primeiro modo de vibração do fluido, mas não permite simular o comportamento não linear do TLD, sendo impossível prever a resposta dinâmica da superfície livre do fluido por esse método.

Love e Tait (2011) elaboram uma formulação para análise do comportamento da onda do TLD para um tanque de geometria arbitrária. Tais autores validaram sua formulação pela comparação com resultados experimentais com tanques de forma retangular, circular e um de forma complexa.

Banerji e Amanta (2011) estudam um modelo de TLD hibrido, em que se usa um TLD rigidamente ligado a um TMD, que por sua vez é acoplado à estrutura por meio de uma mola. Tal sistema otimizado foi capaz de reduzir significativamente a resposta dinâmica da estrutura, se comparado ao modelo de TLD simples. Além disso, tais resultados foram obtidos com um sistema que é mais leve do que o TLD tradicional.

Baleandra *et al* (1997) estudou a eficiência de TLCDs no controle de vibração de diversas estruturas sob a ação aleatória do vento. Nesse estudo foram considerados diversos sistemas estruturais distintos, além de diversas alturas para as estruturas. Dentre as conclusões obtidas nesse estudo, os autores observaram que o posicionamento do TLCD em estruturas flexíveis é um fator de grande influência para a sua eficiência, enquanto que para *shear buildings*, essa influência é desprezível.

Shum *et al* (2007) estuda a utilização de PTLCDs para reduzir a vibração de pontes estaiadas de grandes vãos. O autor utiliza múltiplos PTLCDs sob o deck da ponte de forma que os carregamentos aplicados ao deck são transferidos ao amortecedor, que por sua vez irá reduzir as vibrações laterais e de torção. Em seus resultados, o autor demostra que esse modelo de amortecedor é bastante eficiente para essa aplicação e a escolha do PTLCD dá ao projetista uma grande flexibilidade para definir o dimensionamento e a frequência natural do amortecedor.

3. Fundamentação Teórica e Métodos de Solução

Nesse capítulo, explica-se os fundamentos teóricos desse trabalho, incluindo uma breve introdução à dinâmica das estruturas, como se analisa a vibração de sistemas com um ou mais graus de liberdade e como se calcula a força equivalente de um sismo sobre a estrutura. Além disso, aborda-se o método de solução empregado e explica-se como foi feita toda a modelagem do problema e quais equações e parâmetros de análise foram utilizados. Por fim, descreve-se o método de dimensionamento do TLCD utilizado nesse projeto.

3.1. Conceitos de Dinâmica das Estruturas

A dinâmica das estruturas difere-se da estática em muitos aspectos, como por exemplo, a presença de carregamentos de inércia, que surge da aceleração de massas. Além disso, em problemas dinâmicos, o carregamento externo e os deslocamentos da estrutura são variáveis com o tempo e a vibração pode perdurar além do momento em que o carregamento externo cessa. O próprio carregamento externo pode ser de várias naturezas, como sismos, ventos, explosões ou até mesmo da utilização da estrutura.

O estudo de dinâmica das estruturas pode ser dividido em vibrações de sistemas com um grau de liberdade (SDOF) e vibrações de sistemas com vários graus de liberdade (MDOF). O primeiro é utilizado quando o sistema em questão vibra de tal forma que seu movimento pode ser descrito por apenas uma variável dependente do tempo. Sistemas onde há uma direção principal de vibração que predomina sobre as outras também podem ser estudados como SDOF. Já os casos mais complexos de vibração, onde as várias direções de vibração são importantes, devem ser estudados como MDOF.

Tanto em sistemas SDOF como em MDOF, as estruturas podem ter vibrações livres ou forçadas. Vibrações livres são caracterizadas pela ausência de forças externas, enquanto vibrações

forçadas possuem algum tipo de excitação variável no tempo. Tais excitações podem ser determinísticas ou aleatórias, dependendo se é possível descrevê-las matematicamente por uma equação conhecida ou se é necessária uma análise estatística. As excitações determinísticas podem ser periódicas ou não periódicas. Nesse trabalho será utilizado um sismo idealizado de forma senoidal e, portanto, um caso de vibração forçada determinística periódica.

Os sistemas também podem ser classificados como amortecidos ou não amortecidos. Em sistemas amortecidos há a presença de forças dissipativas, que irão transformar a energia mecânica do sistema em outras formas de energia, reduzindo, assim, a amplitude da vibração ao longo do tempo. Sistemas não amortecidos são caracterizados pela ausência de forças dissipativas e, portanto, são considerados sistemas conservativos, onde a vibração irá perdurar indefinidamente. Naturalmente, sistemas não amortecidos são ideais e impossíveis de acontecer no mundo real.

O amortecimento de uma estrutura pode ser crítico, subcrítico ou supercrítico. O amortecimento crítico ocorre quando a razão de amortecimento crítico é tal que, ao deslocar a estrutura, essa retorna para a sua posição de equilíbrio sem oscilar em torno dela. Amortecimentos subcríticos ocorrem para taxas de amortecimento menores que essa e amortecimentos supercríticos para taxas maiores. Em geral, edifícios comerciais e industriais, torres e outras estruturas reticuladas possuem razão de amortecimento crítico muito baixas, recaindo sempre no caso subcrítico. Portanto, nesse trabalho serão considerados apenas casos de amortecimento subcrítico.

Vibrações forçadas amortecidas possuirão duas parcelas de movimento, uma parcela permanente e uma transiente. A parcela transiente corresponde à vibração do sistema caso essa fosse uma vibração livre. Portanto, em um sistema amortecido, a parcela transiente possui amplitude decrescente devido ou efeito do amortecimento, chegando a um ponto em que o equilíbrio estático é reestabelecido e a estrutura para de vibrar. Já a parcela permanente ocorre devido ao carregamento externo. Assumindo que o carregamento externo seja periódico, a parcela permanente de vibração também será periódica.

As excitações periódicas possuem uma determinada frequência que a caracteriza. Já as estruturas, também possuem uma frequência natural de vibração, que é dependente da massa e da rigidez da mesma. Quando a frequência natural de vibração da estrutura coincide com a frequência

de excitação do carregamento externo, ocorre o fenômeno de ressonância. Na ressonância, as amplitudes de vibração crescem bastante e de maneira rápida. Em sistemas não amortecidos, esse crescimento é contínuo e, matematicamente, poderia continuar infinitamente. Em sistemas reais, isso significaria um aumento das amplitudes de deslocamento até que houvesse o colapso da estrutura. Já em sistemas amortecidos, o crescimento é limitado e é possível controla-lo aumentando o amortecimento da estrutura.

O objetivo do estudo de dinâmica das estruturas é entender as possíveis formas de vibração e quais danos elas podem causar a uma estrutura. Com esse conhecimento, os engenheiros são capazes de dimensionar as estruturas e gerar soluções para os diversos desafios encontrados na concepção de edifícios esbeltos e com pequena rigidez.

3.1.1. Vibrações com um Grau de Liberdade

O primeiro passo para analisar um sistema de um grau de liberdade é identificar as forças atuantes sobre ele. Tipicamente existem quatro tipos de forças presentes: forças de inércia (F_I), forças de amortecimento (F_D), forças elásticas (F_S) e carregamentos externos (P). A Figura 3 representa sistema massa-mola-amortecedor, com uma dada massa, rigidez e amortecimento próprio sob a ação de um carregamento externo. A equação de equilíbrio desse sistema é dada por:

$$F_I(t) + F_D(t) + F_S(t) = P(t) \tag{1}$$

Considerando-se um carregamento externo de forma senoidal, pode-se reescrever a equação de equilíbrio como:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = P_0 sen(\Omega t) \tag{2}$$

Em que:

m: massa do bloco

c: coeficiente de amortecimento

k: rigidez da mola

$\ddot{x}$: aceleração do bloco

$\dot{x}$: velocidade do bloco

x: deslocamento do bloco em relação à posição de equilíbrio

P_0: magnitude do carregamento externo

Ω: frequência de excitação do carregamento externo

t: tempo

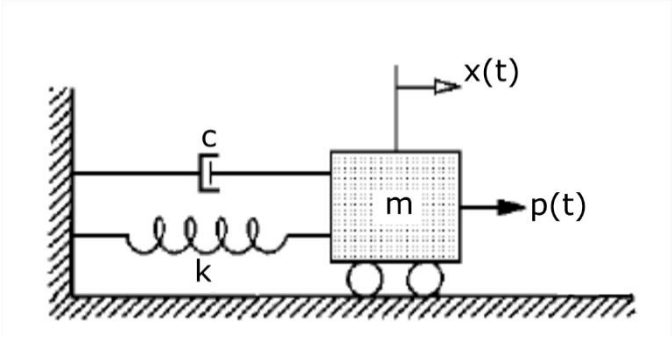

Figura 3 - Sistema massa-mola-amortecedor com um grau de liberdade (CHOPRA, 1995) - Adaptada

Pode-se observar que a equação de movimento obtida se trata de uma equação diferencial ordinária não homogênea de segunda ordem. Considerando-se o caso de amortecimento subcrítico, a solução analítica para ela é mostrada a seguir.

$$x(t) = x_h(t) + x_p(t) \tag{3}$$

$$x_h(t) = e^{-\zeta\omega t}[Asen(\omega_D t) + Bcos(\omega_D t)]$$

$$x_p(t) = \frac{P_0}{k}\left[\frac{1}{(1-r^2)^2 + (2\zeta r)^2}\right][(1-r^2)sen(\Omega t) - 2\zeta r\cos(\Omega t)]$$

Em que:

x_h: solução homogênea da equação de movimento

x_p: solução particular da equação de movimento

$\omega = \sqrt{k/m}$: frequência natural de vibração da estrutura

$\zeta = c/C_c = c/(2m\omega)$: razão de amortecimento crítico da estrutura

$\omega_D = \omega\sqrt{1-\zeta^2}$: frequência natural amortecida da estrutura

$r = \Omega/\omega$: taxa de frequência de excitação para frequência natural de vibração

A e B: constantes obtidas a partir das condições de contorno

3.1.2. Vibrações com Vários Graus de Liberdade

Sistemas com vários graus de liberdade são ainda mais complexos de serem analisados, pois não só são necessárias mais variáveis para se definir a equação de equilíbrio, como também os graus de liberdade possuem um certo nível de interdependência, em que o movimento de um influencia no outro. Dessa forma, a equação de movimento deixa de ser uma EDO de segunda ordem para se tornar um sistema linear de EDOs de segunda ordem. São definidas, portanto, matrizes de massa, amortecimento e rigidez, assim como vetores de aceleração, velocidade, deslocamento e carregamentos. Esse sistema linear pode ser representado de forma matricial, como é mostrado seguir:

$$[m]\{\ddot{x}(t)\} + [c]\{\dot{x}(t)\} + [k]\{x(t)\} = \{P(t)\} \tag{4}$$

Tomando como modelo de estudo um sistema com dois conjuntos massa-mola-amortecedor em série, representado na Figura 4, a Segunda Lei de Newton e a Teoria das Estruturas, tem-se que a equação de movimento matricial assume a seguinte forma:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1(t) \\ \ddot{x}_2(t) \end{Bmatrix} + \begin{bmatrix} c_1 & -c_1 \\ -c_1 & c_1 + c_2 \end{bmatrix} \begin{Bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 + k_2 \end{bmatrix} \begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix} = \begin{Bmatrix} P_1(t) \\ P_2(t) \end{Bmatrix} \tag{5}$$

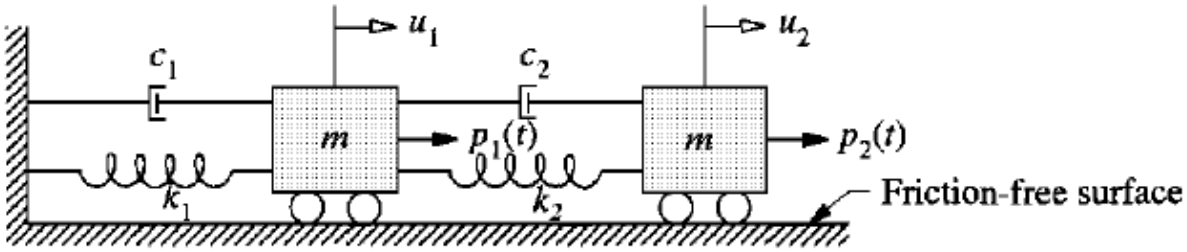

Figura 4 - Sistema massa-mola-amortecedor de dois graus de liberdade (CHOPRA, 1995) - Adaptada

Nota-se da equação obtida, que o próprio arranjo físico dos conjuntos massa-mola-amortecedor gera uma interdependência entre os graus de liberdade. Por exemplo, um

deslocamento imposto à massa 1 irá gerar forças elásticas tanto na próprio massa 1 como na massa 2.

A solução analítica desse problema já se torna bastante complexa se comparada ao sistema massa-mola-amortecedor com apenas um grau de liberdade. Para a análises de sistemas com ainda mais graus de liberdade, torna-se inviável resolver esse problema analiticamente. Portanto, deve-se recorrer a métodos numéricos, como o Método das Diferenças Finitas, que é adotado nesse trabalho.

3.1.3.Solicitações Sísmicas

As solicitações sísmicas são bastante recorrentes em certos territórios e podem representar cargas extremamente elevadas sobre a estrutura. Sua origem geralmente é tectônica, mas também pode ser induzida pela movimentação de grandes volumes de água em barragens, entre outros motivos. A natureza de excitações sísmicas é aleatória, não sendo possível definir uma equação matemática que a descreva.

A ocorrência de um sismo gera uma movimentação da fundação da estrutura e não uma força aplicada a ela diretamente. Dessa forma, é necessário calcular-se uma força equivalente *(P_{eq})* que o sismo provoca na estrutura. Isso pode ser feito utilizando-se a Segunda Lei de Newton, como demonstrado a seguir:

$$P_{eq} = m_s \ddot{x_q} \tag{6}$$

Assumindo-se um modelo de aceleração senoidal do sismo, pode-se obter a seguinte equação:

$$\ddot{x}_q(t) = -\Omega^2 A sen(\Omega t) \tag{7}$$

Substituindo a aceleração senoidal idealizada na equação 6, tem-se:

$$P_{eq}(t) = -m_s \Omega^2 A sen(\Omega t) \tag{8}$$

3.2. Método das Diferenças Finitas

O Método das Diferenças Finitas consiste de um procedimento numérico para a obtenção de soluções aproximadas de equações diferenciais. Existem três abordagens para esse método, as diferenças progressiva, diferenças regressiva e diferenças centradas sendo que o último é o que garante uma melhor aproximação. O método das diferenças centradas aproxima as derivadas de uma função pela inclinação da reta secante que passa um ponto afrente e um ponto atrás do ponto de interesse. Dessa forma, a derivada no ponto passa a ser uma função da coordenada dos pontos vizinhos e da distância entre eles. Esse resultado é representado pela equação a seguir:

$$\dot{x}_i = \frac{x_{i+1} - x_{i-1}}{2\Delta t} \tag{9}$$

É possível também encontrar a aproximação para a segunda derivada, como mostrado a seguir:

$$\ddot{x}_i = \frac{x_{i+1} - 2x_i + x_{i-1}}{\Delta t^2} \tag{10}$$

Dada a equação de movimento abaixo:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = P(t) \tag{11}$$

É possível resolver tal equação substituindo-se as soluções aproximadas para a primeira e segunda derivada de *x*. Dessa forma, obtém-se

$$m\left(\frac{x_{i+1} - 2x_i + x_{i-1}}{\Delta t^2}\right) + c\left(\frac{x_{i+1} - x_{i-1}}{2\Delta t}\right) + k(x_i) = P_i \tag{12}$$

Isso resulta em na solução da equação para um ponto no tempo. Portanto, divide-se o intervalo de tempo de estudo em vários pontos discretos com um passo Δt e aplica-se a solução acima para cada ponto. Dessa forma, obtém-se um sistema linear como mostrada abaixo:

$$\begin{cases} \left(\frac{m}{\Delta t^2}-\frac{c}{2\Delta t}\right)x_{-1}+\left(k-\frac{2m}{\Delta t^2}\right)x_0+\left(\frac{m}{\Delta t^2}+\frac{c}{2\Delta t}\right)x_1=P_0 \\ \left(\frac{m}{\Delta t^2}-\frac{c}{2\Delta t}\right)x_0+\left(k-\frac{2m}{\Delta t^2}\right)x_1+\left(\frac{m}{\Delta t^2}+\frac{c}{2\Delta t}\right)x_2=P_1 \\ \vdots \\ \left(\frac{m}{\Delta t^2}-\frac{c}{2\Delta t}\right)x_{n-2}+\left(k-\frac{2m}{\Delta t^2}\right)x_{n-1}+\left(\frac{m}{\Delta t^2}+\frac{c}{2\Delta t}\right)x_n=P_{n-1} \end{cases} \quad (13)$$

O sistema acima possui alguns valores que não são conhecidos inicialmente e não serão obtidos pelo método. Tais pontos devem ser obtidos pelas condições de contorno do problema. A solução do sistema linear pode ser facilmente obtida computacionalmente, mas há ainda um jeito mais fácil e que exige menos processamento para resolver a EDO. A equação 12 pode ser reorganizada da seguinte forma:

$$x_{i+1}=\left(\frac{m}{\Delta t^2}+\frac{c}{2\Delta t}\right)^{-1}\left(P_i-\left(k-\frac{2m}{\Delta t^2}\right)x_i-\left(\frac{m}{\Delta t^2}-\frac{c}{2\Delta t}\right)x_{i-1}\right) \quad (14)$$

Começando de i = 0, basta obter os valores de x_0 e x_{-1} através das condições de contorno para que a equação possa ser utilizada para encontrar x_1. A partir daí todos os demais valores de x podem ser obtidos de forma iterativa utilizando a equação acima.

Para sistemas com dois graus de liberdade, a massa *m*, amortecimento *c* e rigidez *k* passam a ser representados por matrizes de massa [m], amortecimento [c] e rigidez [k]. A solução iterativa dos deslocamentos pode ser obtida da mesma forma e o resultado é análogo ao encontrado anteriormente (TEDESCO, MCDOUGAL e ROSS, 1999).

$$\{x_{i+1}\}=\left[\frac{[m]}{\Delta t^2}+\frac{[c]}{2\Delta t}\right]^{-1}\left\{\{P_i\}-\left[[k]-\frac{2[m]}{\Delta t^2}\right]\{x_i\}-\left[\frac{[m]}{\Delta t^2}-\frac{[c]}{2\Delta t}\right]\{x_{i-1}\}\right\} \quad (15)$$

3.3. Modelagem da Estrutura com Um Grau de Liberdade

O primeiro passo na modelagem do problema é calcular as propriedades da estruturas ou seja, sua massa, rigidez dos pilares e amortecimento da estrutura. A massa é adotada como um valor fixo e considera-se que ela está toda concentrada na laje do pórtico. Já a rigidez deve ser calculada utilizando-se a Teoria das Estruturas. O modelo adotado consiste de um pórtico simples engastado nas duas bases e, portanto, a rigidez de cada pilar é dada por:

$$k_{pilar} = \frac{12EI}{H^3} \tag{16}$$

Em se tratando de dois pilares paralelos suportando a mesma viga, a rigidez da estrutura como um todo é dada pela soma da rigidez dos dois pilares, ou seja:

$$k_s = \frac{24EI}{H^3} \tag{17}$$

Quanto ao amortecimento, estudos indicam que a razão de amortecimento crítico de estruturas esbeltas costuma ser bastante baixa. A própria norma NBR 6123/88 recomenda que seja adotado uma razão de amortecimento crítico de 2% para estruturas com estrutura aporticada de concreto, sem cortinas. Esse valor foi adotado para todos os casos estudados nesse trabalho.

3.4. Modelagem do TLCD

Na modelagem do TLCD, calcula-se os mesmos parâmetros que foram calculados para a estrutura. A massa do fluido é dada pelo volume de água dentro do tubo multiplicada pela massa específica (ρ_f) da água.

$$m_f = \frac{\pi D^2}{4} L \rho_f \tag{18}$$

Já o amortecimento do fluido é uma função da perda de pressão do líquido ao se mover dentro do tubo. A força de amortecimento F_d atuante em uma seção do fluido pode ser definida em função da pressão no fluido P_f e da área A:

$$F_d = P_f A \tag{19}$$

Já a pressão no fluido pode ser definida pelo produto da perda de carga Δh pelo peso específico do fluido γ_f.

$$\begin{aligned} P_f &= \Delta h \gamma_f \\ P_f &= \left(f \frac{L}{D} \frac{\dot{x_f}^2}{2g} \right) (\rho_f g) \end{aligned} \tag{20}$$

Para calcular o coeficiente de atrito f utilizou-se a equação de Sousa-Cunha-Marques, mostrada abaixo:

$$\frac{1}{\sqrt{f}} = -2 \log_{10} \left(\left(\frac{\varepsilon}{3{,}7D} \right)^{1{,}11} - \frac{5{,}16}{Re} \log_{10} (\frac{\varepsilon}{3{,}7D} + \frac{5{,}09}{Re^{0{,}87}} \right) \tag{21}$$

Em que ε é a rugosidade do tubo e Re é o número de Reynolds, que pode ser calculado pela seguinte expressão, que depende da viscosidade cinemática do fluido v_f:

$$Re = \frac{\dot{x_f} D}{v_f} \tag{22}$$

Substituindo o valor de P_f e de A na equação 19, obtem-se a seguinte expressão:

$$\begin{aligned} F_d &= \left(f \frac{L}{D} \frac{\dot{x_f}^2}{2g} \right) (\rho_f g) \left(\frac{\pi D^2}{4} \right) \\ F_d &= \left(\frac{\pi L D \rho_f}{8} \right) (f |\dot{x_f}|) \dot{x_f} \end{aligned} \tag{23}$$

Essa pode ser reescrita em função de c_k, uma constante, e $c_{\dot{x}}$, um fator de correção dependente da velocidade do fluido:

$$c_k = \frac{\pi L D \rho_f}{8} \tag{24}$$

$$c_{\dot{x}} = f\left|\dot{x}_f\right| \tag{25}$$

$$F_d = c_k c_{\dot{x}} \dot{x}_f \tag{26}$$

Portanto, o coeficiente de amortecimensto do fluido c_f, é calculado para cada velocidade pelo seguinte produto:

$$c_f = c_k c_{\dot{x}} \tag{27}$$

Uma forma de aumentar o coeficiente de amortecimento do fluido é inserir um diafragma pelo qual o fluido deve passar. Tipicamente, insere-se esse mecanismo no meio do trecho horizontal do TLCD. A perda de carga localizada causada pelo diafragma é função de sua área A_c e pode ser calculada com a seguinte expressão:

$$\Delta h_d = \frac{\dot{x}_f^{\,2}}{2g}\left(\frac{A}{A_c} - 1\right)^2 \tag{28}$$

Portanto, a força de amortecimento gerada por esse diagragma é dada por:

$$F_d = \frac{\dot{x}_f^{\,2}}{2g}\left(\frac{A}{A_c} - 1\right)^2 \gamma A \tag{29}$$

Reescrevendo essa expressão, pode-se definir a constante c_{dk} e a variável $c_{d\dot{x}}$ de forma que:

$$c_{dk} = \frac{\rho A}{2}\left(\frac{A}{A_c} - 1\right)^2 \tag{30}$$

$$c_{d\dot{x}} = \left|\dot{x}_f\right| \tag{31}$$

$$c_d = c_{dk} c_{d\dot{x}} \tag{32}$$

$$F_d = c_d \dot{x}_f \tag{33}$$

Portanto, pode-se somar o coeficiente de amortecimento do fluido devido à perda de carga distribuída c_f com o coeficiente de amorteciemento do fluido devido à perda de carga localizada c_d para se obter uma expressão mais geral.

$$c_f = c_k c_{\dot{x}} + c_{dk} c_{d\dot{x}} \tag{34}$$

Por fim, calcula-se a rigidez equivalente do fluido k_f. Uma vez que o fluido é considerado incompressível, não existe uma rigidez real associada a ela. No entanto, existe uma frequência natural de vibração do líquido dentro do tubo. Para calcular essa frequência, assume-se uma vibração livre do fluido dentro do tubo sem forças dissipativas. Nessa situação, quando o deslocamento do fluido é máximo, tem-se a situação demonstrada na Figura 5.

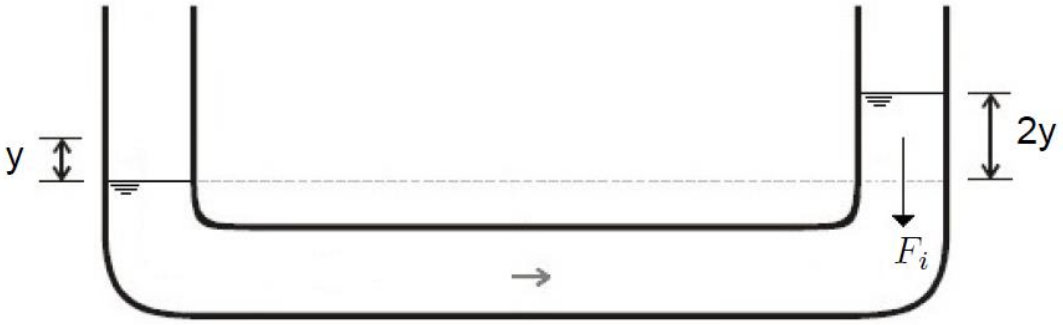

Figura 5 - Tubo em U em vibração livre não amortecida (PESTANA, 2012)

Utilizou-se o método apresentado em Pestana (2012) para se calcular a frequência natural de vibração do fluido. Fazendo o equilíbrio de forças na situação da Figura 5, tem-se o seguinte:

$$-F_i = m_f \ddot{y}$$

$$\rho A_f L \ddot{y} + 2\rho_f A g y = 0$$

$$\rho_f AL\left(-\omega^2 Y sen(\omega_f t)\right) + 2\rho_f Ag\left(Y sen(\omega_f t)\right) = 0$$

$$\left(-\rho_f AL\omega_f^2 + 2\rho_f Ag\right)\left(Y sen(\omega_f t)\right) = 0$$

$$\left(-\rho_f AL\omega_f^2 + 2\rho_f Ag\right) = 0$$

$$\omega_f^2 = \frac{2\rho_f Ag}{\rho_f AL}$$

$$\omega_f = \sqrt{\frac{2g}{L}} \tag{35}$$

A partir dessa frequência natural é possível calcular uma rigidez equivalente do fluido da seguinte forma:

$$\omega_f = \sqrt{\frac{k_f}{m_f}}$$

$$\sqrt{\frac{2g}{L}} = \sqrt{\frac{k_f}{m_f}}$$

$$k_f = \frac{\pi D^2 \rho g}{2} \tag{36}$$

Adicionando-se câmaras de ar comprimido às extremidades livres do TLCD, como é feito nos modelos de PTLCD, pode-se ainda somar a rigidez do fluido à rigidez do gás (PEDROSO, 1992):

$$k_{ar} = 1{,}4\frac{P}{Z}\frac{\pi D^2}{4} * 2 \tag{37}$$

Em que *P* é a pressão de ar dentro da câmara e *Z* é a altura da coluna de ar. Dessa forma, a expressão completa de k_f pode ser escrita como:

$$k_f = \frac{\pi D^2 \rho g}{2} + \ 1{,}4\frac{P}{Z}\frac{\pi D^2}{2} \tag{38}$$

Portanto, a equação de movimento do fluido se torna:

$$m_f \ddot{x} + c_f \dot{x} + k_f x = F(t)$$

$$\left(\frac{\pi D^2}{4} L \rho_f\right)\ddot{x} + (c_k c_{\dot{x}})\dot{x} + \left(\frac{\pi D^2 \rho g}{2} + 1{,}4\frac{P}{Z}\frac{\pi D^2}{4}\right)x = F(t) \tag{39}$$

3.5. Equação de Movimento do Sistema

Uma vez encontradas as massas, rigidezes e amortecimentos do fluido e da estrutura, é possível montar a equação de movimento do sistema. Essa equação irá se assemelhar com a equação 5. No entanto, as matrizes de massa, rigidez e amortecimento terão formas diferentes para esse caso. No sistema estrutura-TLD, o deslocamento e a velocidade em um grau de liberdade não gera forças no outro grau de liberdade. Portanto, as matrizes de rigidez e amortecimento se tornam matrizes diagonais. No entanto, a aceleração da estrutura para a direita, por exemplo, move o fluido para o lado esquerdo do tubo. Esse movimento gera uma força de inércia no fluido. Portanto, a matriz de massa não é diagonal, uma vez que surgem forças no grau de liberdade do fluido quando a estrutura é acelerada e vice-versa. Desse modo, chega-se na seguinte equação de movimento, que é válida para ambos os modelos de TLCD:

$$\begin{bmatrix} m_s + m_f & \frac{b}{L}m_f \\ \frac{b}{L}m_f & m_f \end{bmatrix} \begin{Bmatrix} \ddot{x}_s(t) \\ \ddot{x}_f(t) \end{Bmatrix} + \begin{bmatrix} c_s & 0 \\ 0 & c_f \end{bmatrix} \begin{Bmatrix} \dot{x}_s(t) \\ \dot{x}_f(t) \end{Bmatrix} + \begin{bmatrix} k_s & 0 \\ 0 & k_f \end{bmatrix} \begin{Bmatrix} x_s(t) \\ x_f(t) \end{Bmatrix} = \begin{Bmatrix} P_{eq}(t) \\ 0 \end{Bmatrix} \quad (40)$$

3.5.1. Equação de Movimento para Estruturas Complexas

A equação 40 descreve o movimento de um *shear building* de um pavimento com um TLCD montado sobre ele. Para casos mais complexos, pode-se generalizar essa equação de forma que cada elemento das matrizes e dos vetores sejam representados por submatrizes ou subvetores, como mostrado a seguir:

$$\begin{bmatrix} [M_s] & [O^*] \\ [O^*]^T & [M_f] \end{bmatrix} \begin{Bmatrix} \{\ddot{X}_s(t)\} \\ \{\ddot{X}_f(t)\} \end{Bmatrix} + \begin{bmatrix} [C_s] & [0] \\ [0] & [C_f] \end{bmatrix} \begin{Bmatrix} \{\dot{X}_s(t)\} \\ \{\dot{X}_f(t)\} \end{Bmatrix} + \begin{bmatrix} [K_s] & [0] \\ [0] & [K_f] \end{bmatrix} \begin{Bmatrix} \{X_s(t)\} \\ \{X_f(t)\} \end{Bmatrix} = \begin{Bmatrix} \{\bar{P}_{eq}(t)\} \\ \{0\} \end{Bmatrix} \quad (41)$$

Nessa forma de representação, as submatrizes e subvetores $[0]$ e $\{0\}$ possuem todos os seus elementos nulos, enquanto as demais submatrizes e subvetores serão definidos de acordo com o caso analisado.

Para casos com estruturas de múltiplos pavimentos, deve-se definir as submatrizes genéricas de massa, amortecimento e rigidez da estrutura como mostradas abaixo. As demais submatrizes permanecem idênticas ao caso da equação 40.

$$[M_s]=\begin{bmatrix} m_{s1} & 0 & \cdots & 0 \\ 0 & m_{s2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & m_{sn}+\sum m_{fi} \end{bmatrix} \tag{42}$$

$$[C_s]=\begin{bmatrix} c_{s1} & 0 & \dots & 0 \\ 0 & c_{s2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & c_{sn} \end{bmatrix} \tag{43}$$

$$[K_s]=\begin{bmatrix} k_{s1}+k_{s2} & -k_{s2} & 0 & \dots & 0 \\ -k_{s2} & k_{s2}+k_{s3} & -k_{s3} & \ddots & 0 \\ 0 & -k_{s3} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & k_{sn-1}+k_{sn} & -k_{sn} \\ 0 & 0 & 0 & -k_{sn} & k_{sn} \end{bmatrix} \tag{44}$$

Já para casos em que se deseja instalar múltiplos TLCDs idênticos sobre o último pavimento, deve-se definir as submatrizes genéricas de massa, amortecimento e rigidez do fluido, além da submatriz de acoplamento de massas $[O^*]$ como mostrado a seguir.

$$[M_f]=\begin{bmatrix} m_{f1} & 0 & \cdots & 0 \\ 0 & m_{f2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & m_{f3} \end{bmatrix} \tag{45}$$

$$[C_f]=\begin{bmatrix} c_{f1} & 0 & \dots & 0 \\ 0 & c_{f2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & c_{f3} \end{bmatrix} \tag{46}$$

$$[K_f]=\begin{bmatrix} k_{f1} & 0 & \dots & 0 \\ 0 & k_{f2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & k_{f3} \end{bmatrix} \tag{47}$$

$$[0^*]=\begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \dots & 0 \\ \frac{b}{L}m_{f1} & \frac{b}{L}m_{f2} & \dots & \frac{b}{L}m_{fn} \end{bmatrix} \tag{48}$$

3.6. Fator de Amplificação Dinâmica

De posse da equação do movimento sistema, utiliza-se a rotina computacional desenvolvida pelo autor para se obter o deslocamento da estrutura ao longo do tempo, também conhecido como resposta dinâmica da estrutura. A Figura 6 mostra um exemplo de reposta obtida pelo *software* adotando-se uma altura de coluna d'água de 1,00 m, largura do tanque de 6,00 m e diâmetro de 0,90 m.

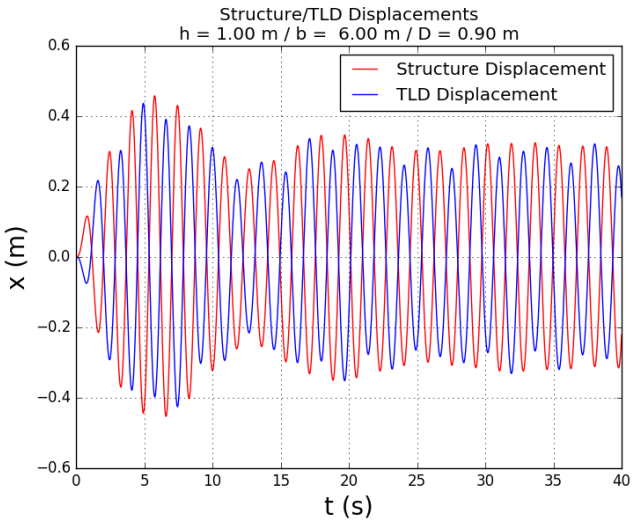

Figura 6 - Exemplo de resposta dinâmica gerada pelo software

Esse procedimento pode ser executado para diferentes parâmetros dos amortecedores, mas os resultados obtidos em cada uma das iterações não são adequados para a comparação. Portanto, é necessário utilizar um novo parâmetro que facilite a interpretação dos resultados de resposta dinâmica da estrutura. O Fator de Amplificação Dinâmica *(DMF)* é um parâmetro que relaciona a resposta dinâmica da estrutura com sua resposta estática, ou seja, o deslocamento da estrutura caso a carga aplicada sobre ela não fosse dinâmica.

$$DMF = \frac{X_d}{X_e} = \frac{X_d}{P_0/k_s} \tag{49}$$

Em que:

X_d: máximo deslocamento da estrutura sob carga dinâmica

X_e: máximo deslocamento da estrutura sob carga estática

3.7. Relação de Frequências

Como já citado anteriormente, para uma mesma intensidade de carregamento, as estruturas apresentam diferentes respostas dinâmicas à medida que se varia a frequência de excitação, sendo a resposta mais crítica a da ressonância. Portanto, é de interesse desse trabalho analisar os deslocamentos gerados por um sismo de diferentes frequências. Para facilitar a interpretação dos resultados, utiliza-se a relação de frequências *r*, que nada mais é do que a razão entre a frequência de excitação *(Ω)* e a frequência natural de vibração da estrutura *(ω)*.

$$r = \frac{\Omega}{\omega} \tag{50}$$

Dessa forma, na ressonância, o valor de *r* é igual à unidade. Para excitações abaixo da frequência de ressonância o valor de *r* é menor que um e para excitações acima da frequência de ressonância o valor de *r* é maior que um.

3.8. Análise da Eficiência do Amortecimento

Com esses parâmetros definidos, é possível comparar a performance de um dado amortecedor plotando-se um gráfico do fator de amplificação dinâmica da estrutura em função da relação de frequências. Para o caso do TLCD, pode-se comparar as curvas de *DMF* em função de *r* para amortecedores com diferentes diâmetros, comprimentos ou alturas de lâmina d'água. Pode-se também comparar os casos com e sem amortecedor para se ter uma noção do benefício que a instalação do amortecedor traz à estrutura. A Figura 7 mostra uma comparação entre o *DMF* em função de *r* da estrutura sem amortecedor e da estrutura com um TLCD simples de 1,00 m de lâmina d'água, 6,00 m de comprimento e 0,90 m de diâmetro.

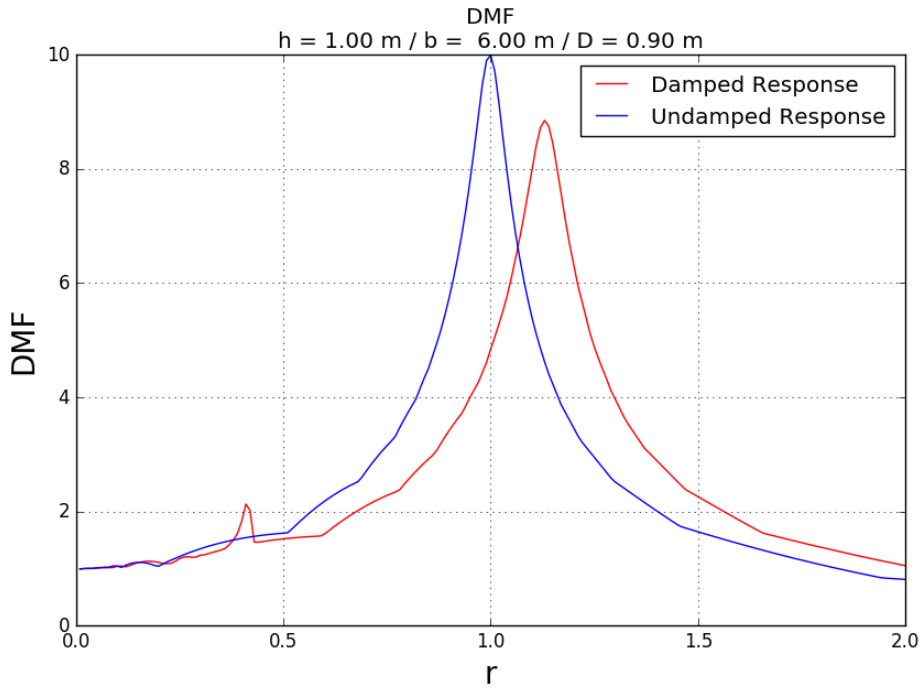

Figura 7 - Comparação entre a resposta dinâmica de uma estrutura sob ação de um sismo com e sem amortecedor

3.9. Processo de Dimensionamento do TLCD

O processo de dimensionamento do TLCD tem por objetivo sintonizar o amortecedor à frequência natural da estrutura para que ele funcione com maior eficiência na ressonância. Além disso, deve-se garantir que as dimensões do TLCD serão suficientes para gerar o amortecimento desejado e, ao mesmo tempo, não sejam muito grandes a ponto de o amortecedor gerar uma carga muito elevada sobre a estrutura.

Como enunciado, primeiramente deve-se encontrar a frequência natural de vibração da estrutura. Isso foi feito com o auxílio do DynaPy, plotando-se os gráficos de fator de amplificação dinâmica em função da frequência de excitação. O primeiro e maior pico do gráfico representa a ressonância do primeiro modo de vibração e os demais picos se localizam nas ressonâncias dos outros modos de vibração.

Com a frequência natural da estrutura encontrada, deve-se ajustar a frequência do amortecedor a esse valor. Para isso, basta fixar o comprimento *L* do amortecedor, como descrito implicitamente pela equação 35 e explicitamente pela equação 51.

$$L = \frac{2g}{\omega_f^2} \tag{51}$$

Em seguida, define-se a largura (*b*) e altura da lamina d'água (*h*) do amortecedor. Como a razão *b/L* é o fator que acopla a massa do fluido à massa do pavimento onde o TLCD foi instalado, quanto maior a largura do amortecedor, melhor será o amortecimento. Deve-se, no entanto, garantir que a altura da lamina d'água seja suficiente para que haja a oscilação vertical do fluido, ou seja, seu deslocamento máximo não pode passar da altura inicial de lamina d'água. Da literatura, observou-se que uma relação de uma largura de 70% do comprimento total do TLCD costuma ser adequada para a maioria dos casos (PESTANA, 2012).

O próximo passo é definir o diâmetro do TLCD de forma a alcançar a massa de fluido desejada. Como será mostrado no capítulo 5, a massa do amortecedor, em relação à massa da estrutura, é um dos fatores determinantes para a eficiência do TLCD. Caso o diâmetro calculado para se obter a massa desejada seja incompatível com as outras dimensões, pode-se também utilizar múltiplos TLCDs com um diâmetro menor.

Por fim, caso os deslocamentos máximos do fluido superem a altura de lamina d'água, deve-se ajustar a área do diafragma para reduzir a vibração do fluido para níveis aceitáveis. A determinação da área ideal do diafragma deve ser feita por tentativa e erro, minimizando os deslocamentos da estrutura, mas respeitando o deslocamento máximo do fluido.

Feito o dimensionamento, deve-se verificar se a massa adicional do TLCD mudou a frequência natural de vibração da estrutura. Em caso afirmativo, deve-se ajustar o comprimento do amortecedor para sintonizá-lo novamente.

Para o dimensionamento de PTLCDs, pode-se utilizar o mesmo procedimento para se alcançar uma frequência natural de vibração do fluido tão próxima quanto possível da frequência natural de vibração da estrutura. Em seguida, adiciona-se a pressão de ar que for necessária para

sintonizar o PTLCD. Essa técnica pode ser usada também para evitar a necessidade de se utilizar múltiplos amortecedores.

3.10. Abordagem das Não-Linearidades da Equação de Movimento

Como visto nas seções anteriores, o termo de amortecimento do TLCD é dependente da velocidade instantânea do fluido e isso faz com que a equação de movimento se torne não linear. Dessa forma, não é possível resolver a equação diretamente, sendo necessária a utilização de um processo iterativo ou de uma forma de aproximação do termo de velocidade. Nesse trabalho escolheu-se aproximar a velocidade instantânea do fluido pela sua velocidade dois passos de tempo atrás. Como a discretização do tempo utilizanda é muito grande, ou seja, o passo de tempo é muito pequeno, pode-se afirmar que essa aproximação é válida e tende a convergir para o valor exato à medida que se aumenta a discretização. Mais detalhes desse procedimento e o código utilizado nessa implementação podem ser encontrados no apêndice M.

4. Aspectos Computacionais

Nesse capítulo, trata-se dos aspectos computacionais envolvidos na solução do problema, como a linguagem de programação utilizada, a arquitetura do *software* desenvolvido pelo autor, as variáveis de entrada, o processo de solução numérica e o pós-processamento. Trata-se também do processo de validação do *software*.

4.1. Python

Python é uma linguagem de programação interpretada, grátis, *open source* e largamente difundida. Ela foi criada por Guido van Rossum com o intuito de ser uma linguagem minimalista, fácil de ser lida e aprendida. Python possui uma comunidade numerosa e ativa que já produziu uma grande quantidade de bibliotecas e pacotes para todo tipo de uso. Dentre eles, destacam-se o Numpy e o Matplotlib. O primeiro é uma biblioteca numérica que contém todo tipo de função para manipulação de equações, sistemas lineares, matrizes, vetores, polinômios, entre outros. O segundo se trata de uma biblioteca gráfica de plotagem 2D e 3D. Utilizando ambas bibliotecas é possível fazer análises numéricas de todo tipo e realizar o pós-processamento com facilidade, tornando-se assim, bibliotecas essenciais para pesquisas científicas.

A adoção do Python como a linguagem de programação escolhida para esse projeto baseia-se na facilidade de ler, entender e modificar os códigos escritos pelo autor por parte de outras pessoas que tenham acesso a eles. Isso garante que futuros projetos nessa área possam se basear nesse, possibilitando o avanço mais rápido do conhecimento acadêmico em dinâmica das estruturas e amortecedores de líquido sintonizado.

4.2. Arquitetura do Programa

O programa criado pelo autor possui uma arquitetura de 3 etapas: pré-processamento, processamento e pós-processamento. No pré-processamento é feita a entrada de dados por meio de uma interface gráfica, onde o usuário introduz os parâmetros geométricos do TLCD e da

estrutura, assim como a massa da estrutura e seu módulo de elasticidade. São fornecidos também as condições de contorno do problema, o tempo de análise, passo e características do sismo.

No início da etapa de processamento o programa calcula todas as outras propriedades importantes para a resolução do problema e monta as matrizes de massa, amortecimento e rigidez do sistema. São montados também os vetores de força, deslocamento, velocidade e aceleração, todos dependentes do tempo. Monta-se ainda o vetor de iteração que será usado no processamento. Com as matrizes e vetores montados, o programa utiliza o Método das Diferenças Finitas e a biblioteca Numpy para resolver as equações de movimento do problema e obter a resposta dinâmica da estrutura. Ainda utilizando a teoria de diferenças finitas, calcula-se os vetores de velocidade e aceleração ao longo do tempo, tomando como partida o vetor de deslocamento obtido.

Por fim, faz-se o pós-processamento com auxílio da biblioteca Matplotlib. Nessa etapa são gerados os gráficos que serão analisados para averiguar a qualidade do amortecedor. O programa fornece diversas opções de plotagens para visualizar os resultados, como gráficos de deslocamento em função do tempo, velocidade em função, do tempo, aceleração em função do tempo, deslocamento em função da velocidade e outros.

Para obter apenas a resposta dinâmica da estrutura para uma única configuração de amortecedor basta realizar essa rotina de 3 etapas. No entanto, caso se deseje fazer uma análise da resposta dinâmica da estrutura em função da frequência de excitação, deve-se executar a mesma rotina de 3 etapas, porém em loop, onde a cada iteração um dos parâmetros do TLCD é alterado. Nesse caso, o pós processamento é feito apenas ao final do último loop. A Figura 8 mostra um diagrama esquemático que representa esse processo.

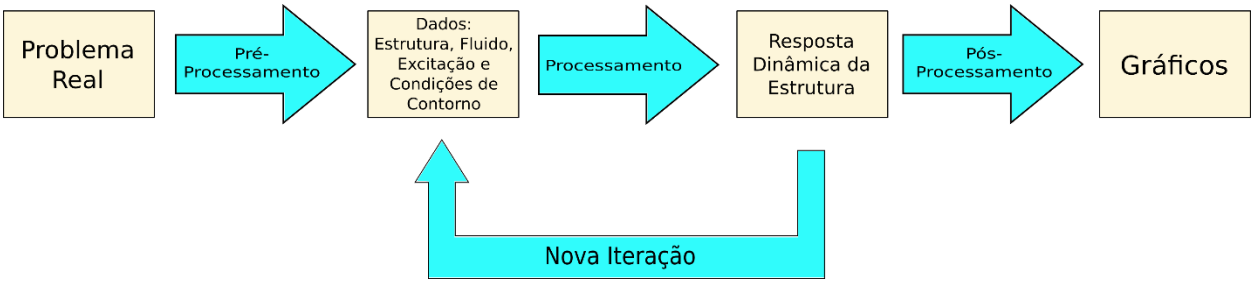

Figura 8 - Diagrama de funcionamento do software DynaPy

Durante a primeira fase desse projeto, desenvolveu-se um protótipo do programa que já era capaz de realizar todas as etapas descritas acima, mas a entrada de dados e a montagem das matrizes era manual e deveria ser feita no próprio código, fazendo com que sua utilização fosse complicada. O código desse protótipo pode ser encontrado pode ser encontrado nos apêndices A e B.

Na segunda fase do projeto, desenvolveu-se uma interface gráfica para o programa e automatizou-se o processo de montagem das matrizes. Isso permitiu não só uma utilização muito mais fácil do programa como também a generalização do procedimento, significando que o programa pode ser rodado para uma estrutura com quantos pavimentos se desejar, sem ser preciso alterar o código. Além disso, foi possível criar um sistema de salvamento de arquivos para que se possa carregar um arquivo com todas as informações digitadas em uma sessão anterior e continuar trabalhando nele. Foram criadas também sub-rotinas para desenhar a estrutura e o amortecedor, o que facilita a visualização dos dados de entrada, e de geração de arquivos de acelerogramas, utilizados na entrada de dados de excitação. Os principais códigos do programa estão disponíveis nos apêndices C a O. Esses e os demais códigos implementados no programa estão disponíveis no repositório do *website* github sob o seguinte endereço: www.github.com/MarioRaul/DynaPy.

4.3. Pré-processamento

Ao iniciar o programa, o usuário deve entrar com os dados do problema através da interface gráfica. Primeiramente, entra-se com os dados da estrutura. Para cada pavimento, o usuário pode adicionar a massa e altura do pavimento, dimensões dos pilares, módulo de elasticidade e tipo de apoio. Ao confirmar a adição do pavimento, o programa gera um desenho com todas as informações já adicionadas para a estrutura, como mostrado na Figura 9.

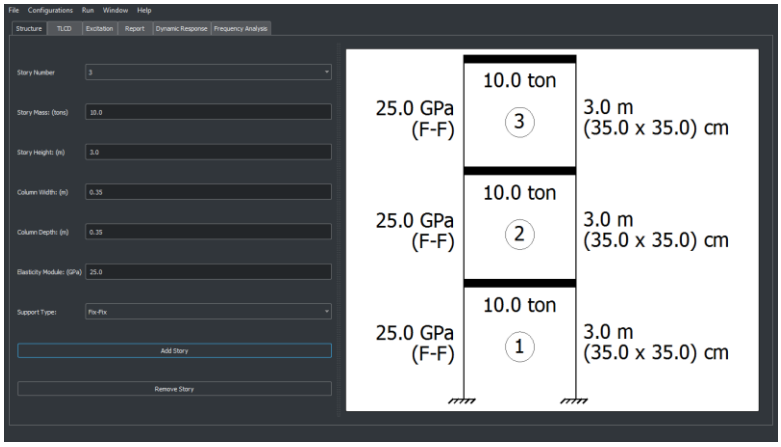

Figura 9 –Aba Structure do software DynaPy

Em seguida, o usuário deve entrar com os dados do amortecedor. No caso do amortecedor simples, são necessários o diâmetro, nível d'água e largura do tubo. Novamente, ao confirmar a adição do TLCD, o programa gera um desenho com as informações do amortecedor escolhido, como mostra a Figura 10.

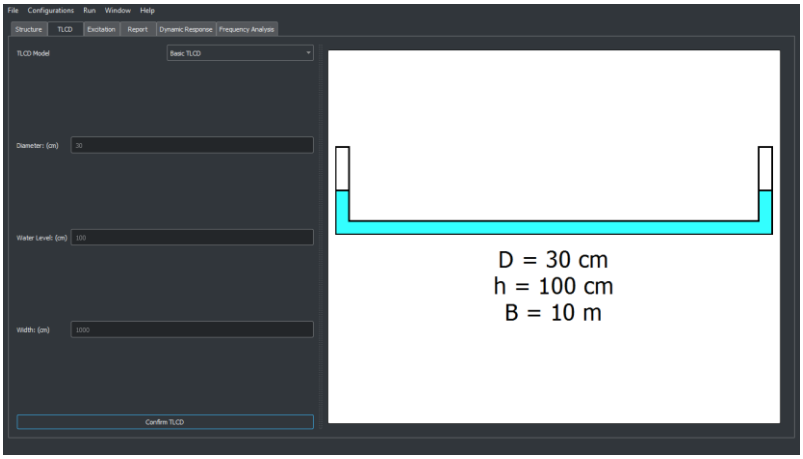

Figura 10 - Aba TLCD do software DynaPy

Por fim, deve-se adicionar o tipo de carregamento que irá excitar a estrutura. São fornecidas duas opções para isso, utilizar uma onda senoidal, onde se configura a amplitude e frequência da onda, assim como a duração da excitação e da análise, ou ler um arquivo de texto que contém um carregamento qualquer. É possível ainda criar o arquivo de texto com o carregamento no próprio programa. A Figura 11, mostra todas as opções de carregamento ser utilizados no programa.

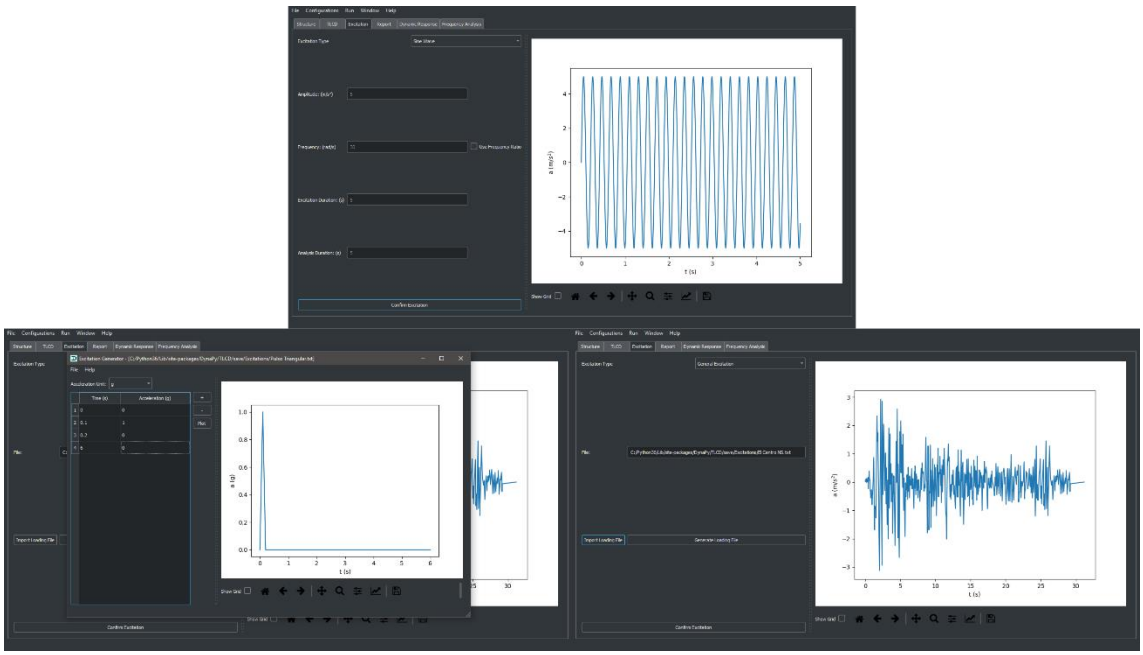

Figura 11 - Aba Excitation do software DynaPy. Aplicação de onda senoidal (acima). Gerador de carregamentos (a esqueda). Terremoto El Centro carreagado ao programa (a direita).

O menu de configurações possui ainda outros ajustes, como tamanho do passo de iteração, condições de contorno, parâmetros do fluido e taxa de amortecimento da estrutura.

4.4. Processamento

Na etapa de processamento o *software* calcula a rigidez, amortecimento e frequência natural do fluido e da estrutura assim como a massa do fluido. Com base nesses parâmetros, o programa monta as matrizes de massa, amortecimento e rigidez. Todas as matrizes são geradas automaticamente, de acordo com os valores entrados e o número de graus de liberdade do problema. Em seguida, o programa monta o vetor tempo, que contém todos os intervalos de tempo subdivididos de acordo com o passo determinado. Após isso, o *software* monta o vetor de forças externas. Na verdade, esse se trata de uma matriz com número de colunas igual ao número de intervalos para o qual a análise iterativa foi subdividida contendo em cada linha o carregamento para o respectivo grau de liberdade ao longo do tempo. Nesse trabalho, utilizou-se apenas carregamentos senoidais, podendo ser alterada sua amplitude e frequência. É possível também definir uma duração do sismo diferente da duração de análise.

4.5. Solução Numérica

Como já mencionado, a solução numérica é feita pelo Método das Diferenças Finitas. Primeiramente são definidas matrizes constantes que aparecem diversas vezes durante o cálculo, para facilitar a escrita das equações. Em seguida, encontra-se a aceleração inicial e o valor do deslocamento para um instante imediatamente anterior ao tempo zero. Com esses dados, é possível realizar a primeira iteração do método usando as equações do capítulo 3. Com a primeira iteração calculada é possível utilizar o método para encontrar os valores de deslocamento para todos os outros intervalos de tempo. Todos esses cálculos são realizados utilizando-se o NumPy e suas funções para trabalhar com matrizes e vetores.

Após calcular o deslocamento em todos os pontos, utiliza-se o conceito de primeira e segunda derivada por diferenças finitas para se calcular a velocidade e aceleração a partir dos deslocamentos. Com essa etapa concluída já se tem todos os dados necessários para análise dos resultados. O programa gera, então, um relatório geral mostrando os dados de entrada e as matrizes geradas, como mostrado na Figura 12. No entanto, a visualização dos resultados na forma de vetores e matrizes de dimensões muito grandes é extremamente complicada. Por esse motivo, realiza-se o pós processamento.

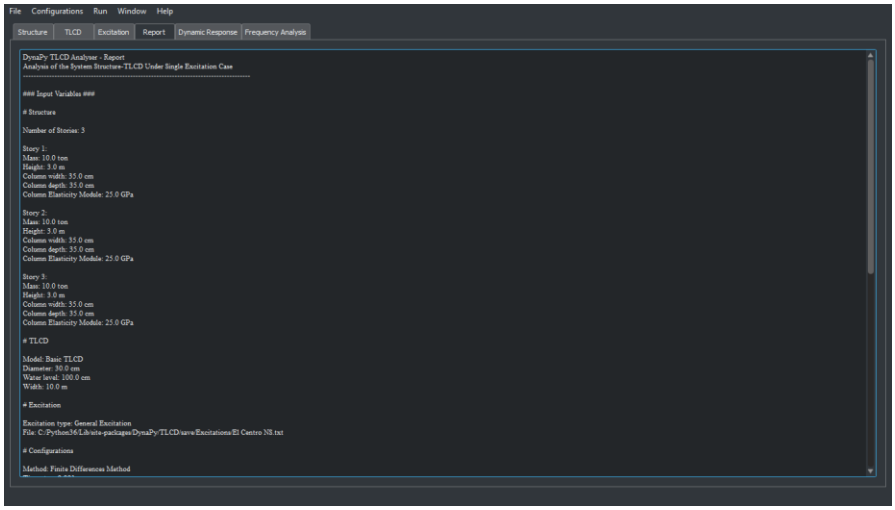

Figura 12 - Aba Report do software DynaPy

4.6. Pós-Processamento com Matplotlib

A etapa de pós-processamento é feita com auxílio da biblioteca Matplotlib. Essa biblioteca permite fazer a plotagem 2D de curvas a partir de um par de vetores, representado os eixos das ordenadas e das abscissas. É possível configurar a plotagem para definir legendas, títulos, cores de curvas, entre outros. O programa permite fazer a plotagem tanto do deslocamento como também da velocidade e da aceleração ao longo do tempo. É possível também plotar outros tipos de gráficos, como o de velocidade em função do deslocamento. Todas essas opções são mostradas na Figura 13.

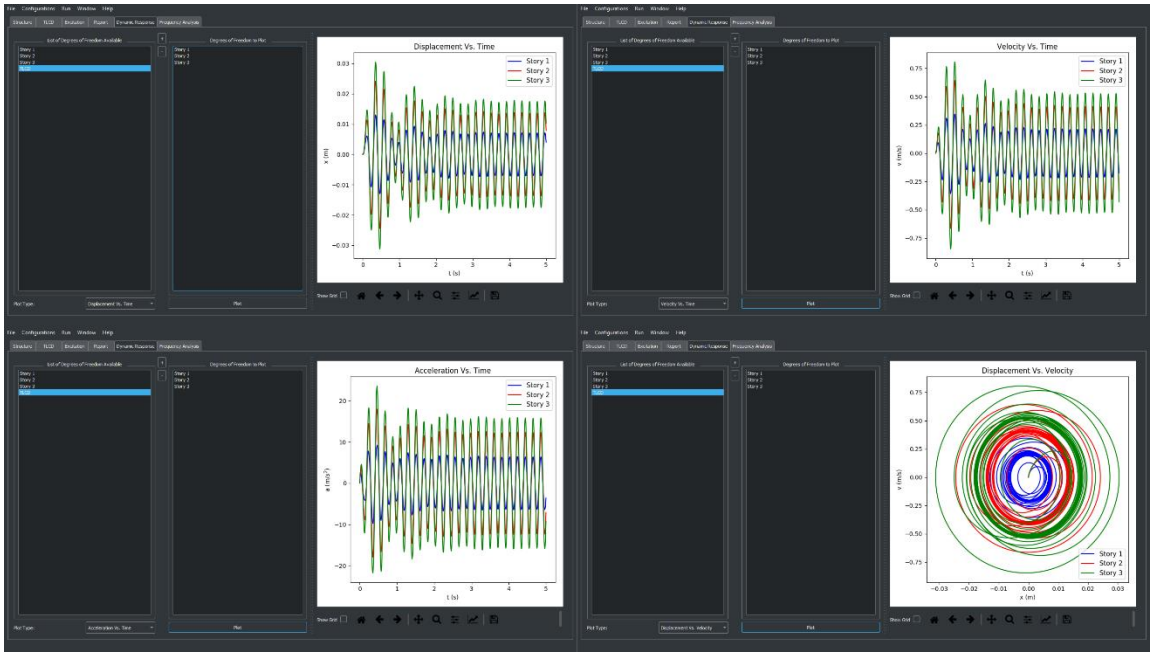

Figura 13 - Aba Dynamic Response do software DynaPy. Plot de Deslocamento Vs. Tempo (superior esquerdo), Velocidade Vs. Tempo (superior direito), Aceleração Vs. Tempo (inferior esquerdo) e Velocidade Vs. Deslocamento (inferior direito)

4.7. Análise do Fator de Amplificação Dinâmica

Nos casos em que o usuário deseja obter as curvas de DMF em função da relação de frequências, as etapas do processamento serão repetidas de forma iterativa em que a cada iteração o programa muda automaticamente o valor de um parâmetro do TLCD, realiza o processamento e armazena o vetor deslocamento para aquela iteração. Ao término do loop o programa calcula os valores de DMF e realiza o pós processamento com o Matplotlib. Esse procedimento foi implantado no protótipo do programa e os resultados obtidos com ele serviram de base para análises mais complexas, realizados com o *software* em seu formato final. Em sua forma final, é possível realizar plotagens do DMF em função da frequência de excitação assim como gráficos do máximo deslocamento em função da frequência. Ambos podem ser visualizados na Figura 14.

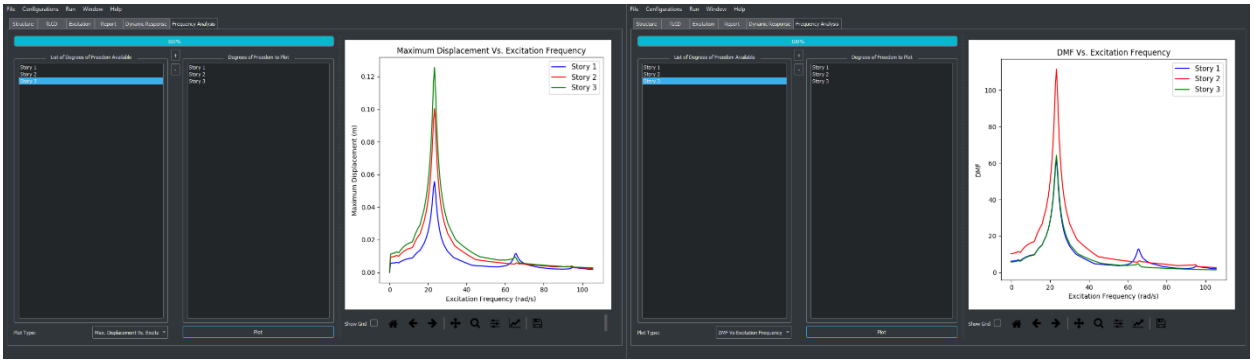

Figura 14 - Aba Frequency Analysis do software DynaPy. Plots de Máximo Deslocamento Vs. Frequência de Excitação (esquerda) e DMF Vs. Frequência de Excitação (direita)

4.8. Validação do Programa

Para comprovar o bom funcionamento do programa e garantir a qualidade dos resultados, foi feito um processo de validação, primeiro comparando-se o resultado do *software* para exemplos simples com a solução analítica e depois comparando resultados de casos mais complexos com os resultados obtidos pelo SAP 2000.

Primeiramente validou-se o programa para casos de um grau de liberdade. Para isso, utilizou-se a solução da equação de movimento representada na equação 3. Modelou-se no programa, portanto, uma estrutura de apenas um pavimento, sem TLCD e sob a ação de um carregamento senoidal. Comparou-se, então, os resultados obtidos pelo programa com a resposta analítica para diferentes frequências de excitação e diferentes valores de taxa de amortecimento da estrutura. Observou-se que em todos os casos a solução numérica era idêntica à analítica. A Figura 15 mostra quatro dessas comparações.

Em seguida, fez-se a validação do programa para casos com vários graus de liberdade. Nessa etapa, a principal ferramenta de validação foi o *software* SAP 2000. Novamente, testou-se diferentes configurações de amortecimento e frequência de excitação e constatou-se que a solução obtida pelo DynaPy era sempre idêntica à obtida pelo SAP 2000. Para o caso específico da Figura 16, calculou-se também a solução analítica por meio da técnica de superposição modal. Como pode ser visto, todas as soluções tiveram respostas compatíveis.

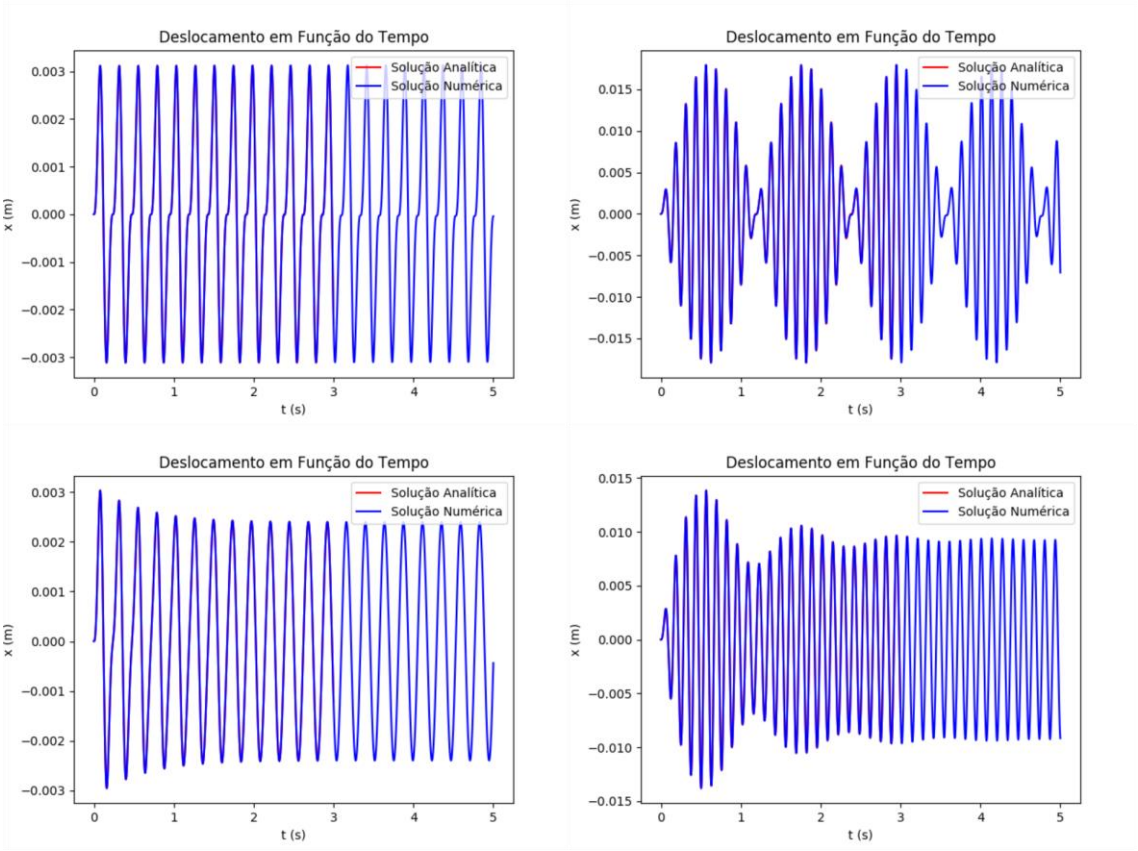

Figura 15 - Validação de difrentes casos de um grau de liberdade.

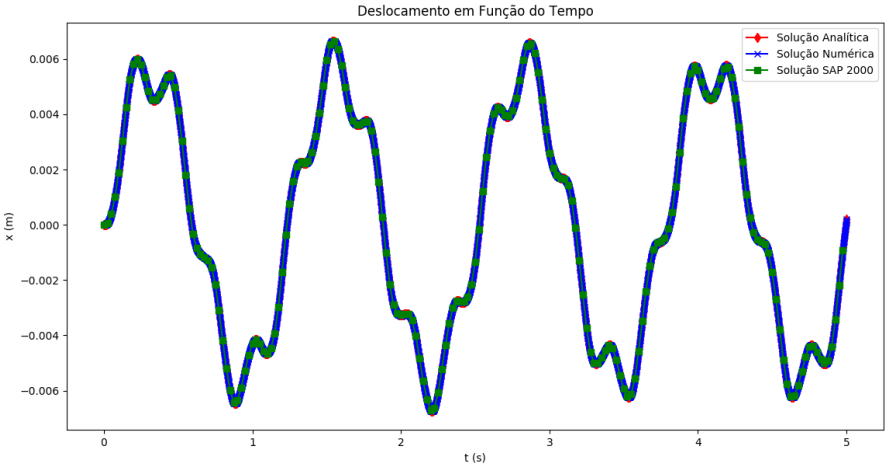

Figura 16 - Validação de caso com 3 graus de liberdade

5. Resultados

Nesse capítulo, apresenta-se os resultados obtidos com auxílio do *software*, tais como os efeitos da variação paramétrica para os modelos estudados e a resposta dinâmica das estruturas equipadas com diversos modelos de TLCD.

5.1. Variação Paramétrica do Primeiro Modelo de TLCD

O primeiro modelo de TLCD analisado foi o de um TLCD simples, sem a câmara de ar. A estrutura se trata de um *shear building* em concreto armado com massa *(m_s)* de 10 toneladas concentrada na laje rígida. Os pilares possuem seção transversal quadrada de 25 cm x 25 cm e altura *(H)* de 10 m. O módulo de elasticidade da estrutura *(E)*, que é de concreto, é de 25 GPa. O momento de inércia *(I)* do pilar pode ser calculado pela seguinte equação:

$$I = \frac{0{,}25 * 0{,}25^3}{12} = 3{,}255 * 10^{-3}\ m^4 \tag{52}$$

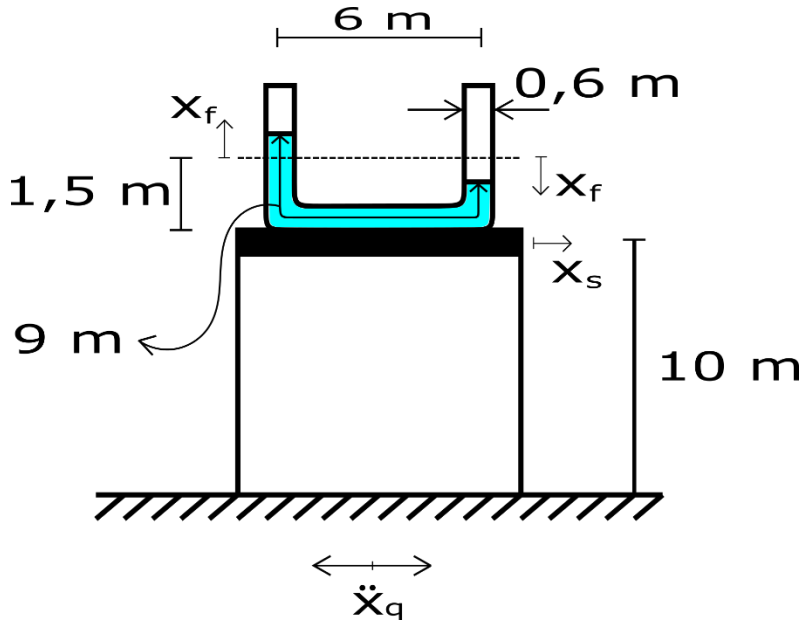

Figura 17 - Primeiro modelo de TLD estudado com estrutura de um pavimento.

O fluido utilizado no amortecedor é a água. Adotou-se que a temperatura ambiente é de 20°C e, portanto, a massa específica da água (ρ_f) adotada foi de 998,2071 kg/m³ e a viscosidade cinemática (υ_f) foi de 1,003 x 10^{-6} m²/s. A aceleração da gravidade (g) adotada foi de 9,807 m/s².

Utilizou-se a abordagem iterativa do programa em fase de potótipo para se estudar o efeito da variação paramétrica do primeiro do TLCD no comportamento das curvas DMF. Primeiramente foi estudado o efeito da altura de lâmina d'água. Fixou-se a largura do tanque em 6 m e o diâmetro em 0,60 m e variou-se a altura de lâmina d'água de 0,5 m a 8,5 m. O resultado observado foi que a variação desse parâmetro não influencia o amortecimento da estrutura de forma significativa.

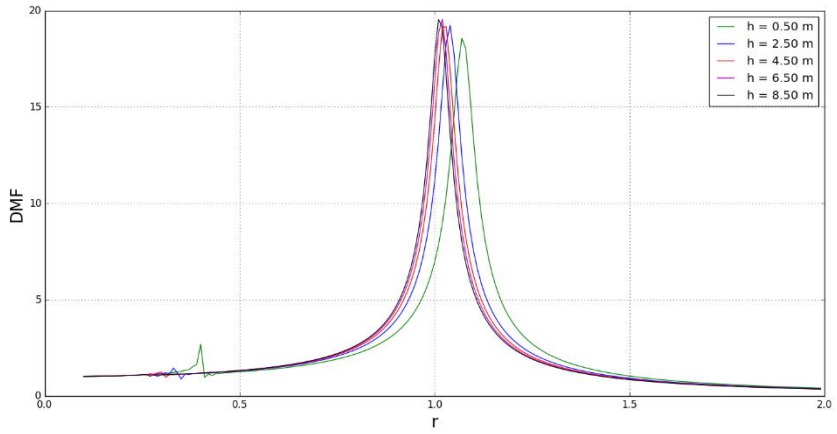

Figura 18 - TLCD Simples - Efeito da variação de altura de lâmina d'água no fator de amplificação dinâmica

O próximo parâmetro a ser estudado foi a largura do tanque. Fixou-se a altura de lâmina d'água em 1,5 m e o diâmetro em 0,60 m, variando a largura de 2 m a 42 m observou-se um aumento significativo no amortecimento da estrutura. Observa-se também um grande deslocamento da frequência natural de vibração da estrutura. Isso se deve ao aumento de massa de fluido à medida que se aumenta a largura do tanque.

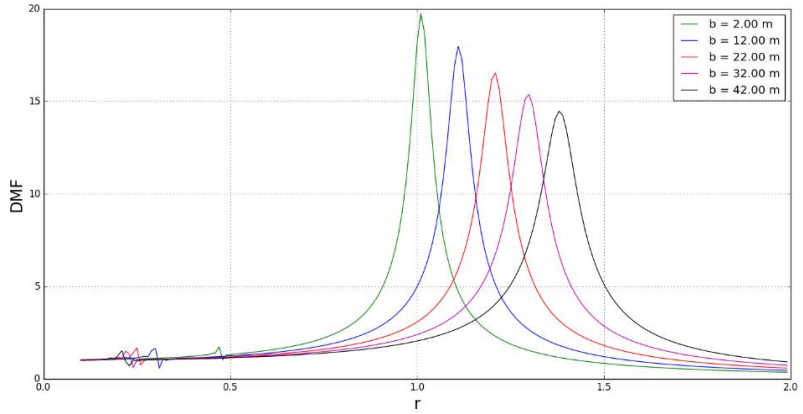

Figura 19 - TLCD Simples - Efeito da variação da largura do tanque no fator de amplificação dinâmica

Por fim, variou-se o diâmetro do tanque de 0,10 m a 2,10 m fixando a altura de lamina d'água em 1,5 m e a largura do tanque em 6 m. Novamente se observou uma grande influência no amortecimento da estrutura e uma significativa mudança na frequência natural de vibração da estrutura. No entanto, o aumento do diâmetro do tanque também levou ao aparecimento de um novo pico local no gráfico de DMF. Esse pico se encontra na frequência natural de vibração do fluido. Nesse caso, o fluido entra em ressonância e induz a estrutura a vibrar com amplitudes muito maiores do que se esperaria caso não houvesse ressonância do fluido.

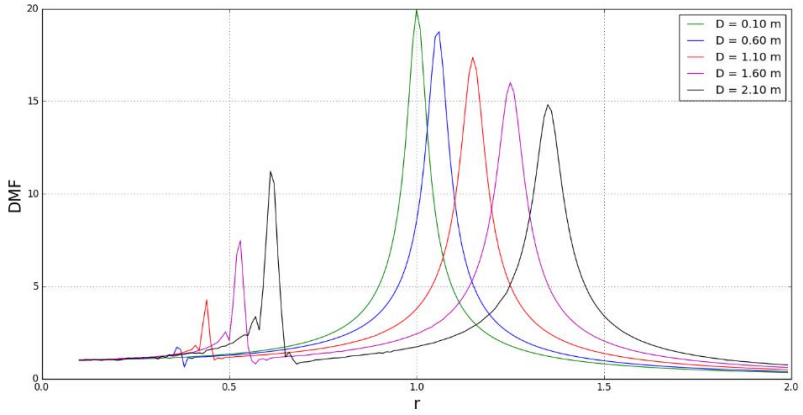

Figura 20 - TLCD Simples - Efeito da variação do diâmetro do tanque no fator de amplificação dinâmica

Esses resultados mostram que os principais fatores que devem ser observados no dimensionamento dos TLCDs são a massa da estrutura, que pode ser controlado principalmente pelo diâmetro do tubo, e a largura do tanque, que influencia na relação *b/L*, comentada na seção 3.9.

5.2. Análise Dinâmica de um Shear Building de 1 Pavimento

A primeira estrutura analisada foi um *shear building* em concreto armado com massa de 10 tonelada, dois pilares de dimensão 20 cm x 20 cm, altura de 10 m e módulo de elasticidade 25 GPa. Foram utilizadas duas estratégias para aumentar o amortecimento da estrutura na frequência de ressonância, equipá-la com múltiplos TLCDs simples e equipá-la com um único TLCD pressurizado com diferentes massas. A Figura 21 mostra a representação gráfica dessa estrutura.

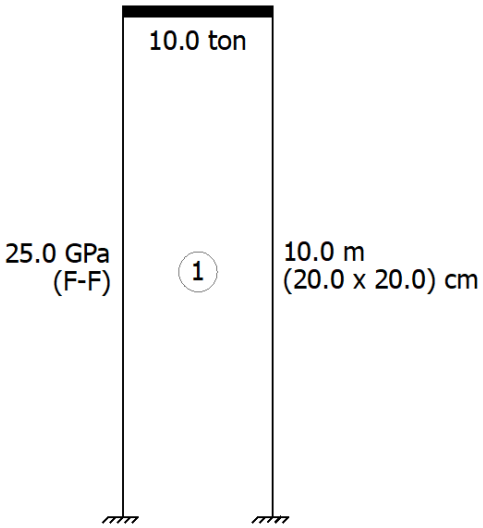

Figura 21 – Shear Building com 1 grau de liberdade

5.2.1.Utilizando Múltiplos TLCDs

A primeira estratégia implementada foi a de se utilizar múltiplos TLCDs simples. Em todos os casos estudados utilizou-se TLCDs idênticos. O dimensionamento foi feito como descrito na seção 3.9. A Tabela 1 mostra o dimensionamento de todas as opções de TLCD analisadas nesse caso. A Figura 22 mostra a resposta dinâmica da em ressonância para cada caso descrito na Tabela 1.

Tabela 1- TLCDs utilizados para análise de shear building de 1 pavimento

Diametro (cm)	Nível d'Água (cm)	Largura (cm)	Quantidade	Frequência Natural (rad/s)	Porcentagem de Massa
-	-	-	0	-	0,00 %
15	40	187	1	2,71	0,47 %
15	40	187	5	2,71	2,35 %
15	40	187	10	2,71	4,70 %

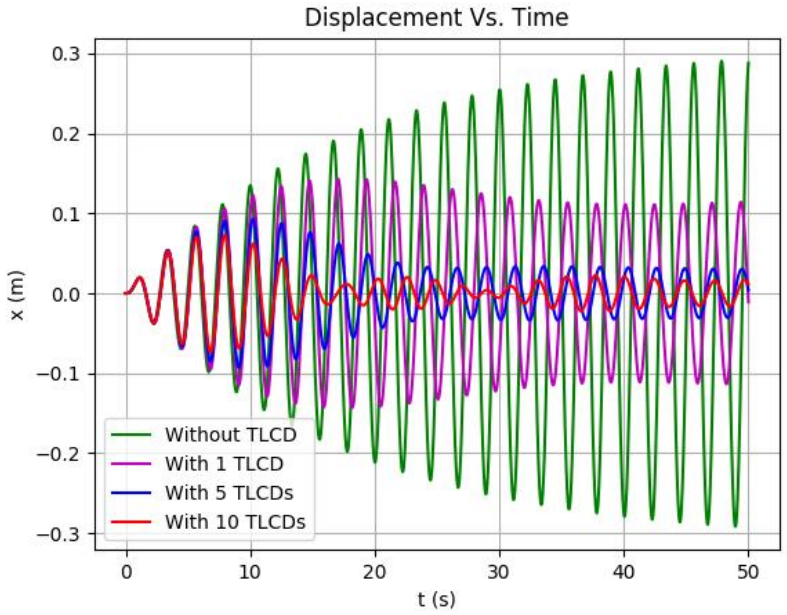

Figura 22 – Comparação entre as respostas dinâmicas de um shear building de 1 pavimento equipado com 0, 1, 5 e 10 TLCDs sem diafragma

Como esperado, o aumento do número de TLCDs e, consequentemente, o aumento da massa do amortecedor em relação à massa da estrutura, diminuiu significativamente a amplitude de vibração. Para o caso com 5 TLCDs, em que a massa do TLCD representa aproximadamente 2,35% da massa da estrutura, observou-se uma diminuição do deslocamento máximo de aproximadamente 66%. No entanto, ao se analisar o deslocamento do fluido, percebe-se que esse passa do seu valor limite, que para esses casos era de 40 cm. A Figura 23 mostra que para o caso com 5 TLCDs o deslocamento máximo do fluido passa de 75 cm, ou seja, os resultados obtidos acima são inválidos, uma vez que as hipóteses básicas de cálculo não foram respeitadas.

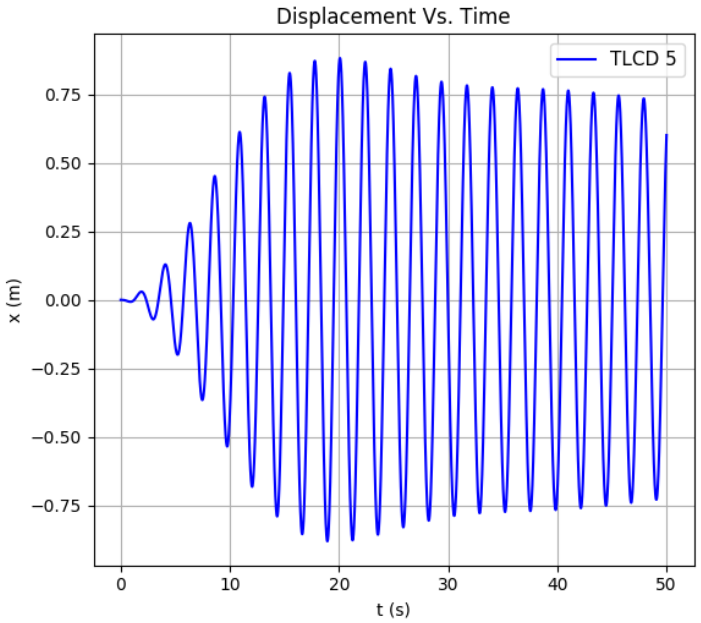

Figura 23 – Resposta dinâmica do fluido para um shear building de 1 pavimento com 5 TLCDs sem diafragma

Para solucionar esse problema, adicionou-se um diafragma aos TLCDs estudados. Para cada caso utilizou-se uma área de diafragma diferente, de forma que o deslocamento do fluido ficasse o mais próximo possível do limite, o que maximiza o amortecimento da estrutura, mantendo as hipóteses básicas de cálculo válidas. A Tabela 2 mostra o dimensionamento de cada um dos TLCDs com diafragma. A Figura 24

Tabela 2- TLCDs com diafragma utilizados para análise de shear building de 1 pavimento

Diametro (cm)	Nível d'Água (cm)	Largura (cm)	Quantidade	Frequência Natural (rad/s)	Porcentagem de Massa	$\frac{Ac}{A}$
-	-	-	0	-	0,00 %	-
15	40	187	1	2,71	0,47 %	0,32
15	40	187	5	2,71	2,35 %	0,36
15	40	187	10	2,71	4,70 %	0,45

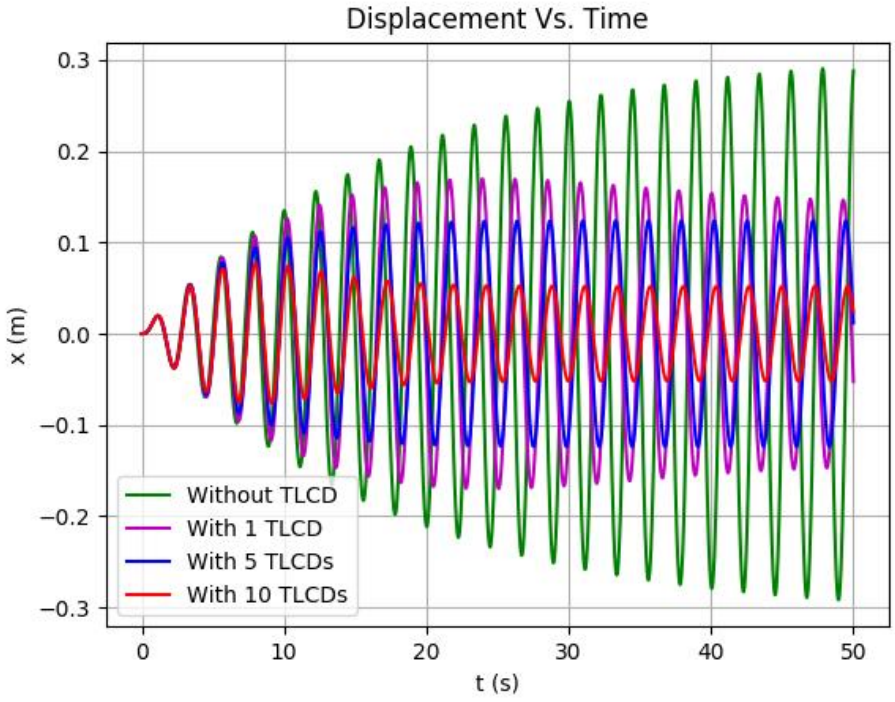

Figura 24 – Comparação entre as respostas dinâmicas de um shear building de 1 pavimento equipado com 0, 1, 5 e 10 TLCDs com diafragma

Como se pode observar, a introdução do diafragma afeta os resultados obtidos, gerando um menor amortecimento da estrutura. No entanto, o uso desse mecanismo é necessário para garantir a validade dos resultados. Ainda assim, foi possível alcançar uma redução de aproximadamente 58% no deslocamento máximo da estrutura utilizando 5 TLCDs.

5.2.2.Utilizando PTLCD

A segunda estratégia implementada foi a de se utilizar um único PTLCD e variar a massa do amortecedor. Começou-se utilizando o modelo de TLCD do caso anterior. Em seguida, aumentou-se as dimensões do PTLCD e ajustou-se a pressão de ar para manter o amortecedor sintonizado. Foi necessário ajustar a área do diafragma da mesma forma que feita anteriormente. Vale ressaltar que em todos os casos manteve-se a relação *b/L* constante. A Tabela 3 mostra todos os dimensionamentos de PTLCD utilizados nessa análise, enquanto a Figura 25 mostra a resposta dinâmica da estrutura em ressonância para cada dimensionamento de PTLCD.

Tabela 3 - PTLCDs utilizados para análise de shear building de 1 pavimento

Diametro (cm)	Nível d'Água (cm)	Largura (cm)	Altura da Coluna de Ar (cm)	Pressão de Ar (cm)	Frequência Natural (rad/s)	Porcentagem de Massa	$\frac{Ac}{A}$
15	40	187	40	0	2,71	0,47 %	0,45
20	53	250	40	0.0093	2,71	1,11%	0,37
20	107	500	40	0,046	2,71	2,23%	0,34
30	107	500	40	0,046	2,71	5,03%	0,53

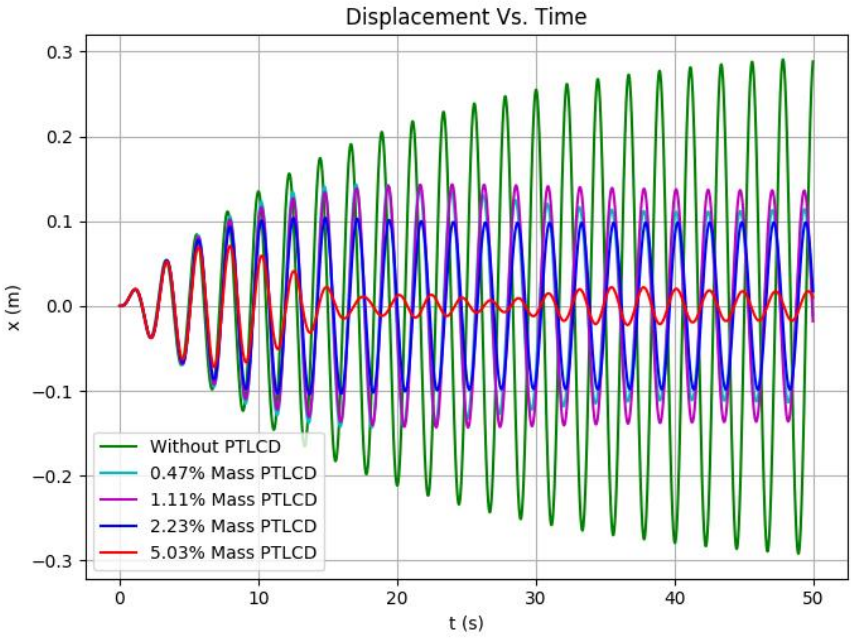

Figura 25 – Comparação entre as respostas dinâmicas de um shear building de 1 pavimento equipado com diferentes PTLCDs

Pode se observar que, assim como no caso anterior, o aumento da massa gerou um aumento significativo no amortecimento. Além disso, observa-se que a resposta do caso do PTLCD com massa do amortecedor igual a 2,23 % da massa da estrutura foi similar à resposta do caso do TLCD com massa do amortecedor igual a 2,35% da massa da estrutura. Isso indica que os dois sistemas são aproximadamente equivalentes, cabendo ao projetista escolher qual opção melhor se adequa ao seu projeto.

5.3. Análise Dinâmica de um Shear Building de 5 Pavimentos

Analisou-se um edifício de cinco pavimentos em que cada pavimento possuía uma massa de 10 toneladas, altura de 3 metros e duas colunas de concreto armado quadradas com lado de 35 cm. Utilizando uma excitação senoidal, plotou-se o gráfico de deslocamento máximo em função da frequência de excitação e encontrou-se que a frequência natural da estrutura, já incluso o peso do TLCD era de 14,84 rad/s. Essa frequência se mostrou muito alta para se utilizar um amortecedor simples e, portanto, escolheu-se o amortecedor pressurizado. Os detalhes do dimensionamento do

PTLCD e do modelo da estrutura podem ser vistos na Figura 26, sendo que a área do diafragama utilizado corresponde a 51% da área do PTLCD.

Fez-se, então a comparação das respostas dinâmica da estrutura em ressonância com e sem o amortecedor, que pode ser visualizado na Figura 27. Nota-se que a utilização do PTLCD reduziu o deslocamento máximo em aproximadamente 45%, e que no regime permanente essa redução é de aproximadamente 80%. Tais resultados mostram que a utilização desse sistema de amortecimento é bastante interessante e que sua implementação em edifícios esbeltos é viável.

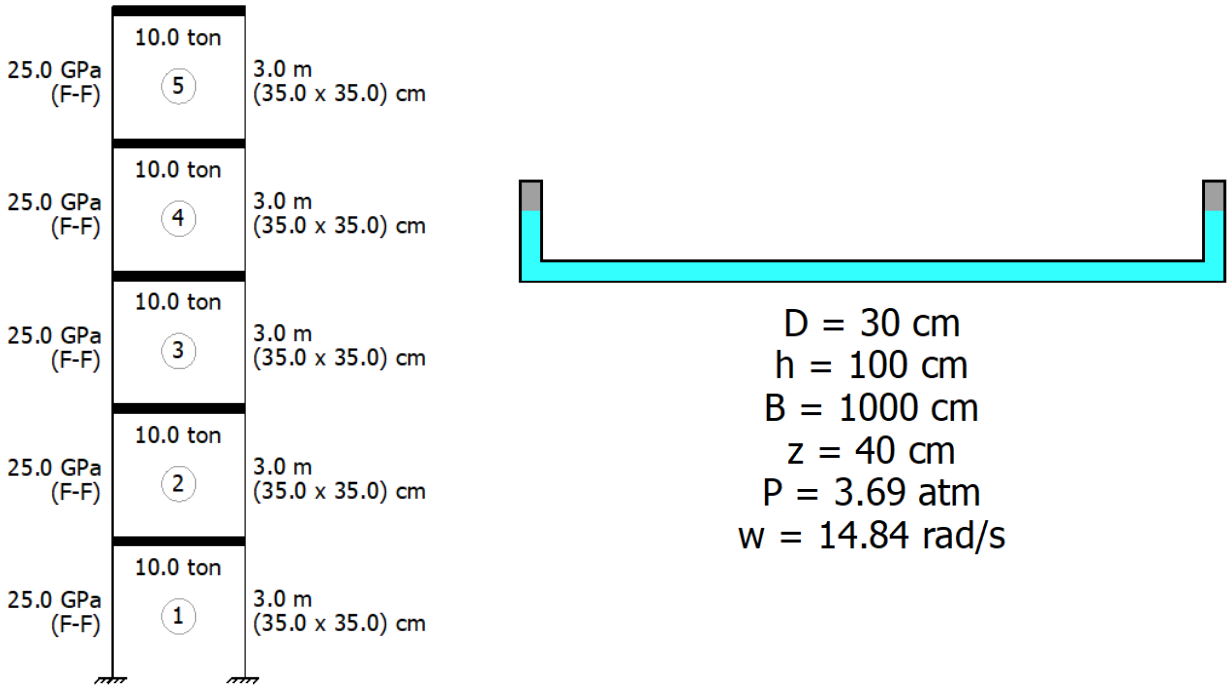

Figura 26 – Modelo de shear building com 5 pavimentos e dimensionamento do PTLCD utilizado

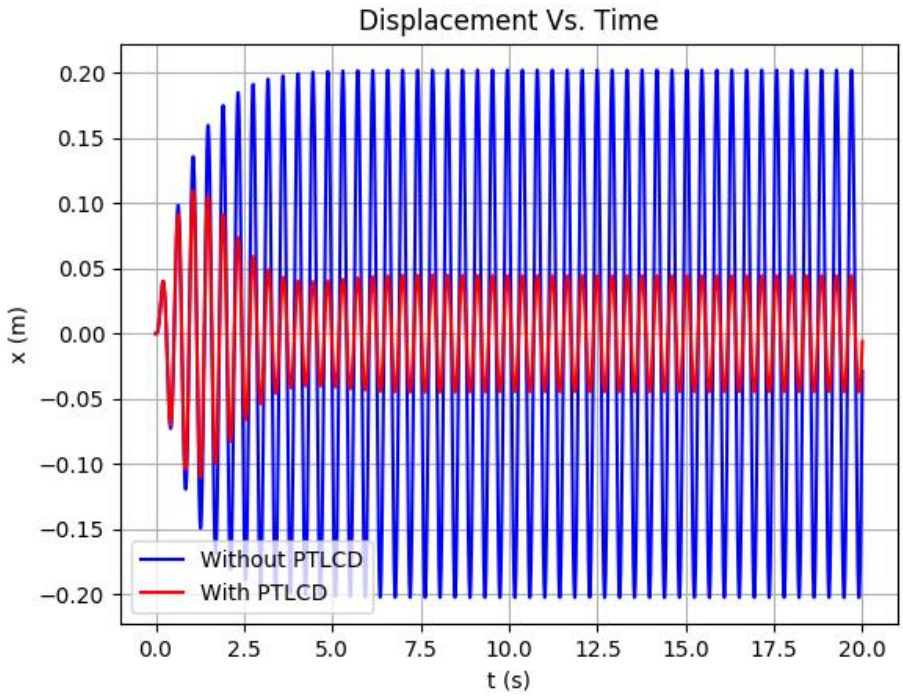

Figura 27 – Comparação entre a resposta dinâmica do ultimo pavimento de um shear building de 5 pavimenots em ressonância com e sem o PTLCD

5.4. Análise Dinâmica de um Shear Building de 20 Pavimentos

De forma semelhante ao exemplo anterior, analisou-se um edifício de 20 pavimentos em que cada pavimento possuía uma massa de 10 toneladas, altura de 3 metros e duas colunas de concreto armado quadradas com lado de 35 cm. Fazendo o gráfico de deslocamento máximo em função da frequência de excitação e encontrou-se que a frequência natural da estrutura, já incluso o peso do TLCD era de 3,98 rad/s. Para esse exemplo, decidiu-se utilizar múltiplos PTLCDs, pois, como a massa total da estrutura é muito grande, o uso de TLCDs simples levaria a uma quantidade extremamente elevada de amortecedores. A Figura 28 mostra o dimensionamento do PTLCD utilizado nesse exemplo. Já a Figura 29 mostra a resposta dinâmica do último pavimento desse *shear building* em ressonância utilizando-se de zero a três PTLCDs iguais aos da Figura 28. Cada PTLCD representa aproximadamente 2% da massa da estrutura e possui um diafragma com área igual a 55%, 100% e 100% da área do PTLCD para os casos com 1, 2 e 3 PTLCDs, respectivamente.

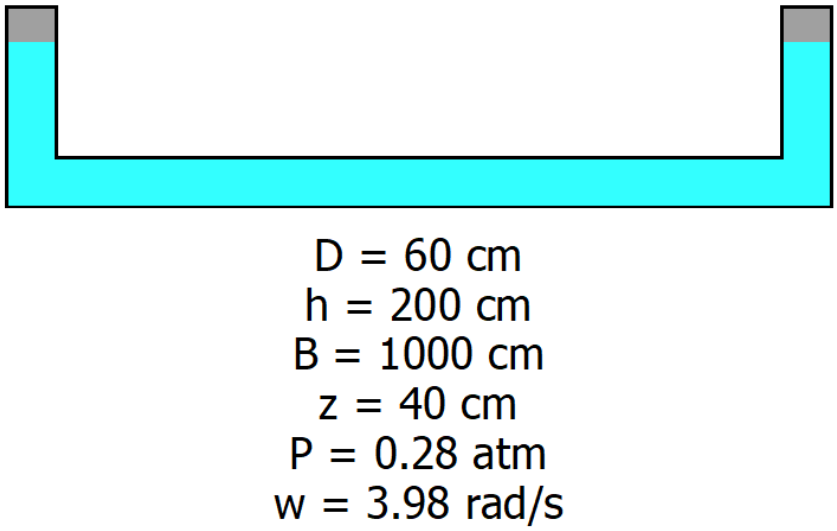

Figura 28 – Modelo PTLCD utilizado na análise de um shear building de 20 pavimentos

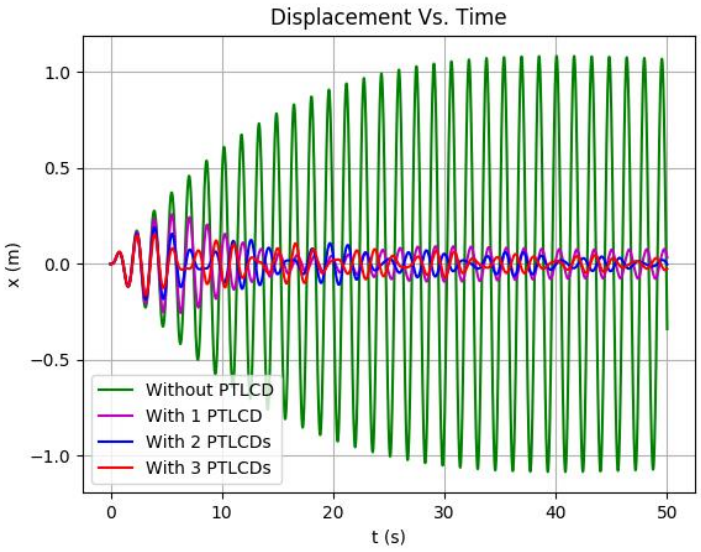

Figura 29 – Comparação entre a resposta dinâmica do ultimo pavimento de um shear building de 20 pavimentos em ressonância com zero a três PTLCDs

Os resultados obtidos nesse estudo de caso, mostram que o uso de múltiplos PTLCDs é viável e gera excelentes resultados, sendo possível reduzir a amplitude máxima de deslocamentos da estrutura em aproximadamente 75% com apenas 2% de massa adicional. No entanto, deve-se tomar cuidado com essa massa adicional, uma vez que ela é toda colocada em cima do último

pavimento, ou seja, efetivamente, a massa desse pavimento foi dobrada com a inserção de um único PTLCD.

5.5. Análise de Estrutura Submetida ao Terremoto El Centro

Utilizando a função de inserção de excitações genéricas do DynaPy, aplicou-se o sismo El Centro a uma estrutura de um pavimento projetada para ser sensível a esse sismo. Ela possuía massa de 2 toneladas, 10 m de altura e pilares de concreto com seção quadrada de lado igual a 20 cm. Já o PTLCD utilizado possuía diâmetro de 20 cm, altura do nível d'água de 50 cm, largura de 5 m, altura da câmara de ar de 40 cm e pressão de ar de 0,3 atm. A estrutura com a massa adicional do PTLCD possuía frequência natural de 6,28 rad/s e o amortecedor foi sintonizado para essa frequência. Foram testados diversos valores para a área do diafragma, sendo escolhido uma área de 20% da área do PTLCD, pois esse foi o valor que gerou menores deslocamentos na estrutura ao excitá-la com o sismo. A Figura 30 mostra o modelo de estrutura e PTLCD estudados nesse caso enquanto a Figura 31 mostra a resposta dinâmca dessa estrutura quando excitada pelo terremoto El Centro com e sem o amortecedor.

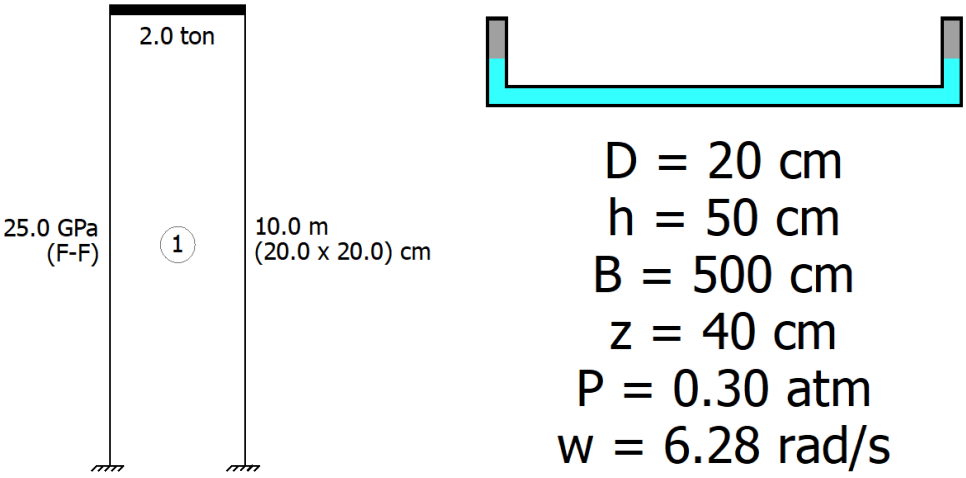

Figura 30 – Modelo de estrutura de 1 pavimento excitado pelo El Centro e PTLCD instalado sobre ela

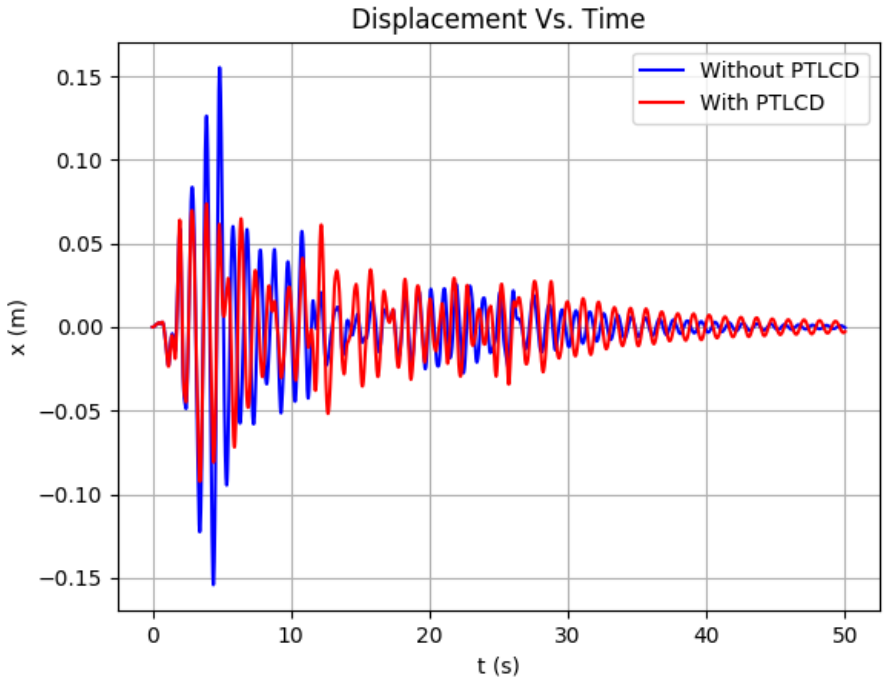

Figura 31 – Comparação entre a resposta dinâmica da estrutura de 1 pavimento excitada pelo El Centro com e sem PTLCD

Observou-se que a inserção do PTLCD reduziu o deslocamento máximo da estrutura em aproximadamente 42%, mas também aumentou o deslocamento em intervalos em que, sem o PTLCD, a estrutura vibrava pouco. O amortecedor gera, portanto, um efeito de regularização da vibração. É importante ressaltar esse ponto, uma vez que a falha de determinados elementos da estrutura podem ocorrer não só devido a um grande deslocamento em um curto período de tempo mas também a uma vibração de menor amplitude, porém de maior duração. Deve-se, portanto, analisar ambos os casos e decidir qual deles é o mais crítico.

6. Conclusões e Recomendações

6.1. Conclusões

A validação do *software* DynaPy através de comparações com soluções analíticas e com soluções numéricas obtidas com o SAP 2000 mostrou que o programa desenvolvido nesse projeto é capaz de gerar resultados de boa qualidade. A sua utilização durante o processo de pesquisa, foi bastante fácil, tendo sido possível modelar um problema, dimensionar o amortecedor e obter os resultados desejados em questão de poucos minutos. A arquitetura orientada a objeto desse programa também se mostrou muito eficiente e fácil de se programar. Com poucas alterações no código é possível implementar novos modelos de TLCD, outros tipos de estrutura e até mesmo novos métodos de solução. Acredita-se, portanto, que o DynaPy é um programa que irá ser útil para muitas pesquisas futuras e sendo ele um programa *open source*, qualquer pesquisador pode ter acesso a ele e sugerir alterações no código.

Já a análise da variação paramétrica indicou que os fatores que mais influenciam no amortecimento da estrutura são a largura e diâmetro do TLCD, enquanto a altura de lâmina d'água gera influências desprezíveis. Além disso, o aparecimento de picos locais no diagrama de DMF devido à ressonância do fluido mostra que existem dimensões do TLCD para o qual a sua instalação pode tornar o problema de vibração na estrutura mais crítico do que se ele não tivesse sido instalado. Dessa forma, o dimensionamento ideal do TLCD seria aquele que garantisse máximo amortecimento à estrutura sem gerar problemas de ressonância do fluido que prejudicassem a performance dinâmica da estrutura.

Por sua vez, os estudos de caso realizados mostrou a viabilidade da utilização de amortecedores de líquido sintonizado para o controle de vibrações. A análise de um *shear building* de 1 pavimento mostrou que tanto o uso de múltiplos TLCDs como o de um único PTLCD pode diminuir significativamente as amplitudes de vibração da estrutura. Em ambos os casos, observou-se que a utilização de um sistema de amortecedores com aproximadamente 2% da massa da estrutura pode diminuir a amplitude de vibração da estrutura em aproximadamente 50%. Já a

análise dos *shear buildings* de 5 e 20 pavimentos mostraram que o uso de PTLCDs pode ser uma solução bastante fácil de ser implementada e que gera bons resultados, seja utilizando um único ou múltipos amortecedores. Por fim, o estudo de caso utilizando o sismo El Centro comprovou a eficiência desse tipo de amortecedor para o controle de vibração de estruturas em situações mais próximas à realidade.

6.2. Recomendações

Tendo em vista a facilidade de se utilizar o *software* DynaPy e de implementar novas funcionalidades a ele, aliada aos bons resultados gerados pelo programa, acredita-se que esse pode ser utilizado para uma série de pesquisas futuras que incluam, entre outros:

- Estudar TLCDs de formas distintas e seção transversal genérica;
- Estudar a aplicação de TLCDs em outros tipos de estruturas, como edifícios flexíveis, pontes, torres, plataformas de petróleo e outros;
- Estudar o comportamento de estruturas sob o efeito de cargas aleatórias de vento, ação de sismos reais e outros carregamentos;
- Estudar o processo de otimização do dimensionamento de amortecedores para diversas estruturas diferentes.

Bibliografia

ABNT. **NBR 6123 - Forças Devido ao Vento em Edificações**. Rio de Janeiro: [s.n.], 1988.

BALEANDRA, T.; WANG, C. M.; RAKESH, G. Effectiveness of TLCD on various structural systems. **Engineering Structures**, 16 Junho 1997. 291-305.

BANERJI, P.; SAMANTA, A. Earthquake vibration control of structures using hybrid mass liquid damper. **Engineering Structures**, 2011.

BLEVINS, R. D. **Flow-Induced Vibration**. 2nd. ed. Florida: Krieger Publishing Company, 2001.

BOMTEMPO, T. B. S. **UM ESTUDO SOBRE A INFLUÊNCIA DO DECK NO COMPORTAMENTO DE PLATAFORMAS OFFSHORE FIXAS SUBMETIDAS A AÇÕES DINÂMICAS**. Universidade de Brasília. Brasília, p. 123. 2014.

CASSOLATO, M. R. **The Performance of a Tuned Liquid Damper Equipped with Inclined and Oscilating Damping Screens**. McMaster University. Hamilton. 2007.

CHOPRA, A. K. **Dynamics of Structures:** Theory and Applications to Earthquake Engineering. New Jersey: Prentice Hall, 1995.

CLOUGH, R. W.; PENZIEN, J. **Dynamics of Structures**. 3rd. ed. Berkley: Computers & Structures, 2003.

FRENCH, A. **Vibrações e Ondas**. Brasília: Universidade de Brasília, 2001.

GAO, H.; KWOK, K. C. S.; SAMALI, B. Optimization of tuned liquid. **Engineering Structures**, Agosto 1996.

HOUSNER, G. W. et al. Structural Control: Past, Present and Future. **Journal of Engineering Mechanics**, 1997.

KENNY, A.; BRODERICK, B.; MCCRUM, D. P. Optimisation of a Tuned Liquid Column Damper for Building Structures. **Recent Advances in Structural Dynamics**, Pisa, Julho 2013.

LOVE, J. S.; TAIT., M. J. Non-linear Multimodal Model for Tuned Liquid Dampers of Arbitrary Tank Geometry. **International Journal of Non-Linear Mechanics**, 2011.

NAUDASCHER, E.; ROCKWELL, D. **Flow-Induced Vibrations**. [S.l.]: International Association for Hydraulic Research, 1994.

PEDROSO, J. **Analogia Mecânica para um Estudo de uma Coluna Oscilante de Fluido Incompressível Comportando Efeitos de Rigidez e Dissipação**. UnB-FT/ENC. Brasília. 1992.

PEDROSO, L. J. **Introdução a Dinâmica de Estruturas**. Publicação didática (parte I), UnB-FT/ENC. Brasília. 2000.

PEDROSO, L. J. **Interação Fluido-Estrutura (Notas de Aula e Apostila Interna de Curso); versão preliminar**. UnB-FT/ENC. Brasília. 2003.

PEDROSO, L. J. **Formulação das Equações de Movimento e Determinação das Frequências Naturais para SS1GL**. UnB-FT/ENC. Brasília. 2005.

PESTANA, I. G. **Controlo de Vibrações em Engenharia Civil - Amortecedor de Colunas de Líquido Sintonizado**. Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa. Lisboa. 2012.

SHUM, K. M.; XU, Y. L.; GUO, W. H. Wind-induced vibration control of long span cable-stayed bridges using multiple pressurized tuned liquid column dampers. **Journal of Wind Engineering and Industrial Aerodynamics**, 19 Junho 2007.

TAIT, M. Modelling and Preliminary Design of a Structure-TLD System. **Engineering Structures**, p. 2644–2655, Outubro 2008.

TEDESCO, J. W.; MCDOUGAL, W. G.; ROSS, A. C. **Structural Dynamics:** Theory and Applications. [S.l.]: Addison Wesley Longman, 1999.

APÊNDICES

A. Estrutura do Software de Cálculo de Resposta Dinâmica da Estrutura – Protótipo Inicial

Primeiramente, são definidos os parâmetros de entrada do TLD, da estrutura e outras propriedades importantes, como a gravidade, massa específica da água e viscosidade cinemática da água, como mostrado a seguir:

```
# TLD Parameters (input parameters) - considering that the tank has appropriate
dimensions for sloshing
h = 1.0      # TLD height (m)
b = 6.0     # TLD length (m)
D = 0.9     # TLD diameter (m)

# Structure Parameters (Square cross-section (0.25x0.25) - Concrete)
E = 25e9                # Young's Module (Pa)
I = (0.25*0.25**3)/12   # Moment of Inertia (m^4)
H = 10                  # Structure height (m)
m_s = 10e3              # Structure Mass (kg)

# Other Properties
g = 9.807        # Gravity acceleration (m/s2)
ro = 998.2071    # Water specific mass (20 °C) (kg/m3)
nu = 1.003e-6    # Water kinetic viscosity (20 °C) (m2/s)
```

Em seguida, o programa calcula as demais propriedades do TLD e da estrutura, como área da seção transversal do tanque, massa, rigidez e amortecimento equivalente do fluido e frequência natural de vibração da estrutura.

```
# TLD Properties (calculated)
A = 0.25*pi*D**2
L = b + 2*h
m_f = L*A*ro                # (b + 2*h)*(0.25*pi*D**2)*ro
c_f = 8*pi*L*nu*ro
k_f = pi*(D**2)*ro*g/2
omega_f = np.sqrt((2*g)/L)

# Structure Properties
k_s = 24*E*I/(H**3)                          # Structure Stiffness
omega_s = np.sqrt(k_s/(m_s + m_f))           # Structure Natural Frequency of Vibration
Cc_s = 2*(m_s + m_f)*omega_s                 # Structure Critical Damping
c_s = Cc_s*0.025                             # Structure Damping (2,5% of critical)
```

Em seguida, o programa monta as matrizes de massa, amortecimento e rigidez do sistema, como demonstrado a seguir:

```
# Mass, Stiffness and Damping matrices (input values for CDM)
m = np.mat(([m_s + m_f, (b/L)*m_f], [(b/L)*m_f, m_f]))     # (kg)
c = np.mat(([c_s, 0], [0, c_f]))                           # (N.s/m)
k = np.mat(([k_s, 0], [0, k_f]))                           # (N/m)
```

Após montada as matrizes, o programa define o passo entre as iterações, a duração total da análise em segundos e monta o vetor de tempo, que servirá para realizar as iterações no método das diferenças finitas.

```
# Step size, total time analysed and time vector (plot precision and length
configurations)
dt = 0.01                        # Step size (s)
tt = 40                         # Total duration (s)
t = np.arange(0, tt+dt, dt)      # Time vector used om iterations (s)
```

É necessário também montar o vetor de duração do sismo separadamente, pois esse pode ter duração diferente do tempo de análise do programa. No entanto, deve-se garantir que o comprimento de ambos seja igual para que o programa não retorne um erro em sua execução.

```
# Quake duration arrays
zero = [0. for i in t]                # Null vector for use on directions with {F(t)} =
0
tq = 40                              # Duration of the quake
tq_ar = np.arange(0, tq+dt, dt)      # Quake duration vector
zeros_q = np.array(([0. for i in list(range(len(t) - len(tq_ar)))]))     # Null vector
for use on t higher than tq
```

Em seguida, define-se a frequência e amplitude do sismo e, a partir desses dados e do vetor de duração do sismo monta-se o vetor de forças externas ao longo do tempo.

```
# Assembly of the force matrix (concatenation of multiple {F(t)} for different times)
OMEGA = omega_s*r                    # (rad/s)
F0 = (m_s + m_f)*(0.10*g)            # (Acceleration amplitude)/OMEGA**2 (N)
F = np.array([F0*np.sin(OMEGA*i) for i in tq_ar])          # First Part of the quake
equation [F1(t)]
F = np.append(F, zeros_q)            # Second part of the quake equation [F1(t)]
(zeros)
F = np.vstack((F, zero))             # Assembly of force vector
[[F1(t)],[F2(t)],...,[Fn(t)]]
F = np.mat(F)
```

O próximo passo é definir as condições de contorno do problema, que no caso é dado como velocidades e posições iniciais nulas. Além disso, define-se as variáveis alpha, beta e gamma, que nada mais são do que matrizes constantes dependentes das matrizes de massa, amortecimento e rigidez.

```
x0 = np.mat(np.zeros((2, 1)))        # (m)
v0 = np.mat(np.zeros((2, 1)))     # (m/s)

alpha = np.mat(m/(dt**2) - c/(2*dt))
beta = np.mat(k - 2*m/(dt**2))
gamma = np.mat(m/(dt**2) + c/(2*dt))
```

Realiza-se então a primeira iteração para resolver a equação de movimento. Na primeira iteração são encontrados os valores de aceleração inicial e posição no tempo i = -1. Após encontrar esses valores é possível calcular o vetor de posição para o primeiro intervalo imediatamente após o início do sismo.

```
x = np.mat(np.vstack((zero, zero)))     # Defines [x] for 2 degrees of freedom (matrix)
x[:, 0] = x0[:, 0]                           # Inserts {x0} in [x] (vector into matrix)
a0 = m.I*(F[:, 0] - c*v0 - k*x0)             # Defines {a0} (vector)
xm1 = x0 - v0*dt + (a0*dt**2)/2              # Defines {x(-1)} at t = - dt (vector)
x[:, 1] = gamma.I*(F[:, 0] + beta*x[:, 0] + alpha*xm1) # Inserts {x1} in [x]
```

Todas as outras iterações são realizadas de forma automática em um loop, como mostrado abaixo:

```
def MDF(alpha, beta, gamma, t, F, x): # Central Differences Method iterations function
    for i in list(range(1, len(t[1:]))):        # Loops from t = dt to t = tt
        x[:, i+1] = gamma.I*(F[:, i] - beta*x[:, i] - alpha*x[:, i-1])

MDF(alpha, beta, gamma, t, F, x)            # Calls Central Differences Method
```

Após realizar todas as iterações, obtém-se um vetor de vetores de deslocamento ao longo do tempo para todo o tempo de análise. Utilizando as derivadas por diferenças finitas é possível, a partir desse vetor, encontrar as velocidades e acelerações ao longo do tempo.

```
i = len(t[1:])
xM1 = gamma.I*(F[:, i] - beta*x[:, i] - alpha*x[:, i-1])
xMais1 = np.concatenate((x[:, 1:], xM1), axis=1)
xMenos1 = np.concatenate((xm1, x[:, 0:-1]), axis=1)

v = (xMais1 - xMenos1)/(2*dt)           # Defines [v] matrix (first derivative of x)
a = (xMais1 - 2*x + xMenos1)/(dt**2)    # Defines [a] matrix (second derivative of x)
```

Por fim, o programa permite plotar os gráficos de deslocamento, velocidade e aceleração ao longo do tempo para ambos os graus de liberdade chamando-se uma das funções a seguir:

```
def plot_displacement(t, x, h, b, D):
    plt.plot(t, x[0].A1, 'r-')
    plt.plot(t, x[1].A1, 'b-')

    plt.title('Structure/TLD Displacements\nh = %.2f m / b =  %.2f m / D = %.2f m' %
(h, b, D))
    plt.legend(['Structure Displacement', 'TLD Displacement'])
    plt.xlabel('t (s)', fontsize=20)
    plt.ylabel('x (m)', fontsize=20)
    plt.grid()
    plt.show()


def plot_velocity(t, v, h, b, D):
    plt.plot(t, v[0].A1, 'r-')
    plt.plot(t, v[1].A1, 'b-')

    plt.title('Structure/TLD Velocities\nh = %.2f m / b =  %.2f m / D = %.2f m' % (h,
b, D))
    plt.legend(['Structure Velocity', 'TLD Velocity'])
    plt.xlabel('t (s)')
    plt.ylabel('v (m/s)')
    plt.grid()
    plt.show()
```

```
def plot_acceleration(t, a, h, b, D):
    plt.plot(t, a[0].A1, 'r-')
    plt.plot(t, a[1].A1, 'b-')

    plt.title('Structure/TLD Accelerations\nh = %.2f m / b =  %.2f m / D = %.2f m' %
(h, b, D))
    plt.legend(['Structure Acceleration', 'TLD Acceleration'])
    plt.xlabel('t (s)')
    plt.ylabel('a (m/s2)')
    plt.grid()
    plt.show()
```

B. Estrutura do Software de Geração dos Gráficos de Fator de Amplificação Dinâmica em Função da Relação de Frequências – Protótipo Inicial

Para gerar os gráficos de Fator de Amplificação Dinâmica em função da Relação de Frequências, primeiramente definiu-se uma função *hloop*, que nada mais é do que o *software* de cálculo de resposta dinâmica da estrutura apresentado anteriormente. A função *hloop* retorna o vetor de vetores de deslocamento. Define-se, então as funções *loop_h, loop_b* e *loop_D*. Cada uma dessas funções tem o objetivo de gerar um gráfico de *DMF* em função de *r* para diferentes valores de altura de lâmina d'água, largura do tanque e diâmetro do tanque, respectivamente. A arquitetura das três funções é idêntica, mudando apenas o parâmetro do TLD que é variado a cada iteração.

O primeiro passo da função é definir um vetor de iteração *h* que irá conter os valores do parâmetro que será variado, no caso a altura de lâmina d'água. Em seguida, define-se o vetor de relações de frequência de 0,1 a 2,0. Em seguida o programa calcula o deslocamento estático da estrutura e os deslocamentos dinâmicos para cada valor de *r*. Esse procedimento é repetido para todos os valores de *h*, montando-se uma matriz em que as colunas contêm os valores de deslocamento dinâmico para diferentes relações de frequência e as linhas contêm os valores de deslocamento dinâmico para diferentes valores de lamina d'água. Por fim, divide-se os deslocamentos dinâmicos pelos deslocamentos estáticos para se obter os *DMFs* e plota-se os resultados obtidos.

```
def loop_h(hmin, hmax, hstep):
    h = np.arange(hmin, hmax + hstep, hstep)
    r = np.arange(0.1, 2.0, 0.01)
    xtotal = np.zeros((len(h), len(r)))
    xref = np.zeros((len(h), 1))
    for i in tuple(range(len(h))):
        xref[i] = np.max(hloop(h=h[i], r=0.01))
```

```
    for j in tuple(range(len(r))):
        xtotal[i, j] = np.max(hloop(h=h[i], r=r[j])[-3000:])

DMF = xtotal
for i in tuple(range(len(h))):
    DMF[i] = DMF[i]/xref[i]

plt.plot(r, DMF[0], 'g-')
plt.plot(r, DMF[1], 'b-')
plt.plot(r, DMF[2], 'r-')
plt.plot(r, DMF[3], 'm-')
plt.plot(r, DMF[4], 'k-')

# plt.title('')
plt.legend(['h = %.2f m' % h[0], 'h = %.2f m' % h[1], 'h = %.2f m' % h[2],
            'h = %.2f m' % h[3], 'h = %.2f m' % h[4]])
plt.xlabel('r', fontsize=20)
plt.ylabel('DMF', fontsize=20)
plt.grid()
plt.show()
```

C. Classe de Definição dos Pavimentos

Cada pavimento da estrutura definida pelo usuário se trata de um objeto definido pela classe “Story”. Os objetos definidos por essa classe possuem atributos de massa, altura, largura, profundidade, módulo de elasticidade e tipo de apoio, que são passados pelo próprio usuário pela utilização da interface gráfica. Ao criar o objeto, a própria classe já calcula a rigidez do pavimento, sua frequência natural e taxa de amortecimento crítico.

Quando o usuário pede para se calcular a resposta dinâmica, o programa introduz o TLCD ao último pavimento e recalcula as propriedades desse. Além disso, o programa calcula o coeficiente de amortecimento para todos os pavimentos. Esse conjunto de dados, armazenados nos atributos dos objetos da classe “Story”, serão utilizados na montagem das matrizes de massa, rigidez e amortecimento do sistema. Abaixo, está apresentado o código que define essa classe.

```
class Story(object):
    def __init__(self, mass=10.e3, height=3., width=.35, depth=.35, E=25.e9,
support='Fix-Fix',tlcd=None, **kwargs):
        self.mass = mass
        self.height = height
        self.width = width
        self.depth = depth
        self.E = E
        self.support = support
        self.tlcd = tlcd

        for (i, j) in kwargs.items():
            exec('self.{} = {}'.format(i, j))

        self.I = (self.width*self.depth**3)/12
        if support == 'Fix-Fix':
            self.stiffness = 24*self.E*self.I/(self.height**3)
        elif support == 'Fix-Pin' or support == 'Pin-Fix':
            self.stiffness = 15*self.E*self.I/(self.height**3)
        elif support == 'Pin-Pin':
            self.stiffness = 6*self.E*self.I/(self.height**3)

        if self.tlcd is None:
            self.naturalFrequency = sqrt(self.stiffness/self.mass)
            self.criticalDamping = 2*self.mass*self.naturalFrequency
        else:
            self.naturalFrequency = sqrt(self.stiffness/(self.mass + self.tlcd.mass))
            self.criticalDamping = 2*(self.mass +
self.tlcd.mass)*self.naturalFrequency

    def calc_damping_coefficient(self, dampingRatio):
        self.dampingCoefficient = self.criticalDamping * dampingRatio
```

D. Classe de Definição do Amortecedor

O amortecedor da estrutura é definido pela classe “TLCD”. Assim como no caso dos pavimentos da estrutura, todas as informações do amortecedor são armazenadas nos atributos do objeto criado pela classe “TLCD”. No caso do amortecedor simples, são definidos apenas o diâmetro, largura do tubo e altura da coluna d’água. A partir desses dados, o programa calcula o comprimento total do tubo, massa de fluido, coeficiente de amortecimento e rigidez equivalentes do amortecedor. Abaixo, apresenta-se o trecho do código que define a classe “TLCD”. Para outros tipos de amortecedor, a lógica de programação é a mesma. Só o que mudam são os parâmetros que são passados e se cria uma ramificação na condicional, para que se verifique outros tipos de amortecedor. Além da definição dos amortecedores, essa classe também possui os métodos que calculam o número e Reynolds, coeficiente de atrito e o fator de correção para a análise não-linear.

```
class TLCD(object):
    def __init__(self, tlcdType='Basic TLCD', diameter=0.6, width=20., waterHeight=1.,
                 gasHeight=0.1, gasPressure=202650,
                 amount=1, contraction=1,
                 configurations=Configurations(), **kwargs):
        self.type = tlcdType
        self.diameter = diameter
        self.width = width
        self.waterHeight = waterHeight
        self.liquidSpecificMass = configurations.liquidSpecificMass
        self.kineticViscosity = configurations.kineticViscosity
        self.pipeRoughness = configurations.pipeRoughness
        self.nonLinearAnalysis = configurations.nonLinearAnalysis
        self.gravity = configurations.gravity
        self.amount = amount

        for (i, j) in kwargs.items():
            exec('self.{} = {}'.format(i, j))

        if self.type == 'Basic TLCD':
            self.length = self.width + 2 * self.waterHeight
            self.mass = pi * ((self.diameter / 2) ** 2) * self.length *
                        self.liquidSpecificMass
            self.stiffness = pi * (self.diameter ** 2) * self.liquidSpecificMass *
                             self.gravity / 2
            self.naturalFrequency = (self.stiffness / self.mass) ** 0.5
            if self.nonLinearAnalysis:
                self.dampingCoefficient = pi * self.length * self.diameter *
                                          self.liquidSpecificMass / 8
            else:
                self.dampingCoefficient = 8 * pi * self.length * self.kineticViscosity
                                          * self.liquidSpecificMass

        if self.type == 'Pressurized TLCD':
            self.gasHeight = gasHeight
```

```
            self.gasPressure = gasPressure

            self.length = self.width + 2 * self.waterHeight
            self.liquidMass = pi * ((self.diameter / 2) ** 2) * self.length *
                              self.liquidSpecificMass
            self.gasMass = 0
            self.mass = self.liquidMass + self.gasMass
            self.liquidStiffness = pi * (self.diameter ** 2) * self.liquidSpecificMass
                                   * self.gravity / 2
            self.gasStiffness = 1.4 * self.gasPressure / self.gasHeight * pi *
                                (self.diameter ** 2) / 2
            self.stiffness = self.liquidStiffness + self.gasStiffness
            self.naturalFrequency = (self.stiffness / self.mass) ** 0.5
            if self.nonLinearAnalysis:
                self.dampingCoefficient = pi * self.length * self.diameter *
                                          self.liquidSpecificMass / 8
            else:
                self.dampingCoefficient = 8 * pi * self.length * self.kineticViscosity
                                          * self.liquidSpecificMass

        self.contracionDampingConstant = self.calculate_contraction_damping_constant()


    def calculate_reynolds(self, velocity):
        return velocity * self.diameter / self.kineticViscosity

    def calculate_friction_factor(self, velocity):
        if velocity == 0.:
            return 0

        Re = self.calculate_reynolds(velocity)
        k = self.pipeRoughness
        D = self.diameter

        b = (k / (3.7 * D) - (5.16 / Re) * np.log10((k / 3.7 * D) +
            (5.09 / (Re ** 0.87))))

        if b < 0:
            return 0

        a = -2 * np.log10(b)
        f = (1 / a) ** 2
        return f

    def calculate_damping_correction_factor(self, velocity):
        f = self.calculate_friction_factor(velocity)
        return f*velocity


    def calculate_contraction_damping_constant(self):
        return 0.5 * self.liquidSpecificMass * self.area * (1 / self.contraction - 1)
** 2

    def calculate_contraction_damping(self, velocity):
        return self.contracionDampingConstant * velocity
```

E. Classe de Definição do Carregamento

Seguindo a padronização de se utilizar programação orientado a objeto, o carregamento também é definido por um objeto, dessa vez, da classe “Excitation”. Para essa classe, define-se duas possibilidade: carregamento em forma de onda senoidal ou excitação genérica. Em ambos os casos, o carregamento sempre é aplicado à base da estrutura, ou seja, é tratado como sismo. Pretende-se generalizar isso para que o carregamento possa ser aplicado também em qualquer um dos graus de liberdade do sistema.

Caso o carregamento informado ao programa seja do tipo onda senoidal, o programa armazena a amplitude, frequência, duração da excitação, tempo de análise e um booleano que define se a frequência informada foi uma frequência relativa ou absoluta. Com esses dados o programa calcula a frequência absoluta, se necessário.

Caso o carregamento informado seja do tipo excitação genérica, o programa toma um arquivo de texto contendo as informações do sismo, lê o arquivo e extrai o vetor de tempo e o vetor de aceleração e os armazena em atributos. Além disso, o nome do arquivo também é armazenado. Abaixo, mostra-se o código que define a classe “Excitation” para ambos os casos de carregamento.

```
class Excitation(object):
    def __init__(self, exctType='Sine Wave', amplitude=5., frequency=20.,
                 relativeFrequency=True, exctDuration=3., anlyDuration=5.,
                 structure=None, tlcd=None, t=None, a=None, fileName=None, **kwargs):
        self.type = exctType
        self.structure = structure
        self.tlcd = tlcd
        if self.type == 'Sine Wave':
            self.amplitude = amplitude
            self.frequency = frequency
            self.frequencyInput = frequency
            self.exctDuration = exctDuration
            self.anlyDuration = anlyDuration
            self.relativeFrequency = relativeFrequency

            self.calc_frequency()

        elif self.type == 'General Excitation':
            self.t_input = t
            self.a_input = a
            self.exctDuration = t[-1]
            self.anlyDuration = t[-1]
            self.fileName = fileName

        for (i, j) in kwargs.items():
            exec('self.{} = {}'.format(i, j))
```

```
def calc_frequency(self):
    if self.relativeFrequency:
        if self.tlcd is None:
            mass = self.structure[len(self.structure)].mass
        else:
            mass = self.structure[len(self.structure)].mass + self.tlcd.mass

        stiffness = self.structure[len(self.structure)].stiffness
        self.frequency = self.frequencyInput * sqrt(stiffness / mass)
    else:
        self.frequency = self.frequencyInput
```

F.Classe de Definição das Configurações

Para facilitar a implementação do código e organizar melhor as informações, alguns dos parâmetros essenciais para resolução do sistema foram definidas na classa “Configurations”. Essa classe contém como atributos o método de resolução da EDO, o passo de tempo utilizado nas iterações, deslocamento e velocidades iniciais, taxa de amortecimento da estrutura, massa específica e viscosidade cinemática do fluido do amortecedor, a aceleração da gravidade e número de pontos de discretização e limite máximo utilizados na análise de frequências, além do booleano “nonLinearAnalysis” que é utilizado para escolher entre a análise do amortecimento do fluido linear e não linear. Abaixo, sem encontra o código que define essa classe.

```
class Configurations(object):
    def __init__(self, method='Finite Differences Method', timeStep=0.005,
                 initialDisplacement=0., initialVelocity=0.,
                 dampingRatio=0.02,
                 liquidSpecificMass=998.2071, kineticViscosity=1.003e-6,
                 gravity=9.807, pipeRoughness=0.0015e-3,
                 dmfDiscretizationPoints=200, dmfUpperLimitFactor=2,
                 nonLinearAnalysis=True):
        self.method = method
        self.timeStep = timeStep
        self.initialDisplacement = initialDisplacement
        self.initialVelocity = initialVelocity
        self.dampingRatio = dampingRatio
        self.liquidSpecificMass = liquidSpecificMass  # Water specific mass (20 °C)
(kg/m3)
        self.kineticViscosity = kineticViscosity  # Water kinetic viscosity (20 °C)
(m2/s)
        self.gravity = gravity       # Gravity acceleration (m/s2)
        self.pipeRoughness = pipeRoughness
        self.dmfDiscretizationPoints = dmfDiscretizationPoints
        self.dmfUpperLimitFactor = dmfUpperLimitFactor
        self.nonLinearAnalysis = nonLinearAnalysis
```

G. Classe de Armazenamento da Entrada de Dados

Devido à escolha de se trabalhar com programação orientado a objeto, todas as informações do programa são definidas em objetos definidos pelas classes “Story”, “TLCD”, “Excitation” e “Configurations”, mostradas nos apêndices anteriores. Todos os objetos definidos por essas classes são armazenados nos atributos de um único objeto, definido pela classe “InputData”. Apenas os dados dos pavimentos da estrutura são armazenados de forma diferente dos demais, pois esse se encontra em um dicionário de objetos, em que a chave do dicionário é o número do pavimento e o valor é um objeto da classe “Story”. Abaixo, mostra-se o código da classe “InputData”.

```
class InputData(object):
    def __init__(self):
        self.stories = {}
        self.tlcd = None
        self.excitation = None
        self.configurations = None
```

H. Classe de Armazenamento da Saída de Dados

Da mesma forma que a entrada de dados é organizada na classe “InputData”, a saída de dados é organizada na classe “OutputData”. Essa classe contém como atributos as matrizes de massa, amortecimento e rigidez e o vetor de vetores de força do sistema. Esses dados são utilizados para calcular a resposta dinâmica da estrutura, que é armazenada no atributo “dynamicResponse”. Abaixo, tem-se o código da classe “OutputData”.

```
class OutputData(object):
    def __init__(self, massMatrix, dampingMatrix, stiffnessMatrix, forceMatrix,
configurations):
        self.massMatrix = massMatrix
        self.dampingMatrix = dampingMatrix
        self.stiffnessMatrix = stiffnessMatrix
        self.forceMatrix = forceMatrix

        self.dynamicResponse = ODESolver(self.massMatrix, self.dampingMatrix,
self.stiffnessMatrix, self.forceMatrix, configurations)
```

I. Função de Montagem da Matriz de Massa

A montagem da matriz de massa é feita através de uma função que toma o dicionário de pavimentos da estrutura e o objeto amortecedor, que foram armazenados em “InputData”, e retorna a matriz de massa. Caso não haja um amortecedor definido, a função gera a matriz de massa sem o grau de liberdade do amortecedor. A montagem é feita criando-se uma matriz com o número correto de graus de liberdade e colocando-se as massas de cada pavimento na diagonal principal da matriz. Caso haja um ou mais amortecedores, suas massas são adicionadas nos últimos elementos da matriz e somada com a massa do último pavimento. Além disso, faz-se o acoplamento das massas entre os amortecedores e o último pavimento da estrutura. Mostra-se abaixo, o código que define essa função.

```
def assemble_mass_matrix(stories, tlcd):
    if tlcd is None:
        n = len(stories)

        M = np.mat(np.zeros((n, n)))

        for i in range(n):
            M[i, i] = stories[i + 1].mass
    else:
        lastStory = len(stories) - 1
        n = len(stories) + tlcd.amount

        M = np.mat(np.zeros((n, n)))

        for i in range(lastStory + 1):
            M[i, i] = stories[i + 1].mass

        M[lastStory, lastStory] += tlcd.mass * tlcd.amount
        for i in range(lastStory + 1, n):
            M[i, i] = tlcd.mass
            M[i, lastStory] = (tlcd.width / tlcd.length) * tlcd.mass
            M[lastStory, i] = (tlcd.width / tlcd.length) * tlcd.mass
```

J. Função de Montagem da Matriz de Amortecimento

Da mesma forma que a matriz de massa, a matriz de amortecimento é gerada por uma função que toma o dicionário de pavimentos e o objeto amortecedor. Da mesma forma, verifica-se a existência de amortecedor para gerar uma matriz com o número apropriado de graus de liberdade. Gerada essa matriz, basta introduzir os coeficientes de amortecimento de cada pavimento na diagonal principal da matriz e, caso haja, introduzir o coeficiente de amortecimento dos amortecedores nos últimos elementos da matriz. O código abaixo mostra a função supracitada.

```
def assemble_damping_matrix(stories, tlcd):
    if tlcd is None:
        n = len(stories)

        C = np.mat(np.zeros((n, n)))

        for i in range(n):
            C[i, i] = stories[i + 1].dampingCoefficient
    else:
        lastStory = len(stories) - 1
        n = len(stories) + tlcd.amount

        C = np.mat(np.zeros((n, n)))

        for i in range(lastStory + 1):
            C[i, i] = stories[i + 1].dampingCoefficient

        for i in range(lastStory + 1, n):
            C[i, i] = tlcd.dampingCoefficient

    return C
```

K. Função de Montagem da Matriz de Rigidez

Seguindo a mesma lógica das matrizes de massa e amortecimento, a matriz de rigidez é gerada por uma função que toma o dicionário de pavimentos e o objeto amortecedor. Novamente, verifica-se a existência de amortecedor para gerar uma matriz com o número correto de graus de liberdade. Em seguida, adiciona-se o coeficiente de rigidez de cada pavimento na diagonal principal e, caso haja, a rigidez equivalente do fluido nos últimos elementos da matriz. Aos elementos imediatamente acima e imediatamente a esquerda da diagonal principal subtrai-se a rigidez do pavimento definido por aquela linha e ao ao elemento anterior da diagonal principal soma-se a rigidez desse pavimento.

```
def assemble_stiffness_matrix(stories, tlcd):
    if tlcd is None:
        n = len(stories)

        K = np.mat(np.zeros((n, n)))

        for i in range(n):
            K[i, i] = stories[i + 1].stiffness

        for i in range(n, 1, -1):
            K[i - 1, i - 2] = -stories[i].stiffness
            K[i - 2, i - 1] = -stories[i].stiffness
            K[i - 2, i - 2] += stories[i].stiffness
    else:
        lastStory = len(stories) - 1
        n = len(stories) + tlcd.amount

        K = np.mat(np.zeros((n, n)))

        for i in range(lastStory + 1):
            K[i, i] = stories[i + 1].stiffness

        for i in range(lastStory + 1, 1, -1):
            K[i - 1, i - 2] = -stories[i].stiffness
            K[i - 2, i - 1] = -stories[i].stiffness
            K[i - 2, i - 2] += stories[i].stiffness

        for i in range(lastStory + 1, n):
            K[i, i] = tlcd.stiffness

    return K
```

L. Função de Montagem do Vetor de Vetores de Carregamento

Como esse projeto trata de problemas de dinâmica das estruturas, os carregamentos estudados são dependentes do tempo. Além disso, estuda-se sistemas com vários graus de liberdade. Portanto, torna-se conveniente representar os carregamentos como um vetor que varia ao longo do tempo, em que cada elemento do vetor é responsável por um grau de liberdade. Para representar essa variação ao longo do tempo, em se tratando de um modelo numérico, em que se discretiza o tempo, utiliza-se um vetor que contém esses vetores de carregamento, sendo que cada elemento desse vetor maior representa um instante de tempo. Na programação desse conceito, forma-se uma matriz, em que as linhas representam os graus de liberdade e as colunas representam os instantes de tempo.

Portanto, a geração dessa matriz (ou vetor de vetores) é feita por uma função que toma o objeto de carregamento, a matriz de massa e o objeto de configurações. Primeiramente define-se um vetor de tempo que contém todos os instantes de tempo que serão utilizados nas iterações. Junto a ele, define-se um vetor de tempo de menor ou igual comprimento, que representa os instantes em que há efetivamente um carregamento sendo aplicado. Em seguida define-se o vetor de forças com comprimento igual ao do vetor total de tempo. Adiciona-se então as linhas da matriz de acordo com o número de graus de liberdade do sistema, levado em consideração a existência ou não de amortecedor.

Então, para cada grau de liberdade e para cada instante de tempo, preenche-se a matriz de carregamento com a força equivalente, que é a massa do pavimento multiplicada pela aceleração imposta. Caso a excitação seja do tipo onda senoidal, utiliza-se uma função senoidal para definir a aceleração em cada instante de tempo. Já se a excitação for genérica, utliza-se a aceleração definida no arquivo de texto usado para carregar a excitação genérica. Nos instantes de tempo em que essa excitação não está definida no arquivo de texto, faz-se uma interpolação linear entre os pontos mais próximos. A seguir, mostra-se o código utilizado para implementar essa função.

```
def assemble_force_matrix(excitation, mass, configurations):
    tlcd = excitation.tlcd
    step = configurations.timeStep
    totalTimeArray = np.mat(np.arange(0, excitation.anlyDuration + step, step))
    excitationTimeArray = np.mat(np.arange(0, excitation.exctDuration + step, step))
    force = 0. * totalTimeArray
    if tlcd is None:
        numberOfStories = mass.shape[0]
    else:
        numberOfStories = mass.shape[0] - tlcd.amount

    for i in range(numberOfStories - 1):
        force = np.concatenate((force, 0. * totalTimeArray), 0)

    if excitation.type == 'Sine Wave':
        for i in range(force.shape[0]):
            storyMass = mass[i, i]
            forceAmplitude = storyMass * excitation.amplitude
            for j in range(excitationTimeArray.shape[1]):
                force[i, j] = forceAmplitude * np.sin(excitation.frequency *
totalTimeArray[0, j])

        if tlcd is None:
            return force
        else:
            for i in range(tlcd.amount):
                force = np.concatenate((force, 0. * force[0, :]), 0)
            return force
    elif excitation.type == 'General Excitation':
        a = []
        t0 = 0
        time = [round(t / step, 0) * step for t in list(totalTimeArray.A1)]
        for t in time:
            if t in excitation.t_input:
                t0 = t
                t0_index = excitation.t_input.index(t)
                a.append(excitation.a_input[t0_index])
            else:
                t1 = excitation.t_input[t0_index + 1]
                a0 = excitation.a_input[t0_index]
                a1 = excitation.a_input[t0_index + 1]
                at = ((a1 - a0) / (t1 - t0)) * (t - t0) + a0
                a.append(at)

        a = np.array(a)
        for i in range(force.shape[0]):
            storyMass = mass[i, i]
            force[i, :] = storyMass * a

        if tlcd is None:
            return force
        else:
            for i in range(tlcd.amount):
                force = np.concatenate((force, 0. * force[0, :]), 0)
            return force
```

M. Classe de Resolução da Equação de Movimento

A solução da equação de movimento é realizada dentro da classe “ODESolver” para que as várias informações obtidas no processo sejam armazenadas em atributos dessa classe. A simples criação de um objeto dessa classe já executa o método de solução da equação. Devem ser passados para esse classe as matrizes de massa, amortecimento, rigidez e o vetor de vetores de carregamento.

Primeiramente a classe roda a função “unpack” que simplesmente renomeia vários dos atributos da classe e cria os vetores de deslocamento, velocidade e aceleração, que serão preenchidos na resolução, e o vetor de tempo, que será utilizado para fazer as iterações. Em seguida, a classe executa o método das diferenças finitas para resolver a equação de movimento, como descrito no capítulo 3. Obtém-se os deslocamentos em função do tempo e fazendo a derivação por diferenças finitas, obtém-se também a velocidade e a aceleração.

Caso haja a presença de amortecedor, o programa executa a versão não linear da rotina de diferenças finitas. Nessa versão, o programa calcula a cada iteração a velocidade do fluido em duas iterações anteriores. Esse valor é utilizado para calcular o fator de correção do amortecimento do fluido e alterar a matriz de amortecimento a cada passa de iteração. Como o método utiliza uma discretização com passos de tempo muito curtos, se torna viável aproximar a velocidade instantânea do fluido pela velocidade do mesmo em dois passos de iteração anteriores, uma vez que esses dois valores tendem a se tornar muito próximos ao se adotar passas de tempo pequenos.

Com esses resultados, pode-se fazer o pós processamento com a biblioteca de plotagem Matplotlib. O código que define essa classe é mostrado a seguir:

```
class ODESolver(object):
    def __init__(self, mass, damping, stiffness, force,
configurations=Configurations(), tlcd=None):
        self.mass = mass
        self.damping = damping
        self.stiffness = stiffness
        self.force = force
        self.configurations = configurations
        self.tlcd = tlcd

        if configurations.method == 'Finite Differences Method':
            if configurations.nonLinearAnalysis and (self.tlcd is not None):
                self.fdm_solver_non_linear()
            else:
                self.fdm_solver()
```

```
    def unpack(self):
        self.M = self.mass
        self.C = self.damping
        self.K = self.stiffness
        self.F = self.force
        self.dt = self.configurations.timeStep
        self.x0 = self.configurations.initialDisplacement
        self.v0 = self.configurations.initialVelocity

        self.x = 0. * self.F
        self.v = 0. * self.F
        self.a = 0. * self.F
        self.t = [i * self.dt for i in range(self.F.shape[1])]

        self.x[:, 0] = self.x0
        self.v[:, 0] = self.v0

        self.a0 = self.M.I * (self.F[:, 0] - self.C * self.v[:, 0] - self.K *
self.x[:, 0])
        self.a[:, 0] = self.a0

    def fdm_solver(self):
        self.unpack()

        self.alpha = (self.M / (self.dt ** 2) - self.C / (2 * self.dt))
        self.beta = (self.K - 2 * self.M / (self.dt ** 2))
        self.gamma = (self.M / (self.dt ** 2) + self.C / (2 * self.dt))

        self.xm1 = self.x[:, 0] - self.v[:, 0] * self.dt + (self.a[:, 0] * self.dt **
2) / 2
        self.x[:, 1] = self.gamma.I * (self.F[:, 0] - self.beta * self.x[:, 0] -
self.alpha * self.xm1)

        for i in list(range(1, len(self.t[1:]))):
            self.x[:, i + 1] = self.gamma.I * (self.F[:, i] - self.beta * self.x[:, i]
- self.alpha * self.x[:, i - 1])

        i = len(self.t[1:])
        self.xM1 = self.gamma.I * (self.F[:, i] - self.beta * self.x[:, i] -
self.alpha * self.x[:, i - 1])
        self.xMais1 = np.concatenate((self.x[:, 1:], self.xM1), axis=1)
        self.xMenos1 = np.concatenate((self.xm1, self.x[:, 0:-1]), axis=1)

        self.v = (self.xMais1 - self.xMenos1) / (2 * self.dt)
        self.a = (self.xMais1 - 2 * self.x + self.xMenos1) / (self.dt ** 2)

    def fdm_solver_non_linear(self):
        self.unpack()

        C_original = copy(self.C)
        correctionStart = self.C.shape[1] - 1
        correctionStop = correctionStart - self.tlcd.amount
        self.dampingVelocityArray = copy(self.v[-1, :])

        velocity = self.dampingVelocityArray[0, 0]

        correctionFactor = self.tlcd.calculate_damping_correction_factor(velocity)
        for i in range(correctionStart, correctionStop, -1):
            self.C[i, i] *= correctionFactor

        self.alpha = (self.M / (self.dt ** 2) - self.C / (2 * self.dt))
```

```
        self.beta = (self.K - 2 * self.M / (self.dt ** 2))
        self.gamma = (self.M / (self.dt ** 2) + self.C / (2 * self.dt))

        self.xm1 = self.x[:, 0] - self.v[:, 0] * self.dt + (self.a[:, 0] * self.dt **
2) / 2
        self.x[:, 1] = self.gamma.I * (self.F[:, 0] - self.beta * self.x[:, 0] -
self.alpha * self.xm1)

        for i in list(range(1, len(self.t[1:]))):
            if i >= 2:
                self.C = copy(C_original)
                self.dampingVelocityArray[0, i + 1] = (self.x[-1, i-2] - self.x[-1,
i]) / (2 * self.dt)
                velocity = abs(self.dampingVelocityArray[0, i + 1])

                correctionFactor =
self.tlcd.calculate_damping_correction_factor(velocity)
                contractionDampingCoefficient =
self.tlcd.calculate_contraction_damping(velocity)

                for j in range(correctionStart, correctionStop, -1):
                    self.C[j, j] *= correctionFactor
                    self.C[j, j] += contractionDampingCoefficient


                self.alpha = (self.M / (self.dt ** 2) - self.C / (2 * self.dt))
                self.beta = (self.K - 2 * self.M / (self.dt ** 2))
                self.gamma = (self.M / (self.dt ** 2) + self.C / (2 * self.dt))

            self.x[:, i + 1] = self.gamma.I * (self.F[:, i] - self.beta * self.x[:, i]
- self.alpha * self.x[:, i - 1])

        i = len(self.t[1:])
        self.xM1 = self.gamma.I * (self.F[:, i] - self.beta * self.x[:, i] -
self.alpha * self.x[:, i - 1])
        self.xMais1 = np.concatenate((self.x[:, 1:], self.xM1), axis=1)
        self.xMenos1 = np.concatenate((self.xm1, self.x[:, 0:-1]), axis=1)

        self.v = (self.xMais1 - self.xMenos1) / (2 * self.dt)
        self.a = (self.xMais1 - 2 * self.x + self.xMenos1) / (self.dt ** 2)
```

N. Classe de Resolução do Sistema – Resposta Dinâmica

Todas as classes e funções dos apêndices C a M são partes de módulos auxiliares do programa que são importados no arquivo principal. A classe “RunSimulationThread” é uma classe do arquivo principal responsável por tomar os dados de entrada, salvos pela classe “InputData”, gerar as matrizes de massa, amortecimento, rigidez e força e executar a rotina de solução da equação de movimento, gerando assim um objeto da classe “OutputData”. Todo esse procedimento pode levar alguns segundos em computadores baixa ou média performance. Em função disso, utilizou-se o conceito de *multithreading* na implementação dessa classe. Isso significa que todo o processamento de resolução do sistema é realizado em uma linha de processamento separada do resto do programa, o que garante que o programa principal não irá travar durante a execução desse código. Os resultados obtidos nesse processamento são retornados ao programa principal através de um sinal emitido por essa classe ao término de sua execução. Abaixo, mostra-se o código da classe “RunSimulationThread”.

```
class RunSimulationThread(QThread):
    mySignal = pyqtSignal(OutputData)

    def __init__(self, inputData_, parent=None):
        super(RunSimulationThread, self).__init__(parent)
        self.inputData = inputData_

    def run(self):
        # Assemble matrices
        mass = assemble_mass_matrix(self.inputData.stories, self.inputData.tlcd)
        damping = assemble_damping_matrix(self.inputData.stories, self.inputData.tlcd)
        stiffness = assemble_stiffness_matrix(self.inputData.stories,
self.inputData.tlcd)
        force = assemble_force_matrix(self.inputData.excitation, mass,
self.inputData.configurations)

        outputData_ = OutputData(mass, damping, stiffness, force,
self.inputData.configurations)
        self.mySignal.emit(outputData_)
```

O. Classe de Resolução do Sistema – Análise de Frequência

Analogamente à classe “RunSimulationThread”, a classe “RunSetOfSimulationsThread” também é definida no arquivo principal do programa e é implementada usando *multithreading*. Essa classe é responsável por fazer a análise dinâmica da estrutura para diversas frequências diferentes, coletando o máximo deslocamento em cada análise. Dessa forma, é possível gerar um gráfico de deslocamento máximo em função da frequência de excitação. A implementação dessa classe é muito similar à anterior, diferindo apenas em alguns detalhes. Primeiramente, utiliza-se um loop para executar a análise para diversas frequências. Segundo, define-se um outro sinal, responsável por mandar a porcentagem do código que já foi concluída. Essa porcentagem será passada para uma barra de progresso, que se faz necessária devido ao longo tempo de processamento dessa classe, mesmo em computadores de alta performance. Abaixo, mostra-se o código utilizado nessa implementação.

```
class RunSetOfSimulationsThread(QThread):
    mySignal = pyqtSignal(list)
    percentageSignal = pyqtSignal(float)

    def __init__(self, inputData_, frequencies, parent=None):
        super(RunSetOfSimulationsThread, self).__init__(parent)
        self.inputData = inputData_
        self.frequencies = frequencies

    def run(self):
        displacmentList = []
        dmfList = []
        totalIter = len(self.frequencies)

        self.mass = assemble_mass_matrix(self.inputData.stories, self.inputData.tlcd)
        self.damping = assemble_damping_matrix(self.inputData.stories,
self.inputData.tlcd)
        self.stiffness = assemble_stiffness_matrix(self.inputData.stories,
self.inputData.tlcd)

        for i, j in zip(self.frequencies, range(len(self.frequencies))):
            resp = self.simulation(self.inputData, i)
            x = resp[0]
            dmf = resp[1]
            displacmentList.append(x)
            dmfList.append(dmf)

            percentageDone = (j + 1) / totalIter * 100
            self.percentageSignal.emit(percentageDone)
        signal = [self.frequencies, displacmentList, dmfList]
        self.mySignal.emit(signal)
```

```
    def simulation(self, inputData_, frequency):
        inputData_.excitation.frequencyInput = frequency
        inputData_.excitation.relativeFrequency = False
        inputData_.excitation.calc_frequency()

        force = assemble_force_matrix(inputData_.excitation, self.mass,
inputData_.configurations)

        outputData_ = OutputData(self.mass, self.damping, self.stiffness, force,
inputData_.configurations)
        return [outputData_.maxDisplacement, outputData_.DMF]
```