**RESUMO**

O presente trabalho destinou-se em projetar uma aplicação na linguagem de programação C# (C Sharp) capaz de analisar os movimentos oscilatórios em vigas sujeitas a vibrações livre e forçada através do Método de Elementos Finitos, considerando a influência de absorvedores dinâmicos de vibração e, principalmente, chapas piezoelétricas. Este código foi desenvolvido seguindo os padrões SOLID e API RESTful para conferir manutenibilidade e simplicidade e facilitar seu entendimento.

O principal objetivo deste estudo foi analisar a vibração em uma viga com chapas piezoelétricas em diferentes configurações, comparando-se os potenciais piezoelétricos e as deflexões para o caso sem as chapas. O modelo analisado consistiu em uma viga de alumínio pinada em suas duas extremidades, vista somente no plano.

**Palavras chave**: Vibrações. Piezoelétrico. Programação Orientada a Objetos.

Introdução

Recentemente, percebe-se um crescente uso no método dos Elementos Finitos para realizar análises estruturais no ramo da engenharia. Isso ocorre pois, com advento desta tecnologia, foi possível simular em softwares as condições previstas de trabalho da peça ou equipamento com grande precisão e confiabilidade, propiciando redução em custos.

Seguindo com esta linha de raciocínio, buscou-se com este projeto desenvolver um código matemático capaz de analisar os movimentos oscilatórios em vigas submetidas a forçamento harmônico, levando em consideração, principalmente, a ação de chapas piezoelétricas e os potenciais gerados por estas e, em segundo plano, absorvedores dinâmicos de vibração. Além disso, pretende-se comparar os resultados obtidos para as frequências naturais de vibração e deflexões máximas com vigas sem os componentes supracitados.

Busca-se com este projeto desenvolver um código matemático em linguagem C# capaz de analisar os movimentos oscilatórios em vigas sobre ação de uma força perpendicular a estas, utilizando cálculos numéricos e levando em consideração a ação de absorvedores dinâmicos de vibração ou chapas piezoelétricas nestas barras. Além disso, pretende-se comparar os resultados obtidos a vigas sem os componentes supracitados. Por fim, serão feitos programas adicionais em linguagem de programação C com rotinas específicas para dar continuidade à pesquisa desenvolvida pelo professor orientador deste trabalho.

Por se tratar de uma pesquisa que será continuada pelo professor orientador junto a outros alunos, o programa, que realizará os cálculos supracitados, seguirá o padrão API RESTful, que é uma interface que fornece dados em um formato padronizado baseado em requisições HTTP, e o padrão SOLID, relacionada a programação orientada a objetos, que estabelece alguns princípios para tornar o código mais limpo, simples e ter manutenibilidade.

EXPLICAÇÃO DOS CONCEITOS:

1. API RESTful
2. Padrão SOLID

ANÁLISE DE CORPO RÍGIDO

A fim de iniciar os estudos sobre vibrações, buscou-se, analisar modelos simples envolvendo um conjunto massa-mola/amortecedor com um grau de liberdade e outro com dois graus de liberdade excitados harmonicamente, variando o coeficiente de amortecimento, para se obter vibrações livres harmônicas, subamortecidas, criticamente amortecidas e superamortecidas. Para as soluções numéricas foram escolhidos o método Runge Kutta de Quarta Ordem, que consiste em um método iterativo para resolver equações diferenciais com problemas de valor inicial, pois os exemplos atuais se enquadram neste caso, em que foi considerado o deslocamento e a velocidade iniciais iguais a zero. O algoritmo base utilizado para este método está descrito no Apêndice 1.

A SE PENSAR: e Newmark-β e seus resultados foram comparados.

Para esta tarefa, primeiramente, será analisado o diagrama de corpo livre para cada caso, seguindo as diretrizes da segunda lei de Newton, para obter a equação diferencial de movimento que servirá de input para o modelo matemático.

MODELO COM UM GRAU DE LIBERDADE

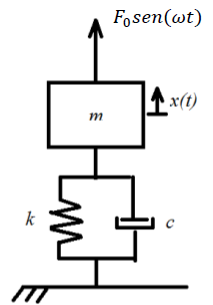


Imagem XX – Modelo massa-mola/amortecedor com um grau de liberdade

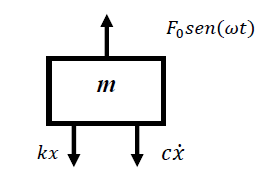


Imagem XX – Diagrama de corpo livre para modelo massa-mola/amortecedor com um grau de liberdade

De acordo com a segunda lei de Newton e assumindo o sentido positivo para cima, temos que:

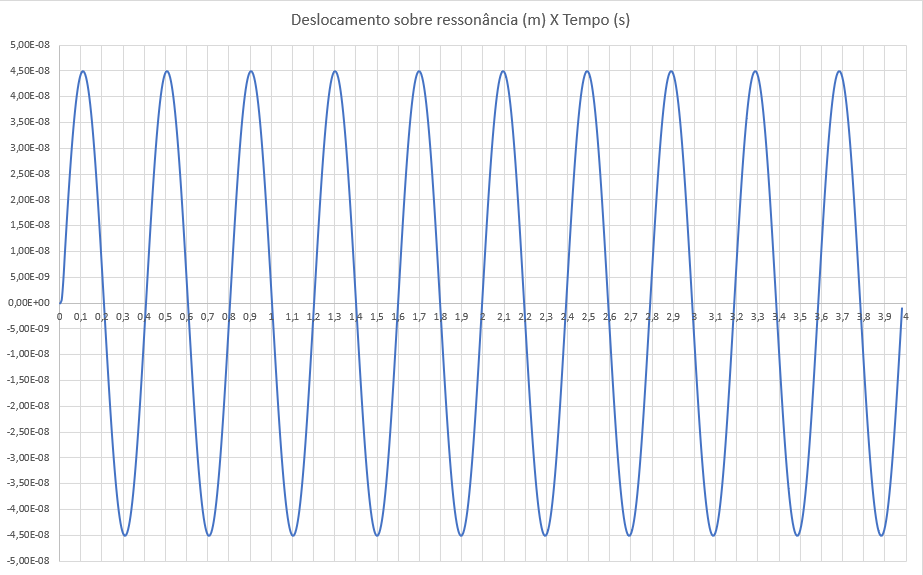
Vale salientar que a aceleração, representada por a, pode ser reescrita como a segunda derivada do deslocamento, que neste caso é expressado por x.

Portanto, a equação YY pode ser reescrita conforme abaixo e será possível obter a equação diferencial de movimento para um conjunto massa-mola/amortecedor com um grau de liberdade.

Já que esta é uma equação diferencial de segunda ordem, pretende-se converter essa equação em um sistema para facilitar a utilização do método numérico supracitado. Vale ressaltar que as mudanças a serem feitas serão baseadas nas relações bases entre deslocamento, velocidade e aceleração.

Isolando a derivada da velocidade para adequar-se ao padrão:

RESULTADOS



MODELO COM DOIS GRAUS DE LIBERDADE

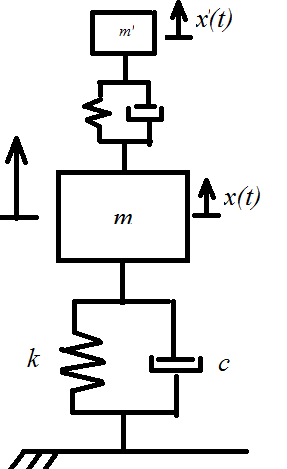


Imagem XX – Modelo massa-mola/amortecedor com dois graus de liberdade

ANÁLISE POR ELEMENTOS FINITOS

Neste capítulo é apresentado o objetivo principal da pesquisa: análise por elementos finitos de vigas com e sem chapas piezoelétricas sob excitação harmônica. Será utilizado a equação diferencial de movimento na forma matricial usado juntamente ao método de integração numérica de Newmark. Vale ressaltar que outros métodos numéricos também foram implementados no código, permitindo que este seja escolhido no corpo da requisição HTTP.

Modelo de viga

No capítulo anterior foi analisado o comportamento oscilatório para massas pontuais ou discretas, que não leva em consideração as dimensões do corpo, assumindo-se que suas medidas são aproximadamente iguais ao longo dos eixos e suas massas estão concentradas em seu centro de massa. Vale salientar que esta abordagem, por não considerar os parâmetros mencionados, apresenta limitações e os resultados obtidos podem apresentar grande disparidade com os valores reais.

Para contornar a dificuldade citada acima, foi utilizado o método dos elementos finitos, que consiste em dividir o corpo a ser analisado em pequenos elementos com dimensões finitas

Código numérico para solução das análises

Apêndice 1 – Algoritmo base para o método numérico Runge Kutta de Quarta Ordem

using IcVibracoes.Core.DTO.NumericalMethodInput.RigidBody;

using System.Threading.Tasks;

namespace IcVibracoes.Core.NumericalIntegrationMethods.RungeKuttaForthOrder

{

/// <summary>

/// It is responsible to execute the Runge Kutta Forth Order numerical integration

/// method to calculate vibration.

/// </summary>

public abstract class RungeKuttaForthOrderMethod<TInput>

: IRungeKuttaForthOrderMethod<TInput>

where TInput : RigidBodyInput

{

/// <summary>

/// Calculates the value of the differential equation of motion for a specific time,

/// based on the force and angular frequency that are passed.

/// For each case, with one or two degrees of freedom, there is a different

/// differential equation of motion.

/// </summary>

/// <param name="input"></param>

/// <param name="time"></param>

/// <param name="y"></param>

/// <returns></returns>

public abstract Task<double[]> CalculateDifferencialEquationOfMotion(

TInput input,

double time,

double[] y);

/// <summary>

/// Calculates the response of the Runge Kutta Forth Order numerical integration.

/// </summary>

/// <param name="input"></param>

/// <param name="timeStep"></param>

/// <param name="time"></param>

/// <param name="y"></param>

/// <returns></returns>

public async Task<double[]> CalculateResult(

TInput input,

double timeStep,

double time,

double[] y)

{

int arrayLength = y.Length;

double[] result = new double[arrayLength];

double[] t1 = new double[arrayLength];

double[] t2 = new double[arrayLength];

double[] t3 = new double[arrayLength];

double[] y1 = await this.CalculateDifferencialEquationOfMotion(

input,

time,

y)

.ConfigureAwait(false);

for (int i = 0; i < arrayLength; i++)

{

t1[i] = y[i] + 0.5 \* timeStep \* y1[i];

}

double[] y2 = await this.CalculateDifferencialEquationOfMotion(

input,

time + timeStep / 2,

t1)

.ConfigureAwait(false);

for (int i = 0; i < arrayLength; i++)

{

t2[i] = y[i] + 0.5 \* timeStep \* y2[i];

}

double[] y3 = await this.CalculateDifferencialEquationOfMotion(

input,

time + timeStep / 2,

t2)

.ConfigureAwait(false);

for (int i = 0; i < arrayLength; i++)

{

t3[i] = y[i] + timeStep \* y3[i];

}

double[] y4 = await this.CalculateDifferencialEquationOfMotion(

input,

time + timeStep,

t3)

.ConfigureAwait(false);

for (int i = 0; i < arrayLength; i++)

{

result[i] = (y1[i] + 2 \* y2[i] + 2 \* y3[i] + y4[i]) \* (timeStep / 6);

}

return result;

}

}

}