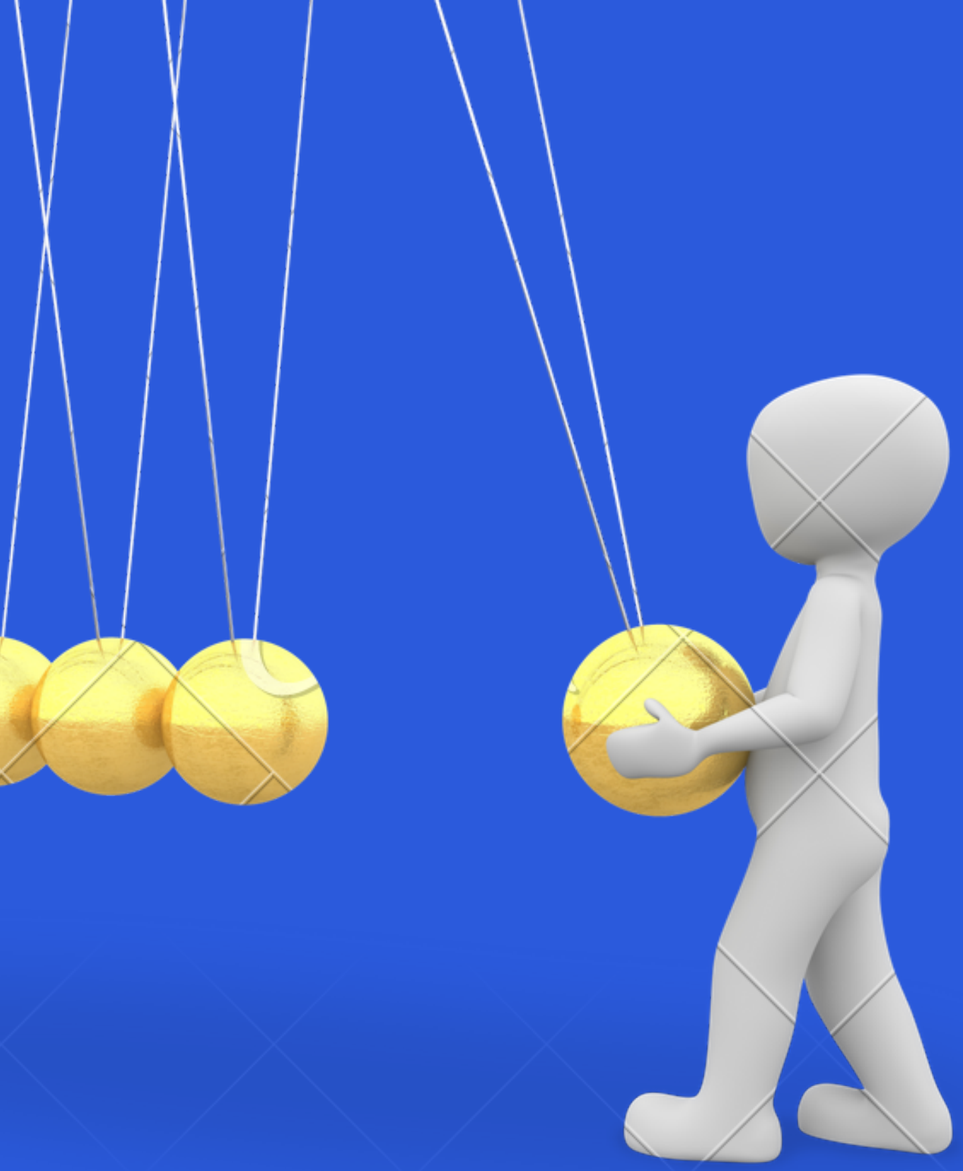
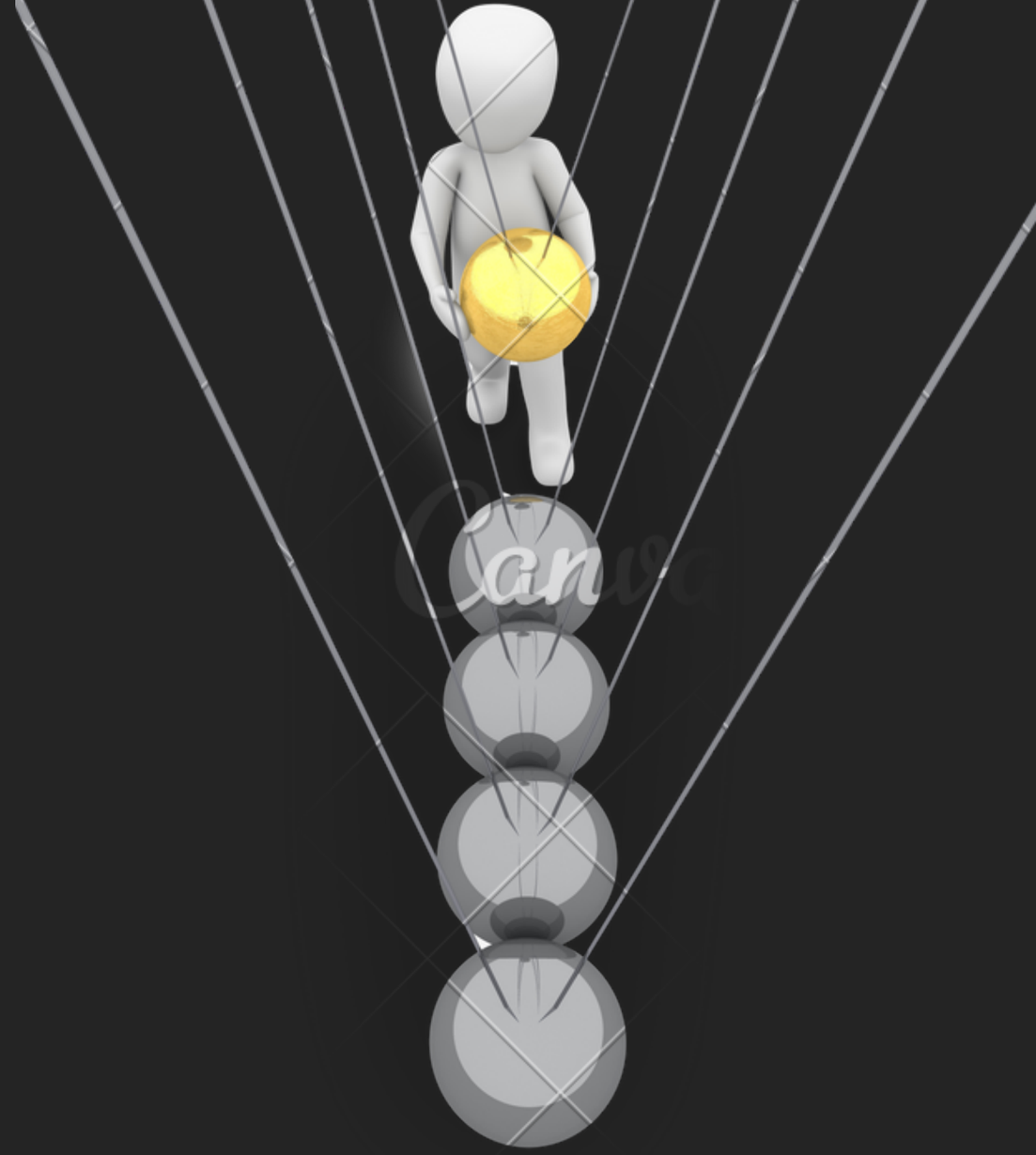


Métodos Numéricos I

Tema 3 - Pêndulos

Integrantes

Caio Cavalcante de Souza Mota
Felipe Menezes Cavalcante
Louis Ian da Silva dos Santos
Adislania Martins Freires
João Vitor Pinheiro Nascimento



Visão Geral

a*d

Amplitude de deslocamento de um pêndulo

d1, d2,..., dn

são deslocamentos do pêndulo

Cd = v

Resolvidos pelo sistema linear

di = detCi/detC

- detC é o determinante da matriz dos coeficientes C
- detCi é o determinante da matriz obtida trocando-se a coluna i pelo vetor d dos termos independentes

Método de Gauss

O método consiste em eliminar os elementos de forma a obter um sistema equivalente com uma matriz triangular superior

Método de Gauss-Jordan

Uma extensão da eliminação de Gauss que zera os elementos acima do pivô, após percorrer a matriz

O código

Método de Gauss

```
void Matriz::gauss()
{
    float novoValor = 0.0f;
    float pivo = 0.0f;
    for (int i = 1; i < this->getLinhas(); i++)
    {
        pivo = this->get(i, i);
        for (int j = i + 1; j <= this->getLinhas(); j++)
        {
            float multiplicador = this->get(j, i) / pivo;
            for (int k = i; k <= this->getColunas(); k++)
            {
                novoValor = this->get(j, k) - multiplicador * this->get(i, k)
                this->set(j, k, novoValor);
            }
        }
    };
}
```

Método de Gauss-Jordan

```
void Matriz::gauss_jordan()
{
    float novoValor = 0.0f;
    float pivo = 0.0f;
    float multiplicador = 0.0f;
    for (int i = 1; i <= this->getLinhas(); i++)
    {
        pivo = this->get(i, i);
        for (int j = 1; j <= this->getLinhas(); j++)
        {
            if (j != i)
            {
                multiplicador = this->get(j, i) / pivo;
                for (int k = i; k <= this->getColunas(); k++)
                {
                    novoValor = this->get(j, k) - multiplicador * this->get(i, k);
                    this->set(j, k, novoValor);
```

```cpp
class Matriz
{
private:
    int linhas, colunas;          // Linhas e colunas da matriz
    vector<vector<float>> matriz; // Vector 2D contendo os dados da
    vector<float> deslocamentos;  // Vector com os valores dos dees

public:
    Matriz(); // Inicializador da matriz
    Matriz(vector<vector<float>>);
    ~Matriz();                    // Destrutor da matriz
    int getLinhas();              // Retorna o numero de linhas da ma
    int getColunas();             // Retirna o numero de colunas da m
    void print();                 // Imprime a matriz
    vector<float> get(int);       // Retorna linha da matriz
    float get(int, int);          // Retorna elemento da matriz
    void set(int, int, float);    // Altera elemento da matriz
    void montarB(vector<float>, float); // Monta a matriz A com o ve
    float determinante();         // Calcula determinante da matriz
    void gauss();                 // Eliminação de Gauss
    void gauss_jordan();          // Eliminação de Gaus-Jordan
    void resolveGauss();          // Resolve por Gauss. presume exist
    void printDeslocamentos();    // Imprime os valores d
    void printAmplitudes();       // Imprime os valores a*d
};
```

```cpp
void Matriz::printDeslocamentos()
{
    cout << endl;      You, 14 hours ago • lógica de sa
    cout << "Deslocamentos: ";
    for(int i = 0; i < colunas - 1; i++)
    {
        cout << this->deslocamentos[i] << " ";
    }
    cout << endl;
}

void Matriz::printAmplitudes()
{
    float a = deslocamentos[colunas-1];

    cout << endl;
    cout << "Amplitudes de deslocamento: ";
    for(int i = 0; i < colunas - 1; i++)
    {
        cout << a*deslocamentos[i] << "   ";
    }
    cout << endl;
}
```

```cpp
void Matriz::print()
{
    // Imprime a matriz
    for (int i = 0; i < this->getLinhas(); i++)
    {
        for (int j = 0; j < this->getColunas(); j++)
        {
            cout.width(12);
            cout << this->get(i + 1, j + 1) << " ";
        }
        cout << endl;
    }

    // Imprime os deslocamentos
    for (int i = 0; i < this->deslocamentos.size(); i++)
    {
        cout.width(12);
        cout << this->deslocamentos[i]  << " ";
    }
    cout << endl << endl;
}
```

```cpp
void Matriz::resolveGauss()
{
    this->gauss();
    int ultimaColuna = this->colunas - 1;
    for (int i = this->linhas - 1; i >= 0; i--)
    {
        this->deslocamentos[i] = this->matriz[i][ultimaColuna];
        for (int j = this->linhas - 1; j > i; j--)
        {
            this->deslocamentos[i] -= this->deslocamentos[j] * this->matriz[i][j];
        }
        this->deslocamentos[i] /= this->matriz[i][i];
    }
}
```

Conclusões

CONCLUSÕES

01

O método de Gauss-Jordan é mais custoso computacionalmente na fase de eliminação

02

O método de Gauss é mais custoso computacionalmente na fase de solução da matriz triangular superior

03

Solucionamos a matriz de coeficientes C ao fim de Gauss-Jordan utilizando o mesmo método de solução de matriz triangular superior de Gauss, por ser equivalente

04

O método de Gauss-Jordan pode ser otimizado por meio de método, que, ao invés de resolver uma matriz T superior, apenas faria o calcule
di = (Vi / Cii)

OBRIGADO!

Dúvidas?