Universidade Federal do Rio de Janeiro

Departamento de Matemática Aplicada
Instituto de Matemática

Metódos Matemáticos em Mecânica Geométrica

Gil Sales Miranda Neto

Orientador Prof. Dr. Alejandro Cabrera
Instituto de Matemática
Departamento de Matemática Aplicada
Universidade Federal do Rio de Janeiro

01 de Setembro de 2019

Contents

1 Definindo os metódos para o algoritmo

1.1 Dados de Input

- $\vec{r}_{i0}(t) \in \mathbb{R}^3$, $i = 1, \dots, N$
- $m_1, \dots, m_N,$, massas do sistema
- $\vec{J}_0$, matriz do momento angular inicial

1.2 Dados de Output

$$\vec{r}_i(t) = R(t)\vec{r}_{i0}(t),\ i = 1, \dots, N$$

Onde $\vec{r}_i(t)$ é a posição a partir de um sistema inercial, com origem em $CM(t)$, centro de massa.
$R(t) \in SO(3)$ é a matriz de rotação e também a incógnita do problema.

1.3 Sistema de Equações Diferenciais

Obtemos $R(t)$ a partir de um sistema de equações diferenciais que vem da conservação do momento angular total, quando visto de um referencial inercial.

Incógnitas do sistema

- $R(t) \in SO(3)$, matriz de rotação
- $\vec{\pi}(t) \in \mathbb{R}^3$

O sistema

$$\begin{cases} R^{-1}(t)\dot{R}(t) & = \psi^{-1}(I_0^{-1}(t)(\vec{\pi} - \vec{L}_0(t)) \\ \dot{\vec{\pi}} & = \vec{\pi} \times (I_0^{-1}(t)(\vec{\pi} - \vec{L}_0(t)) \end{cases}$$

Condições iniciais da equação diferencial: $\begin{cases} R(t_0) = \mathbb{1} \\ \vec{\pi}(t_0) = \vec{J}_0 \text{ (momento angular inicial)} \end{cases}$

Onde ψ é uma matriz $3x3$ antissimétrica, ou seja $\psi^T = -\psi$

$I_0(t) \in Mat_{3x3}(\mathbb{R})$ matriz simética, tensor de inércia.

Função Tensor de Inércia

$F_I : (\vec{r}_1, \ldots, \vec{r}_N) \mapsto I(\vec{r}_1, \ldots, \vec{r}_N)$, é a função 'tensor de inércia' de uma configuração qualquer, a qual depende das massas.

Então $I_0(t) = F_I(\vec{r}_{01}(t), \ldots, \vec{r}_{0N}(t))$

Função Momento Angular

$F_L : (\vec{r}_1, \ldots, \vec{r}_N, \vec{v}_1, \ldots, \vec{v}_N) \mapsto L(\vec{r}_1, \ldots, \vec{r}_N, \vec{v}_1, \ldots, \vec{v}_N)$, é a função 'momento angular', a qual depende das massas.

Então: $\vec{L}_0(t) = F_L(\vec{r}_{01}(t), \ldots, \vec{r}_{0N}(t), \dot{\vec{r}}_{01}(t), \ldots, \dot{\vec{r}}_{0N}(t))$

O algoritmo

2

2.1 Dependência de pacotes

O algoritmo é compatível com a linguagem Python na versão 3.x ou superior, e utiliza os seguintes pacotes

- NumPy: `https://github.com/numpy/numpy` - para tratamentos de cálculos matemáticos e utilização de vetores e matrizes
- NumPy-Quaternion: `https://github.com/moble/quaternion` - para lidar com operações de quaternions, a biblioteca é escrita em C++ e portada para Python para melhor velocidade
- SciPy: `https://github.com/scipy/scipy`
- MatPlotLib: `https://github.com/matplotlib/matplotlib` - para lidar com plotagem dos gráficos 2D/3D e animações

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import quaternion as q
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
from matplotlib.animation import FuncAnimation
from mpl_toolkits.mplot3d.art3d import juggle_axes
```

Listing 2.1: Dependencies import