UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CAMPUS MOSSORÓ
CENTRO DE ENGENHARIAS
CURSO BACHARELADO EM ENGENHARIA MECÂNICA

LUCAS LEONARDO LOPES SILVA

ANÁLISE CINEMÁTICA DE UMA PLATAFORMA DE STEWART POR MEIO DA MODELAGEM DE DINÂMICA DE MULTICORPOS

MOSSORÓ - RN

2019

LUCAS LEONARDO LOPES SILVA

ANÁLISE CINEMÁTICA DE UMA PLATAFORMA DE STEWART POR MEIO DA MODELAGEM DE DINÂMICA DE MULTICORPOS

Projeto de conclusão de curso apresentado a Universidade Federal Rural do Semi-Árido – UFERSA, Campus Mossoró, para obtenção do título Bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Alex Sandro de Araujo Silva

MOSSORÓ - RN

2019

S586a Silva, Lucas Leonardo Lopes.

Análise cinemática de uma plataforma de Stewart por meio da modelagem de dinâmica de multicorpos / Lucas Leonardo Lopes Silva. - 2019.

77 f. : il.

Orientador: Alex Sandro de Araujo Silva.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Engenharia Mecânica, 2019.

1. Análise Cinemática. 2. Plataforma Stewart. 3. Dinâmica Multicorpos. I. Silva, Alex Sandro de Araujo, orient. II. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas

da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

LUCAS LEONARDO LOPES SILVA

ANÁLISE CINEMÁTICA DE UMA PLATAFORMA DE STEWART POR MEIO DA MODELAGEM DE DINÂMICA DE MULTICORPOS

Monografia apresentada a Universidade Federal Rural do Semi-Árido – UFERSA, Campus Mossoró, para obtenção do título Bacharel em Engenharia Mecânica.

APROVADA EM: 18 / 03 / 2019

BANCA EXAMINADORA

Profº. Dr. Alex Sandro de Araujo Silva – UFERSA
Presidente

Profº. Dr. Fabricio Jose Nobrega Cavalcante
Primeiro Membro

Profº. Dr. Zoroastro Torres Vilar
Segundo Membro

A João Silva Neto, meu pai, que me ajudou, trabalhou e esteve ao meu lado durante este curso

A Lucia Maria Lopes Silva, minha mãe, que junto ao meu pai, me deu apoio e força para superar as dificuldades e sempre se preocupou comigo.

A Deus, por me conceber saúde e coragem para enfrentar a rotina diária.

AGRADECIMENTOS

Aos meus pais, João Silva Neto e Lucia Maria Lopes Silva, que se esforçam tanto para dar o melhor que podem, que me apoiam no que faço e contribuem da forma que podem para que eu possa construir meu futuro, deixando apenas o que eles podem me oferecer: a oportunidade de estudar;

A minha namorada, Sarah Sunamyta da Silva Gouveia, que desde de C&T, esteve sempre ao meu lado auxiliando nos momentos difíceis, compartilhando de todo conhecimento obtido, carinho, amor, dedicação e sempre me fazendo sorrir nos momentos bons e ruins;

Ao meu orientador Alex Sandro de Araujo, que me ajudou nesse período de orientação, fornecendo os conhecimentos, não só na orientação, mas também nas matérias que pude pagar com ele, tendo sempre paciência para retirar dúvidas, fazendo com que novos conhecimentos fossem absorvidos;

Aos professores que puderam participar da banca, Fabricio Jose Nobrega Cavalcante e Zoroastro Torres Vilar, contribuindo assim para a realização deste trabalho;

Ao meu amigo André Henrique da Silva Brandão (Coruja), que enfrentou muitas dificuldades comigo, durante este curso;

A todos que conheci durante todos os períodos e que de alguma forma contribuíram para esse fim;

A todos os meus familiares, tios, primos e avos, que me apoiaram.

RESUMO

O presente estudo, trata-se de uma análise cinemática, processo que visa estudar os movimentos de um determinado mecanismo, visando obter dados sobre suas posições, velocidades e acelerações em um determinado instante. Isso foi realizado por meio de uma revisão bibliográfica de artigos, dissertações, livros, monografias, teses e outros meios de informação, afim de obter conteúdo, dados e informações que foram julgadas necessárias e suficientes, para o entendimento de um mecanismo especifico, plataforma de Stewart. A plataforma de Stewart é um mecanismo paralelo, composto por duas bases e seis braços, sendo muito usado em simuladores de movimentos. O estudo dos comportamentos de posição, velocidades e acelerações da mesma, foi implementado com o auxílio de um software de manipulação numérica, MATLAB. Logo, após uma breve apresentação de conceitos introdutórios sobre mecanismos, análise cinemática e o software MATLAB, foi explicado os procedimentos tomados para que a realização da análise fosse possível. Portanto, com os resultados obtidos, pode-se dizer que o projeto selecionado da plataforma de Stewart obteve um funcionamento correto, realizando os movimentos propostos sem erros significativos, de acordo com a precisão usada para os cálculos, e obtendo um conjunto de dados satisfatório para futuras análises e implementações.

Palavras-chave: Plataforma de Stewart. Cinemática. MATLAB.

ABSTRACT

This present study is a kinematic analysis, process that aims to study the movements of a specific mechanism, with purpose of collect data of yours positions, velocities and accelerations in an instant. This was done by means of a bibliography review of articles, dissertations, books, monographs, theses and others materials, in order to get information and data that may be judged needed and sufficient, for the study of specify mechanism, Stewart platform. The Stewart platforms is a parallel mechanism compost for two bases and six arms, with multiplies applications in motion simulators. The realization of kinematic analysis for the study of actions of positions, velocities and accelerations of same, this method has been implemented with a helper software, MATLAB. Soon, after a quickly show of concepts, that include knowledge about mechanisms kinematic analysis and the used software MATLAB, will be explained all the steps for the realization of kinematics analysis. Therefore, with obtained results, the selected project of Stewart platform it was successful, making the proposed moves without significantly error and obtain a satisfactory data set for future analysis and implementations.

Keywords: Stewart platform. Kinematic. MATLAB

LISTA DE FIGURAS

Figura 1 - Mecanismo de Gough 16
Figura 2 - Plataforma de Stewart 17
Figura 3 - Maquina Universal de Teste de Pneus 18
Figura 4 - Plataforma com 6 Graus liberdade 18
Figura 5 - Mecanismo de quatro barras 20
Figura 6 - Principais tipos de juntas 21
Figura 7 - Sistema de coordenadas local em um corpo 21
Figura 8 - Sistemas de referência com origens não coincidentes 22
Figura 9 - Exemplo para obtenção de restrições absolutas 24
Figura 10 - Exemplo para obtenção de restrições relativas 24
Figura 11 - Exemplo para obtenção das restrições motoras relativas 26
Figura 12 - Representação Gráfica do Método de Newton-Raphson 37
Figura 13 - Plataforma De Stewart utilizada 39
Figura 14 – Numeração dos corpos 41

LISTA DE FLUXOGRAMAS

Fluxograma 1 - Exemplo de operador SE 34
Fluxograma 2 - Exemplo do operador ELSE 35
Fluxograma 3 - Exemplo do operador IFELSE 35
Fluxograma 4 - Procedimento para análise Cinemática 38

LISTA DE GRÁFICOS

Gráfico 1 - Análise de Posição 48
Gráfico 2 - Comportamento das variáveis X14 e Y14 49
Gráfico 3 - Orientação do corpo 14 50
Gráfico 4 - Comportamento das variáveis e54 e e55 51
Gráfico 5 - Variação de Velocidade linear do corpo 14 52
Gráfico 6 - Aceleração Linear Corpo 14 53
Gráfico 7 - Comportamento da Função Motora 54

LISTA DE QUADROS

Quadro 1 - Resumo das Principais Janelas do MATLAB 32
Quadro 2 - Distâncias entre as origens dos sistemas de coordenadas global e local 42
Quadro 3 - Parâmetros de Euler Iniciais 42
Quadro 4- Vetores de posição local para cada junta 43
Quadro 5 - Relações vetoriais das principais restrições 43
Quadro 6 - Modelagem das restrições 44
Quadro 7 - Jacobiano das restrições 45

SUMÁRIO

1. INTRODUÇÃO 14
2. OBJETIVOS 15
2.1. Objetivo Geral 15
2.2. Objetivos Específicos 15
3. REFERÊNCIAL TEÓRICO 16
3.1. Plataforma de Stewart 16
3.2. Dinâmica de Multicorpos 19
3.2.1. Conceitos Introdutórios 19
3.2.2. Análise Cinemática 23
3.2.3. Parâmetros de Euler 30
3.3. Matlab 32
3.3.1. Conceitos básicos 32
3.3.2. Procedimento Numérico para análise cinemática 36
4. MATERIAIS E MÉTODOS 39
4.1. Modelo da plataforma 39
5. RESULTADOS E DISCUSSÕES 42
5.1. Analise Cinemática 43
5.2. Discussões sobre os resultados da análise cinemática 47
6. CONSIDERAÇÕES FINAIS 56
REFERÊNCIAS 57
APÊNDICE A – Códigos e funções para a análise cinemática 60
Código Principal 60
Análise de Posição 61
Restrições 62
Armazenamento dos dados da plataforma 64
Modelagem dos Parâmetros de Euler 67
Funções das restrições 68
Matriz Jacobiana 68
Funções dos Jacobianos das restrições 71
Análise de velocidade 72
Função Nu 73
Funções De Auxilio 73
Análise de Aceleração 75
Função Gama 75
Funções Gama das Principais Restrições 76

1. INTRODUÇÃO

Uma plataforma de Stewart, é um mecanismo que pode assumir diversas configurações, sua configuração geral está baseada em duas bases interligadas por meio de atuadores. Esse mecanismo é usado em diversas aplicações como movimentação de cargas e posicionamento de objetos e outros mecanismos e máquinas (SMIT, 2000).

O projeto de uma plataforma ou de qualquer outro mecanismo, requer um estudo avançado sobre dinâmica de multicorpos, desde dos princípios básicos como leis de Newton, à assuntos mais complexos como resolução de equações algébricas não lineares.

De acordo com Haug (1989), a modelagem e resolução das equações de restrição de um dado mecanismo é uma tarefa complexa, visto o alto número iterações necessárias para a representação correta do mecanismo, logo procedimentos numéricos com o auxílio de computadores é recomendável para tal tarefa.

Como dito por Dasgupta e Mruthyunjaya (1998), o elevado número de corpos presentes em uma plataforma de Stewart padrão, pode ocasionar uma carga matemática muito elevada, dificultando a realização das análises previstas, onde um dos principais motivos é a singularidade.

Portanto, diversos softwares podem ser usados para esses cálculos, um que merece destaque é o MATLAB, que segundo Gilat (2004) pode realizar diversas tarefas com matrizes e vetores, desde de modelagens, cálculos, análises gráficas e execução de algoritmos.

Além da modelagem computacional, a construção das equações de restrições e cálculo das funções presentes na análise cinemática são os processos mais importantes para a realização de uma análise sem erros, visto que um simples sinal invertido é motivo para a obtenção de um jacobiano singular.

2. OBJETIVOS

2.1. Objetivo Geral

Analisar uma plataforma de Stewart usando a abordagem de Dinâmica de Multicorpos com o auxílio do método numérico de Newton Raphson para a solução de equações de restrições.

2.2. Objetivos Específicos

- Estudo das características do mecanismo plataforma de Stewart presentes na literatura técnica e especializada;
- Estudo da técnica de modelagem de Dinâmica de Multicorpos;
- Modelagem da análise cinemática;
- Implementação computacional da análise cinemática usando o software MATLAB;

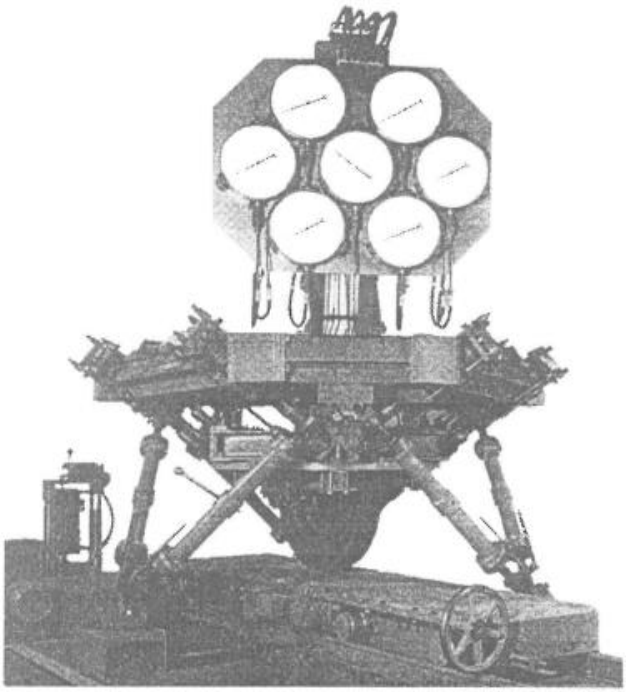
3. REFERÊNCIAL TEÓRICO

3.1. Plataforma de Stewart

De acordo com Hay (1999) protótipos de simuladores de pilotagem aérea começaram a surgir por volta de 1965, propostos por Stewart como uma plataforma com seis graus de liberdade, a partir disso inúmeros designs baseados nessa plataforma foram surgindo, dando origem a diversos mecanismos para as mais diversas aplicações.

De acordo com Bingul e Karah (2012) a plataforma proposta por Stewart, foi baseada no mecanismo desenvolvido por Gough, havendo modificações na sua geometria, tanto é que alguns autores utilizam a nomenclatura plataforma de Stewart-Gough ou plataforma de Stewart generalizada, o dispositivo que foi utilizado como base é demonstrado na Figura 1.

Figura 1 - Mecanismo de Gough



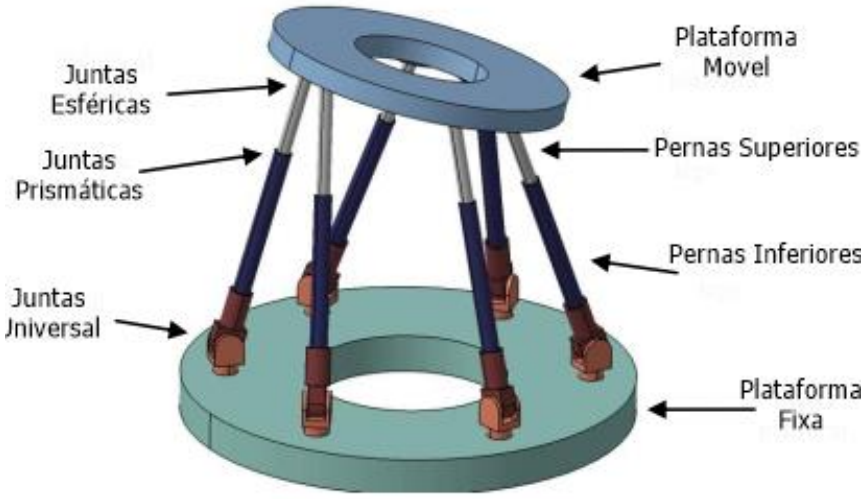
Fonte: Hay (1999)

Du Plessis (2001) comenta a diferença entre os mecanismos robóticos, mostrando as diferenças entre os mecanismos em serie e paralelos, embora os mecanismos em serie sejam mais preferidos industrialmente para substituir atividades humanas, os mesmos possuem uma baixa capacidade de carga e uma fraca performance dinâmica, em outras palavras não tem uma boa precisão de posicionamento. Dessa forma os mecanismos em paralelo têm uma certa superioridade em determinadas tarefas, concluindo o autor descreve que a plataforma de Stewart se encaixa como um mecanismo paralelo.

Segundo Smit (2000) a plataforma de Stewart é um mecanismo robótico paralelo, que obedece a uma composição genérica de: duas bases, uma superior (móvel) e uma inferior (fixa), onde essas bases são interligadas por seis braços/pernas mecânicas, conhecidos como atuadores lineares, esses que possuem pernas superiores e inferiores, interligadas por juntas prismáticas. Esses atuadores lineares, estão ligados a plataforma fixa por meio de juntas esféricas, que proporcionam movimento nas três dimensões.

Hay (1999) complementa que são os atuadores lineares que determinam a posição e orientação da plataforma, pois a medida com que eles contraem ou estendem, a plataforma modifica sua localização, uma exemplificação de uma plataforma de Stewart pode ser vista na Figura 2.

Figura 2 - Plataforma de Stewart



Fonte: Damic e Cohodar (2014)

Smit (2000) demonstra na sua pesquisa, que as vantagens dos mecanismos robóticos paralelos sobre os que atuam em série, garantem uma vasta área de aplicações para os mesmos. Vantagens como melhor distribuição de carga sobre os atuadores lineares, melhor rigidez da plataforma como um todo, e um uso mais justificado para os atuadores lineares, devido as cargas de tração e compressão. Dessa forma, as variações geométricas da plataforma de Stewart garantem uma aplicabilidade em: simuladores de aviação, dispositivos com alta precisão de posicionamento, como mesas cirúrgicas, mecanismos que trabalharão sobre condições de movimento constante e controladores que necessitam de seis graus de liberdade para seu movimento. Alguns exemplos de mecanismos podem ser vistos nas Figuras 3 e 4.

Figura 3 - Maquina Universal de Teste de Pneus

Fonte: Mikrolar (2017)

Figura 4 - Plataforma com 6 Graus liberdade

Fonte: Technologies (2017)

Um problema citado por Smit (2000) é a singularidade (posição e orientação da plataforma onde não há um equilíbrio das forças externas, levando ao colapso) causada pelo limitado espaço de trabalho. Logo Dasgupta e Mruthyunjaya (1998) mostram que as limitações de movimento ficam restritas apenas pelos limites de movimento das juntas, porém haverão configurações onde os atuadores e juntas perderam alguns graus de liberdade, vindo a perder o controle.

Segundo Bingul e Karah (2012), esses problemas geram uma dificuldade muito grande nas análises cinemática e dinâmica dessas plataformas, pois para o design e controle dos atuadores da plataforma é necessária uma alta precisão do modelo dinâmico, isto se torna complicado de obter devido ao arranjo fechado da estrutura, da relação intrínseca existente entre os parâmetros do sistema e da alta não-linearidade das equações de restrições cinemáticas e dinâmicas do sistema.

3.2. Dinâmica de Multicorpos

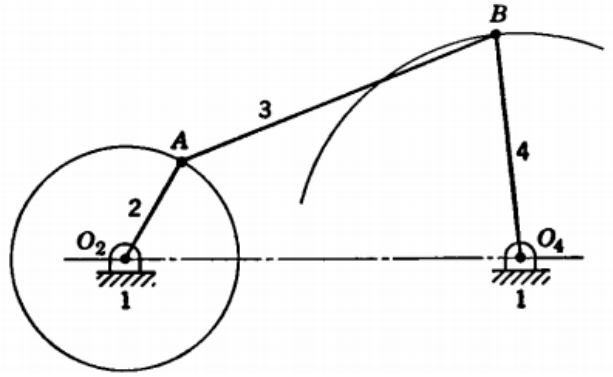
3.2.1. Conceitos Introdutórios

Sistemas de multicorpos são sistemas mecânicos, que possuem uma certa quantidade de corpos, interligados de maneira que haja um movimento relativo entre os corpos, essas conexões podem ser feitas por meio de molas, amortecedores e juntas (WIJCKMANS,1996).

O campo da dinâmica de multicorpos se baseia em dois conceitos principais, o da cinemática e o da cinética, onde de acordo com Hibbeler (2011) a cinemática não envolve as forças presentes no sistema, e sim apenas os aspectos geométricos do movimento, já a cinética fica responsável pela a análise das forças envolvidas no movimento. De acordo com Norton (2009), os conceitos de cinemática e cinética são de fundamental importância para a análise de um mecanismo.

Mabie (1980) define mecanismo como um meio de transmitir e controlar um movimento, podendo ser dividido em diversos tipos de acordo com os seus componentes. Um dos que apresenta configuração menos complexa e é bem importante em relação a quantidade de tarefas que pode realizar, é o mecanismo de quatro barras ou quadrilátero articulado, como é apresentado na Figura 5.

Figura 5 - Mecanismo de quatro barras



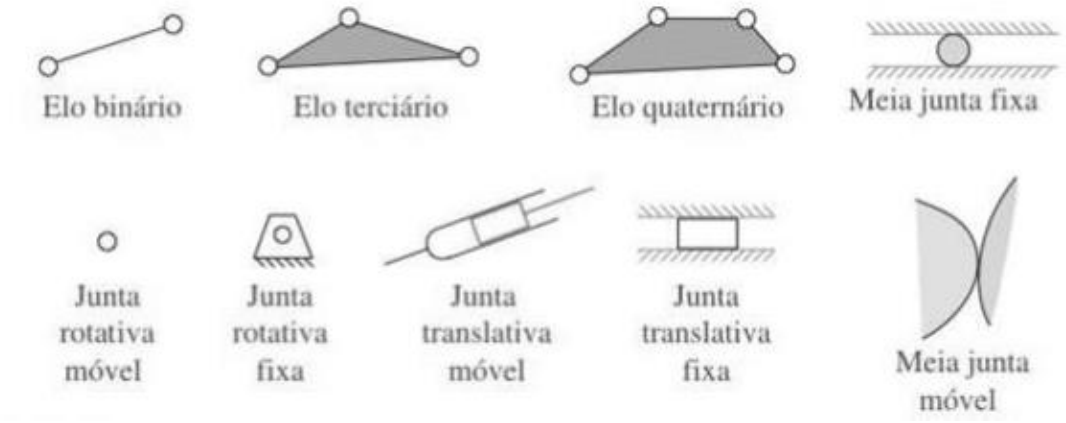
Fonte: Mabie (1980)

Complementando o raciocínio, Meriam e Kraige (2006) informam que um mecanismo nos procedimentos de análise pode ser considerado um como corpo rígido, composto de partículas, onde as mesmas não sofrerão variação de movimento e que a deformação sofrida, devido as forças atuantes no movimento, por essas partículas podem ser consideradas insignificantes em comparação com todo o corpo.

De acordo com Halliday, Resnick e Walker (2008) os tipos de movimento que um mecanismo pode realizar são de grande importância para que se possa determinar o número de graus de liberdade e tipos de juntas que um mecanismo possui. Dessa forma um mecanismo ou um corpo ligado a ele pode realizar rotação pura, onde um ponto do corpo não apresenta deslocamento em relação ao restante do corpo, translação, onde todos os pontos do corpo descrevem trajetórias paralelas e por último, movimento complexo, onde há a existência de rotação e translação em um único movimento.

Norton (2009) descreve alguns conceitos ligados aos tipos de movimentos que um mecanismo pode realizar, dentre eles o número de graus de liberdade, onde este é definido como o número de variáveis necessárias para descrever uma única posição no espaço em qualquer instante. Desta forma é definido que uma junta é a união entre dois ou mais elos de corpos diferentes. A Figura 6 descreve os principais tipos de juntas.

Figura 6 - Principais tipos de juntas

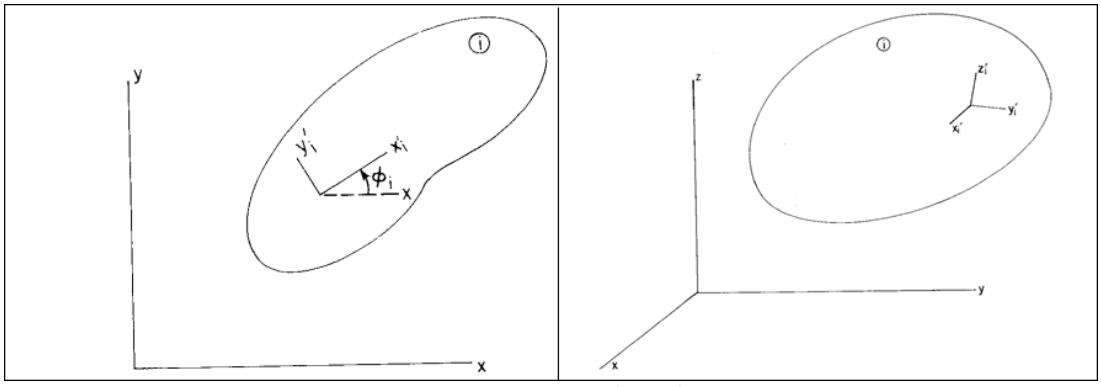


Fonte: Norton (2012)

Para a análise cinemática de um mecanismo, Wijckmans (1996) afirma que quaisquer conjuntos de variáveis que especificam a posição e orientação de todos os corpos do mecanismo é denominado de coordenadas generalizadas, onde estas podem ser cinemáticas ou motoras.

Nikravesh (1988) segue o raciocino mostrando que, para o procedimento de obtenção das equações de restrições, é fixado um sistema de coordenas local nos corpos a serem analisados, onde as coordenadas desse plano de referência local podem ser escritas em função das coordenadas do plano de referência global, originando equações de coordenadas generalizadas para cada corpo estudado na análise. A Figura 7 pode exemplificar o que foi dito tanto para um sistema planar como espacial.

Figura 7 - Sistema de coordenadas local em um corpo



Fonte: Haug (1989)

Dada a necessidade do uso de dois ou mais planos de coordenadas Josephs e Huston (2002) abordam em sua bibliografia os procedimentos matemáticos para a transformação de coordenadas, de um sistema de referência para outro.

Nikravesh (1988) apresenta que os vetores que descrevem as posições nos dois planos podem ser relacionados por uma matriz de rotação denominada por A, que pode variar em um sistema espacial de acordo com a ordem das rotações realizadas. Conforme a Equação 1.

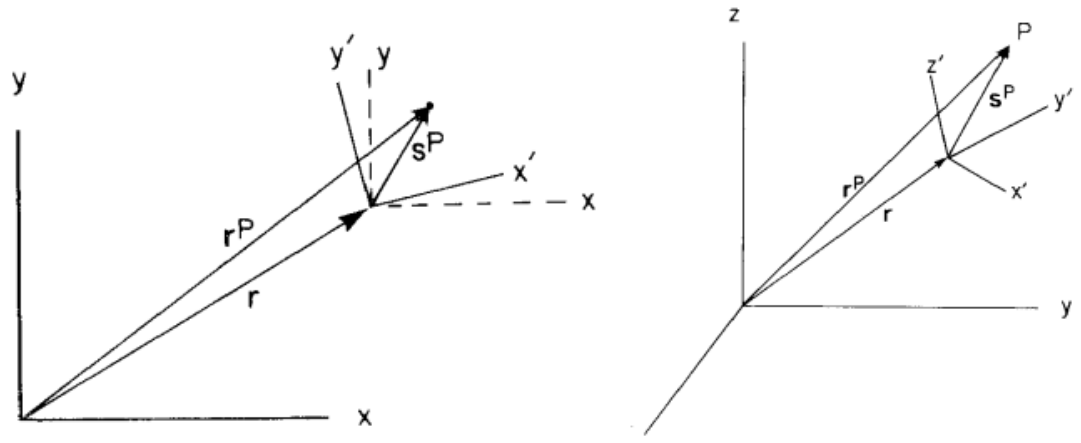
$$A = \begin{bmatrix} \cos\phi & -\operatorname{sen}\phi \\ \operatorname{sen}\phi & \cos\phi \end{bmatrix} ou \begin{bmatrix} \cos\phi & -sen\,\phi & 0 \\ sen\,\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Ou pela Equação 2:

$$s = As' \quad (2)$$

Outra condição bastante comum abordada por Haug (1989) é quando as origens dos sistemas de coordenadas não estão sobrepostas. Nessa condição, é necessário o uso de um vetor auxiliar, para descrever a posição correta dos planos de referência, geralmente o local. De maneira mais clara a Figura 8 ilustra uma situação onde os planos de referência não possuem as origens coincidentes para os dois sistemas 2D e 3D.

Figura 8 - Sistemas de referência com origens não coincidentes



Fonte: Haug (1989)

Como é observado o vetor $\vec{r}$ liga as duas origens dos planos de referência, o vetor $\overrightarrow{S^P}$ indica a distância da origem do sistema de coordenadas local a um ponto desejado e o vetor $\overrightarrow{r^P}$ é o vetor resultante da soma vetorial de $\vec{r} + \overrightarrow{S^P}$.

3.2.2. Análise Cinemática

De acordo com Bauchau (2011) as restrições cinemáticas são independentes do tempo e as restrições motoras são equações que dependem do tempo, sendo estas representadas das seguintes formas pelas Equações 3 e 4.

$$\Phi^k(q) = 0 \tag{3}$$

$$\Phi^D(q,t) = 0 \tag{4}$$

O termo (q) nas Equações 3 e 4 representam um conjunto de coordenadas generalizadas julgadas necessárias e suficientes para informar a posição do corpo. Para um corpo em um plano, três coordenadas são necessárias para mostrar a sua posição, enquanto para um corpo no espaço, seis coordenadas generalizadas são necessárias. O termo (t) na Equação 4, representa que as restrições motoras dependem do tempo. Assim as equações de restrições de um mecanismo podem ser reescritas como na Equação 5.

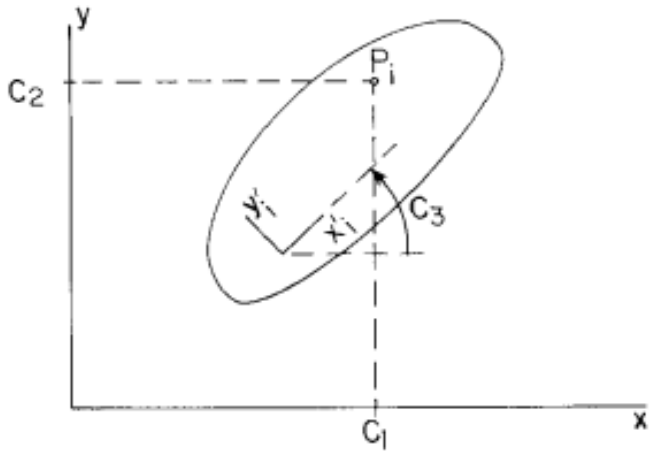
$$\Phi(q,t) = \begin{bmatrix} \Phi^k(q) \\ \Phi^D(q,t) \end{bmatrix} = 0 \tag{5}$$

Wijckmans (1996) descreve que as restrições cinemáticas e motoras de um sistema podem ser classificadas em função das coordenadas de duas formas, absolutas e relativas. Quando as restrições são escritas em função das coordenadas absolutas significa dizer que as restrições são escritas em função do sistema de referência global ou também conhecido por restrições entre o corpo e solo, porém quando as restrições são escritas em função das coordenadas relativas significa que as restrições serão descritas em função do sistema de referência local de cada corpo ou entre pares de corpos.

Segundo Haug (1989) quando uma análise de restrições cinemáticas é realizada em um único corpo, as equações de restrições podem ser expressas

em função das coordenadas generalizadas do mesmo. Dessa maneira para um corpo descrito como o da Figura 9, têm-se as seguintes restrições absolutas.

Figura 9 - Exemplo para obtenção de restrições absolutas



Fonte: Haug (1989)

$$\Phi^{ax(i)} = x_i^P - C_1 = x_i + x_i'^P \cos\phi_i - y_i'^P \operatorname{sen}\phi_i - C_1 \quad (6)$$

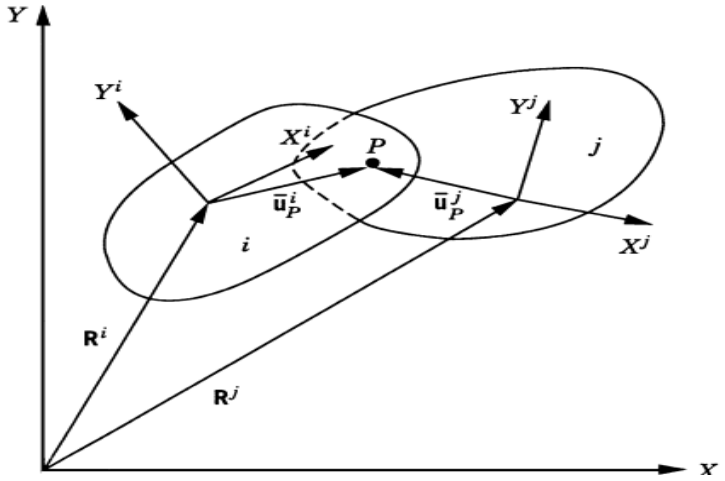
$$\Phi^{ay(i)} = y_i^P - C_2 = y_i + x_i'^P \operatorname{sen}\phi_i - y_i'^P \cos\phi_i - C_2 \quad (7)$$

$$\Phi^{a\phi(i)} = \phi_i^P - C_3 = 0 \quad (8)$$

Dessa forma as equações de restrições cinemáticas para o corpo mostrado na Figura 9, são apresentadas pelas Equações 6 e 7, responsáveis pelas restrições de translação do corpo, e a Equação 8, responsável pela restrição de rotação do corpo. Onde nas Equações 6, 7 e 8, $C1, C2\ e\ C3$ são constantes, $q_i = [x_I, y_I\ ]^T$ e $q_i'^P = [x_i'^P, y_i'^P\ ]^T$ são as coordenadas no plano de referência global e local, respectivamente.

Enquanto isso Shabana (2001) demonstra uma análise de restrições cinemáticas relativas em um sistema planar, ou seja, dois corpos estão presentes, unidos por uma junta mecânica representada pelo ponto P, como é visto na Figura 10.

Figura 10 - Exemplo para obtenção de restrições relativas



Fonte: Shabana (2001)

Nesse caso têm-se que o ponto P é representado no sistema de coordenadas local de cada corpo pelo vetor $\bar{u}_P^i$ para o corpo i e $\bar{u}_P^j$ para o ponto j. Por fim, a equação de restrição entre esses dois corpos poderia ser escrita a partir da diferença dos vetores posição do ponto P, em relação a cada corpo, no sistema de coordenadas global, como na Equação 9:

$$r_P^i - r_P^j = 0 \tag{9}$$

Portanto, a diferença entre os vetores posição da junta P para cada corpo pode ser reescrita como apresentado na Equação 10:

$$R^i + A^i \bar{u}_P^i - R^j - A^j \bar{u}_P^j = 0 \tag{10}$$

Por fim, os termos A^i e A^j representam as matrizes de rotação para os corpos i e j.

De maneira análoga ao conceito das restrições cinemáticas, absolutas e relativas, a teoria apresentada por Nikravesh (1988) sobre restrições motoras, absolutas e relativas, difere apenas que as mesmas são dependentes do tempo. Dessa forma, as restrições motoras absolutas para um corpo semelhante ao da Figura 9, ficariam expressas da seguinte forma apresentada nas Equações 11, 12 e 13:

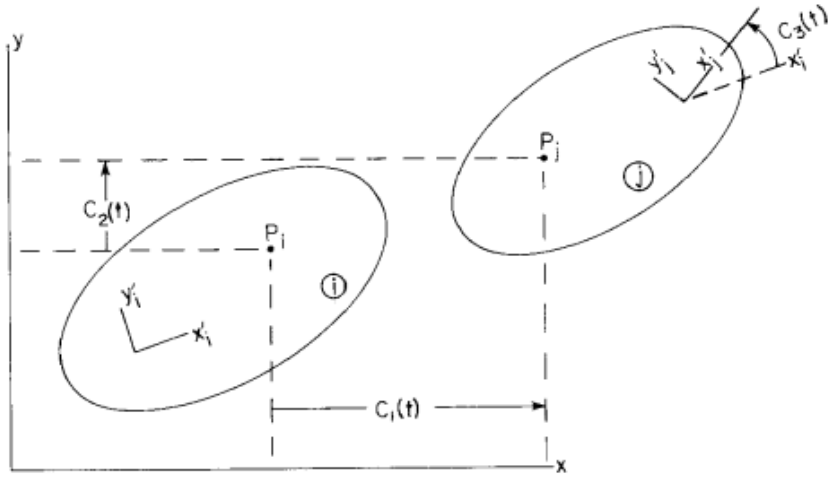
$$\Phi^{axd(i)} = x_i^P - C_1(t) = 0 \tag{11}$$

$$\Phi^{ayd(i)} = y_i^P - C_2(t) = 0 \tag{12}$$

$$\Phi^{a\phi d(i)} = \phi_i - C_3(t) = 0 \tag{13}$$

E de forma semelhante as restrições cinemáticas relativas, as restrições motoras relativas podem ser expressas com o mesmo raciocínio, no que difere apenas que o sistema apresentado não possui os corpos em contato direto por uma junta, e sim por um atuador mecânico, sendo necessária a inclusão de uma restrição motora para essa distância dos dois pontos, para o corpo apresentado na Figura 11, as restrições motoras podem ser expressas da seguinte forma, conforme as Equações 14 a 17:

Figura 11 - Exemplo para obtenção das restrições motoras relativas



Fonte: Haug (1989)

$$\Phi^{rxd(i,j)} = x_j^P - x_i^P - C_1(t) = 0 \quad (14)$$

$$\Phi^{ryd(i,j)} = y_j^P - y_i^P - C_2(t) = 0 \quad (15)$$

$$\Phi^{r\phi d(i,j)} = \phi_j - \phi_i - C_3(t) = 0 \quad (16)$$

$$\Phi^{rrd(i,j)} = \left(x_j^P - x_i^P\right)^2 - \left(y_j^P - y_i^P\right)^2 - (C_4(t))^2 = 0 \quad (17)$$

Para sistemas espaciais, os conceitos apresentados das equações 06 a 17, podem ser replicados apenas com o acréscimo de um terceiro eixo de coordenadas e mais duas variáveis de orientação.

Um outro parâmetro de vital importância para a análise cinemática é o Jacobiano ou matriz jacobiana, que adaptando o conceito matemático de Flemming e Gonçalves (1999), é definida como uma matriz quadrada formada a partir das derivações das equações de restrição pelo o vetor de coordenadas generalizadas presentes no sistema analisado.

Sabendo identificar todas as juntas físicas, movimentos e equações de restrições cinemáticas, Negrut (2014) mostra que para realizar a análise de posição de um mecanismo, ou seja, obter suas posições e orientações em um determinado instante de tempo, é necessário resolver as equações de restrições. Dado que as equações de restrições são dadas pela Equação 5, ao se resolver esse sistema para o vetor de coordenadas generalizadas q, é obtido um vetor que contém os valores para as respectivas coordenadas que ali estavam.

De acordo com Haug (1989) as equações de restrições possuem uma alta não-linearidade, tornando uma solução analítica praticamente impossível, além de que, devido a essa característica há uma grande possibilidade de que erros apareçam na formulação das equações de restrições, gerando que as restrições e movimentos do mecanismo possam não ser fisicamente satisfeitos e por consequência haja a não existência de uma solução para o sistema.

Shabana (2001) diz que na análise de posição, a solução do sistema de equações algébricas não lineares é feita por meio de métodos numéricos, como por exemplo o algoritmo de Newton-Raphson. Onde o mesmo trata de um processo iterativo que pode ser ativado por meio de um vetor de chute inicial, caso esse chute seja em um ponto num determinado tempo t, esse ponto pode ser determinado como q_i e a solução exata pode ser escrita como $q_i + \Delta q_i$, nesse caso, Δq_i é conhecido como o vetor das diferenças de Newton, pois o mesmo armazena as variações de posições que decorrem nos instantes de tempo.

Dessa forma Negrut (2014) demonstra o teorema da função implícita, onde o mesmo diz que, ao se considerar que em algum instante de tempo será encontrada uma solução para as equações de restrições onde $\Phi(q,t) = 0$,ou seja t_0, e se o Jacobiano da matriz de restrições é não singular, pode-se concluir que existe uma solução única, diferente da solução para o tempo onde $\Phi(q,t) = 0$. Portanto, em um pequeno intervalo de tempo existe uma dependência funcional explicita do vetor de coordenadas generalizadas e o tempo.

$$q = f(t), para\ todo\ |t - t_0| > \delta$$

Dessa forma a condição garante que o mecanismo assume uma configuração única em cada instante de tempo e que as equações de restrições foram formuladas de forma correta.

Considerando o teorema da função implícita valido, Nikravesh (1988) mostra que quando as equações de restrições tiverem uma solução igual ou aproximadamente a zero, a obtenção da solução pode ser escrita conforme a Equação 18:

$$\Phi(q,t) + \Phi_{qi}\Delta q_i = 0 \tag{18}$$

Onde pela Equação 19,

$$\Phi_{qi}\Delta q_i = -\Phi(q,t) \tag{19}$$

Assim, a Equação 19 pode ser resolvida para o vetor das diferenças de Newton, ou para um caso não iterativo, para o vetor das posições q, simplesmente manipulando para que o vetor das equações de restrições $\Phi(q,t)$ seja "dividido" pelo Jacobiano do mesmo. Obtendo a seguinte solução mostrada na Equação 20:

$$\Delta q_i = \left(\Phi_{qi}\right)^{-1} * -\Phi(q,t) \tag{20}$$

Para um caso iterativo o vetor das diferenças de Newton será atualizado, de acordo como o número de iterações. Armazenando todas as posições requisitadas, até que a norma do vetor seja menor ou igual a uma tolerância especificada, de acordo com a Equação 21.

$$q_{i+1} = q_i + \Delta q_i \tag{21}$$

De acordo com Wijckmans (1996) a diferenciação do vetor de restrições em relação ao tempo, utilizando a regra da cadeia, mostra que a velocidade é calculada da seguinte maneira:

A partir da Equação 18:

$$\Phi(q,t) + \Phi_{qi}q = 0 \tag{22}$$

Derivando em função do tempo (t), obtém-se a Equação 23:

$$\Phi_t + \Phi_q\dot{q} = 0 \tag{23}$$

De forma análoga a obtenção da posição a Equação 24 é apresentada:

$$\Phi_q \dot{q} = -\Phi_t = \nu \tag{24}$$

E assim, desde que o Jacobiano seja não singular e as soluções das equações de restrição tenham sido obtidas, é possível obter-se as velocidades para o mecanismo de acordo com a Equação 29.

$$\dot{q} = \left(\Phi_q\right)^{-1} * -\Phi_t \tag{29}$$

De forma semelhante a obtenção da equação de velocidade Shabana (2001) mostra que, diferenciando-se em função do tempo a equação da velocidade têm-se que:

A partir da Equação 24:

$$\Phi_q \dot{q} = -\Phi_t = 0 \tag{28}$$

Obtêm-se a Equação 29,

$$\frac{d}{dt}(\Phi_q \dot{q} + \Phi_t) = 0 \tag{29}$$

Derivando em função do tempo, por meio da regra da cadeia a Equação 30 é modelada,

$$(\Phi_q \dot{q} + \Phi_t)_q \dot{q} + \frac{\partial}{\partial t}(\Phi_q \dot{q} + \Phi_t) = 0 \tag{30}$$

Manipulando os termos chega-se na Equação 31,

$$(\Phi_q \dot{q})_q \dot{q} + (\Phi_t)_q \dot{q} + \Phi_{qt} \dot{q} + \Phi_q \ddot{q} + \Phi_{tt} = 0 \tag{31}$$

Organizando, a Equação 32 é modelada

$$\Phi_q \ddot{q} + (\Phi_q \dot{q})_q \dot{q} + 2\Phi_{qt} \dot{q} + \Phi_{tt} = 0 \tag{32}$$

De forma semelhante a análise de posição e velocidade têm-se a Equação 33 mostrando a função Gama e a Equação 34 mostrando o cálculo das acelerações,

$$\Phi_q \ddot{q} = -\left(\Phi_q \dot{q}\right)_q \dot{q} - 2\Phi_{qt} \dot{q} - \Phi_{tt} = \Upsilon \tag{33}$$

$$\ddot{q} = \left(\Phi_q\right)^{-1} * \Upsilon \tag{34}$$

E assim, é obtido o vetor de acelerações para o mecanismo.

Haug (1989) prova por meio de uma identidade da matriz de rotação, que existe um vetor de velocidade angular para o plano de referência local em relação ao plano de referência global, considerando que tanto o vetor de posição

quanto a matriz de rotação sejam funções temporais, em sistemas espaciais. Dessa forma, uma relação entre a matriz de rotação e o vetor de velocidade angular ω existe, e por meio de uma derivação temporal o vetor de aceleração angular passar a existir.

3.2.3. Parâmetros de Euler

Como foi demonstrado a análise de corpos em sistema espaciais requer um conjunto de coordenadas espaciais, além das coordenadas de posicionamento outras quatro são propostas pelo teorema de Euler, onde o mesmo diz “Se a origem de dois planos de referência baseados na regra a mão direita, coincidem, então eles podem ser descritos por uma rotação sobre o mesmo eixo”.

Em alguns casos a resolução de sistemas de equações pode não ser possível, dado a falta de parâmetros do problema, como por exemplo em casos de projetos de equipamentos. Dessa forma, os parâmetros de Euler entram para auxiliar na resolução.

Por meio de relações trigonométricas Shabana (2001), demonstra os parâmetros de Euler, representando-os em um vetor P, apresentado na Equação 35.

$$P = [e_0, e_1, e_2, e_3]^T \tag{35}$$

As relações entre os vetores unitários dos sistemas de origem única, podem ser substituídos na matriz de rotação A, mostrada na Equação (1). Os parâmetros geram uma nova restrição para o sistema, que deve obedecer a uma restrição, desde que não sejam independentes (HAUG,1989). A restrição é mostrada a seguir na Equação 36, junto com a matriz de rotação modificada pelos parâmetros, na Equação 37.

$$P^T P = e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \tag{36}$$

$$A = 2 * \begin{bmatrix} e_0^2 + e_1^2 - \frac{1}{2} & e_1e_2 - e_0e_3 & e_1e_3 + e_0e_2 \\ e_1e_2 + e_0e_3 & e_0^2 + e_2^2 - \frac{1}{2} & e_2e_3 - e_0e_1 \\ e_1e_3 - e_0e_2 & e_2e_3 + e_0e_1 & e_0^2 + e_3^2 - \frac{1}{2} \end{bmatrix} \quad (37)$$

Assim, com o acréscimo dos parâmetros de Euler, o número de equações de restrições pode ser recalculado, para uma quantidade de sete vezes o número de corpos presentes no sistema (Shabana,2001).

Além do que já foi mostrado, diversas propriedades dos parâmetros de Euler relacionadas a derivadas, relações e manipulações algébricas são descritas nas bibliografias, dentre algumas de importância elevada para esse trabalho têm-se na Equação 38:

$$G = [-e, -\tilde{e} + e_0 I] = \begin{bmatrix} -e_1 & e_0 & e_3 & -e_2 \\ -e_2 & -e_3 & e_0 & e_1 \\ -e_3 & e_2 & -e_1 & e_0 \end{bmatrix} \quad (38)$$

Que é uma matriz auxiliar utilizada em diversas funções, no presente trabalho ela é utilizada junto a uma manipulação, para garantir que os jacobianos de rotação das restrições do sistema possam ser dependentes das variações dos parâmetros de Euler. Na Equação 39

$$\tilde{\omega}' = A^T * \dot{A} \quad (39)$$

Outra relação de suma importância, utilizada para encontrar a matriz antissimétrica da velocidade angular do sistema, visto que sem isso não é possível obter a análise de aceleração.

Os parâmetros de Euler ainda influenciam as equações de velocidade, aceleração e por consequência as equações de movimento restringidas. Porém dada a seguinte condição apresentada por Haug (1989) onde derivada da restrição de Euler é realizada, é visto que não há uma influência das variáveis de velocidade angular na mesma, garantindo assim que essa restrição de velocidade não dependa da mesma e seja satisfeita dada essa condição, onde o mesmo raciocínio serve para a aceleração. Ou seja, nas análises de velocidade e aceleração para uma análise cinemática, as restrições de Euler podem ser ignoradas.

3.3. Matlab

3.3.1. Conceitos básicos

Neste tópico será abordado todos os conceitos que foram julgados importantes para o entendimento do programa MATLAB, de acordo com o assunto abordado, além de todo o procedimento detalhado dos métodos numéricos, que foi de maneira proposital ocultada nos tópicos anteriores, afim de que eles fossem detalhadamente explicados aqui.

Segundo Gilat (2004), o MATLAB é uma linguagem computacional com muitas ferramentas, seu nome vem do inglês Matrix Laboratory, ou laboratório de matrizes, dado que todos os seus dados são computados em matrizes. Com diversas funções, o MATLAB pode ser utilizado para computação matemática, modelagem e simulações, análise e processamentos de dados, visualização e gráficos e também para o desenvolvimento de algoritmos.

De acordo com Campos Filho (2007), um algoritmo é um conglomerado de comandos, que ao serem executados por uma máquina, resultam em uma sucessão finita de acontecimentos, esse procedimento garante uma abstração dos detalhes minuciosos da linguagem de programação podendo focar nos aspectos matemáticos do método.

Gilat (2004) em seus capítulos introdutórios, resume que o MATLAB é composto por 8 janelas principais, a partir das quais já se torna possível o entendimento da interface básica do mesmo, um resumo das suas interfaces pode ser visto no Quadro 1 a seguir.

Quadro 1 - Resumo das Principais Janelas do MATLAB

Janela	Tarefa
Comando	Janela Principal onde é inserido as variáveis e onde ocorre a execução dos algoritmos
Figura	Mostra a saída de comandos gráficos
Editora	Cria e depura arquivos scripts e funções
Ajuda	Fornece informação sobre os comandos

Ferramentas	Fornece acesso a ferramentas e documentos
Histórico de comandos	Armazena os logs de comandos que foram inseridos na janela de comando principal
Espaço de trabalho	Garante informação sobre as variáveis que estão sendo utilizadas
Diretório atual	Mostra os arquivos no atual diretório

Fonte: Adaptada de Gilat (2004)

Chin (2012), demonstram algumas tarefas que podem ser realizada no mesmo, dentre elas: entrada e saída de comandos, que é feita na janela de comando; operações aritméticas; operações algébricas, desde de fatoração polinomial a resolução de equações; gerenciamento de variáveis, essas que podem ser divididas em 3 grandes grupos, números de ponto flutuante, sequência de caracteres e expressões simbólicas.

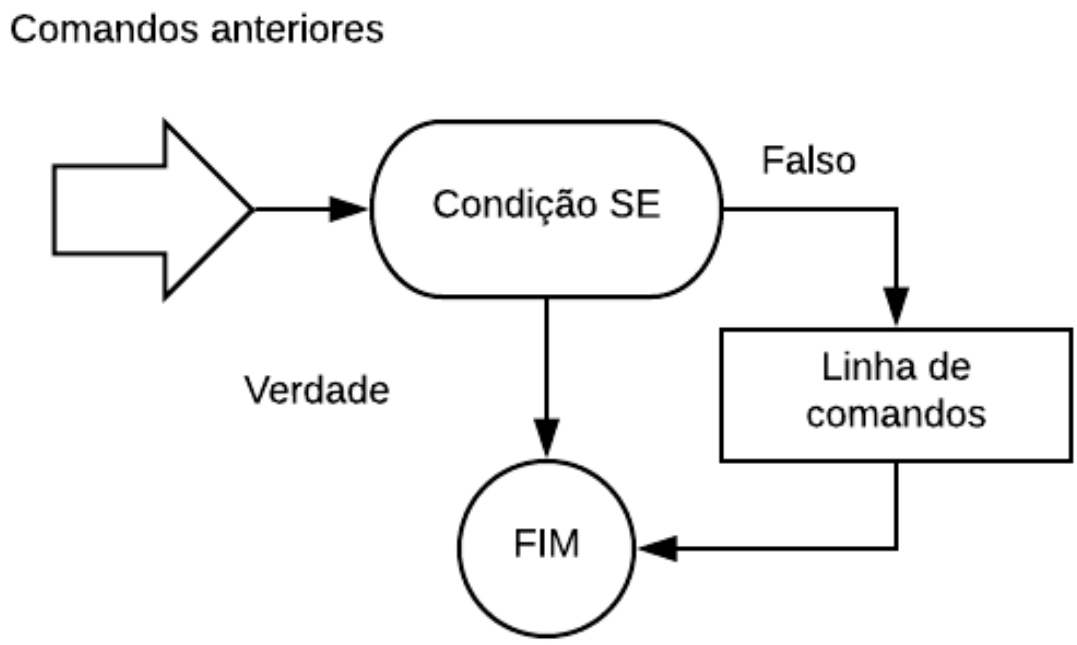
Campos Filho (2007) explica que o procedimento básico para a resolução de problemas computacionais segue um roteiro, o qual o mesmo demonstra que o primeiro passo é a definição do problema, saber o que deve ser resolvido; o segundo é modelagem matemática, transformar o problema real em um problema com linguagem matemática; a solução numérica é a terceira etapa do procedimento, onde nela é elaborado o algoritmo, a codificação do programa e o processamento do mesmo, e como quarto e último passo têm-se a análise dos resultados, onde nela é visto se a modelagem matemática para o problema foi feita de forma correta, observando se os resultados foram coerentes.

Hunt, Lipsman e Rosenberd (2001) avançam um pouco mais nos conceitos básicos de programação do MATLAB, definindo os tipos de operadores e suas funções. Abordando primeiro os operadores lógicos, o primeiro operador abordado é AND, que indica uma condição entre dois operandos, se uma é ativado o outro não é; o segundo operador é o OR, que informa a condição entre dois operandos onde tanto somente um pode ser ativado como ambos podem ser ativados e como terceiro operador tem-se o NOT, que atua somente sobre um operando, indicando o oposto do operando.

Chapra e Canale (2015) tratam de demonstrar os operadores de condição, explicando que os mesmos têm a função de tomar uma decisão para

a execução de um grupo de comandos, também é dito que esses tipos de operadores são denominados de ramificações, pois indicam mudanças de caminhos. Uma das estruturas de condição é a if – end, que significam se - fim, respectivamente, por meio delas é possível estabelecer um critério de desvio, que seria o if, e um critério de parada, end. Uma exemplificação pode ser dada a partir do Fluxograma 1,

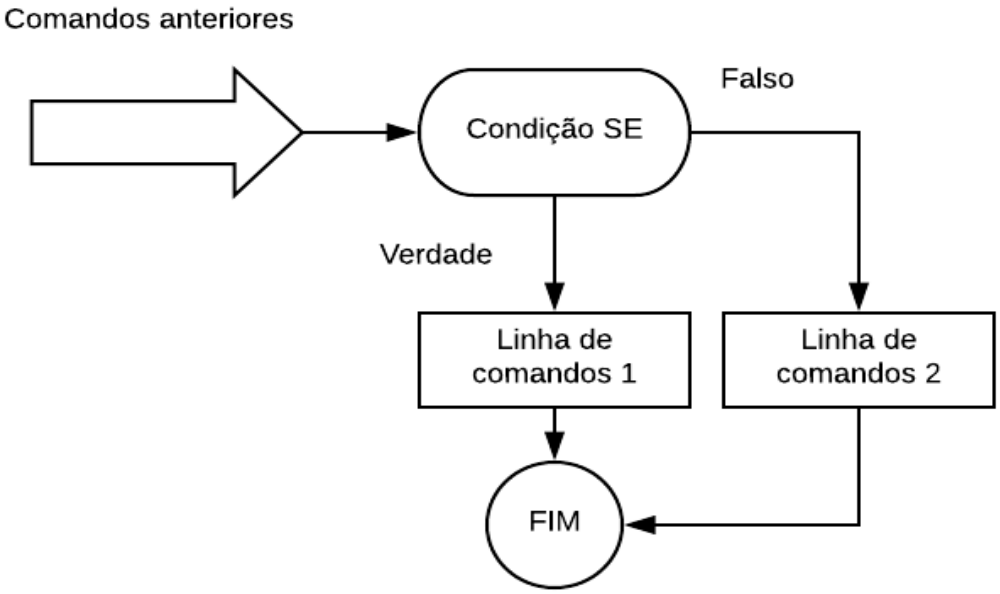
Fluxograma 1 - Exemplo de operador SE



Fonte: Adaptado de Gilat (2004)

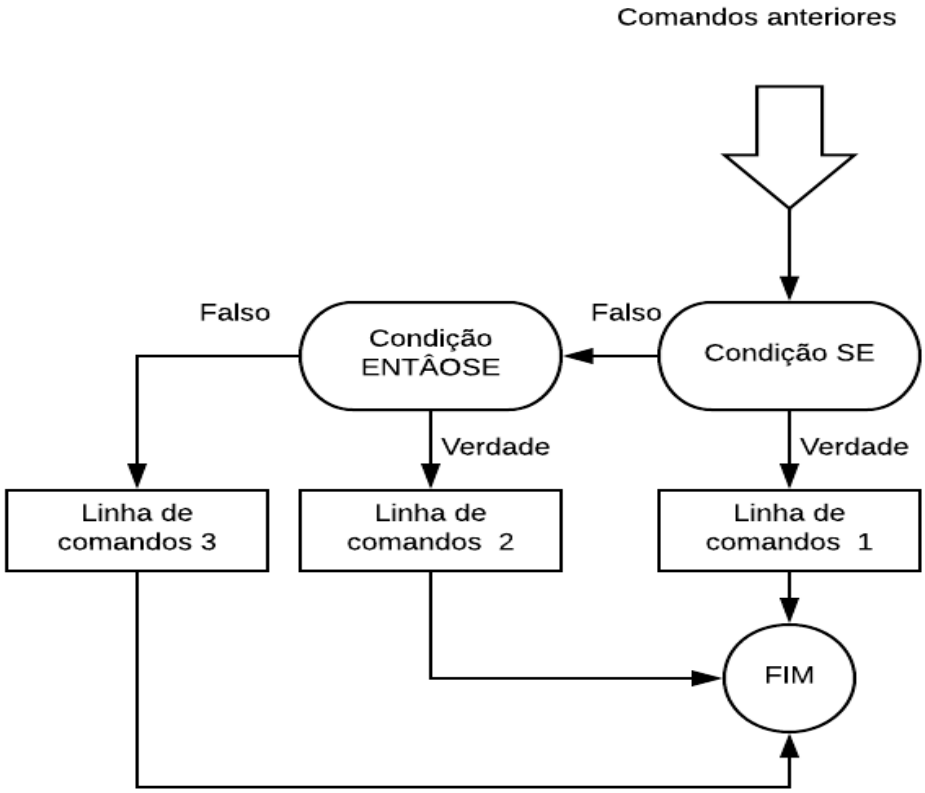
Hunt, Lipsman e Rosenberd (2001) continuam demonstrando os operadores de condição, agora com os comandos else (então) e ifelse (então se), semelhantes ao comando if (se), esses dois comandos também indicam mudança de caminho caso uma dada condição seja atingida, fazendo com seja tomada um novo conjunto de comandos. O comando else, indica uma condição para a tomada de um conjunto de comandos específicos, enquanto o ifelse indica uma outra condição, caso a segunda condição seja atendida, uma exemplificação mais clara pode ser vista nos Fluxogramas 2 e 3.

Fluxograma 2 - Exemplo do operador ELSE



Fonte: Adaptado de Gilat (2004)

Fluxograma 3 - Exemplo do operador IFELSE



Fonte: Adaptado de Gilat (2004)

Outra forma para processos iterativos é apresentada por Chapra e Canale (2015), conhecidos como loops, os comandos envolvidos nestes processos são

executados inúmeras vezes, onde em cada execução á atribuído um valor para as variáveis em uso, ou seja, se está sendo analisado a posição de um mecanismo durante um intervalo de tempo, a cada instante que se passa é atribuído a variável posição um novo valor.

Entrando a fundo no assunto de loops, Gilat (2004) explica que no MATLAB existem dois tipos de loops, os loops for – end e os loops while – end. Os loops do tipo for – end (para - fim), excutam os comandos situados no seu interior por um número determinado de vezes, a quantidade de iterações que esse tipo de loop executará é determinada, geralmente, por uma condição, assim, quando o código estiver em execução, ao chegar na linha do loop, ele ficará “preso” no mesmo, até que a condição seja atendida, quando isso acontecer o programa sai do loop e segue o restante do algoritmo.

Os loops do tipo while – end (enquanto - fim), são explicados por Hunt, Lipsman e Rosenberd (2001), esse tipo de loop, trabalha na situação onde não são conhecidos a quantidade de passos, ou seja, não é atribuída uma meta, como nos loops for – end, nesse caso quando o algoritmo em execução lê a linha onde o while está, ele verifica se a condição estabelecida por esse while foi atingida, caso não, o programa ignora o while e passa para o end, não executando o loop, caso o contrário, o algoritmo executa os comandos armazenados dentro do loop, e volta para o início para assim averiguar se a condição foi atingida.

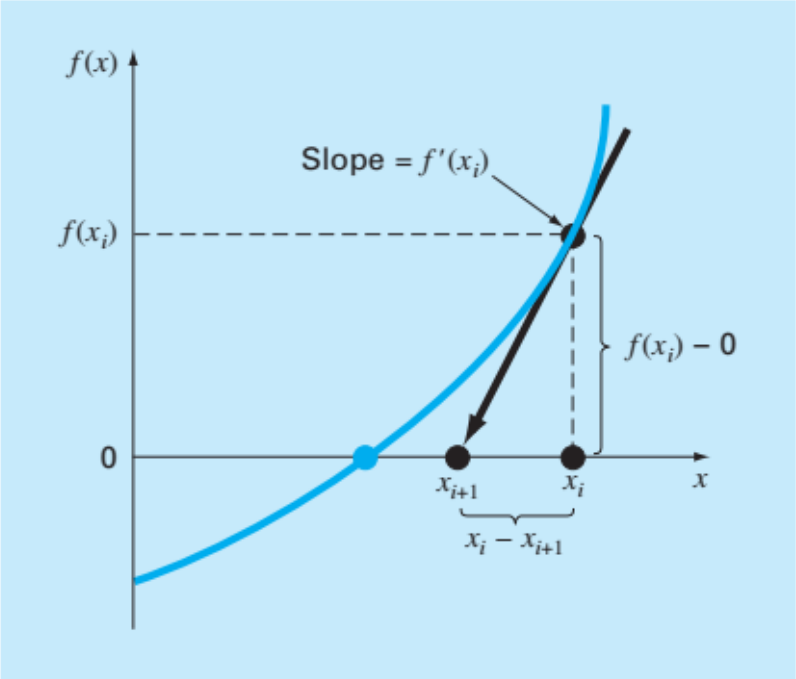
Apresentada uma noção dos comandos básicos para utilização do MATLAB, uma explicação um pouco mais profunda dos algoritmos de Newton-Raphson e Newmark podem ser feitas agora.

3.3.2. Procedimento Numérico para análise cinemática

Como foi dito anteriormente, no tópico de análise cinemática, um dos algoritmos utilizados para se realizar o procedimento é o de Newton – Raphson, Chapra e Canale (2015) definem esse método numérico como um dos mais utilizados para a busca de raízes de equações, onde se um chute inicial numa raiz é dado como x_i, uma tangente pode ser estendida a partir do ponto

$[x_i, f(x_i)]$, assim o ponto onde a tangente cruza o eixo X representa o valor da estimativa da raiz. Graficamente, a Figura 12, pode exemplificar o que foi dito.

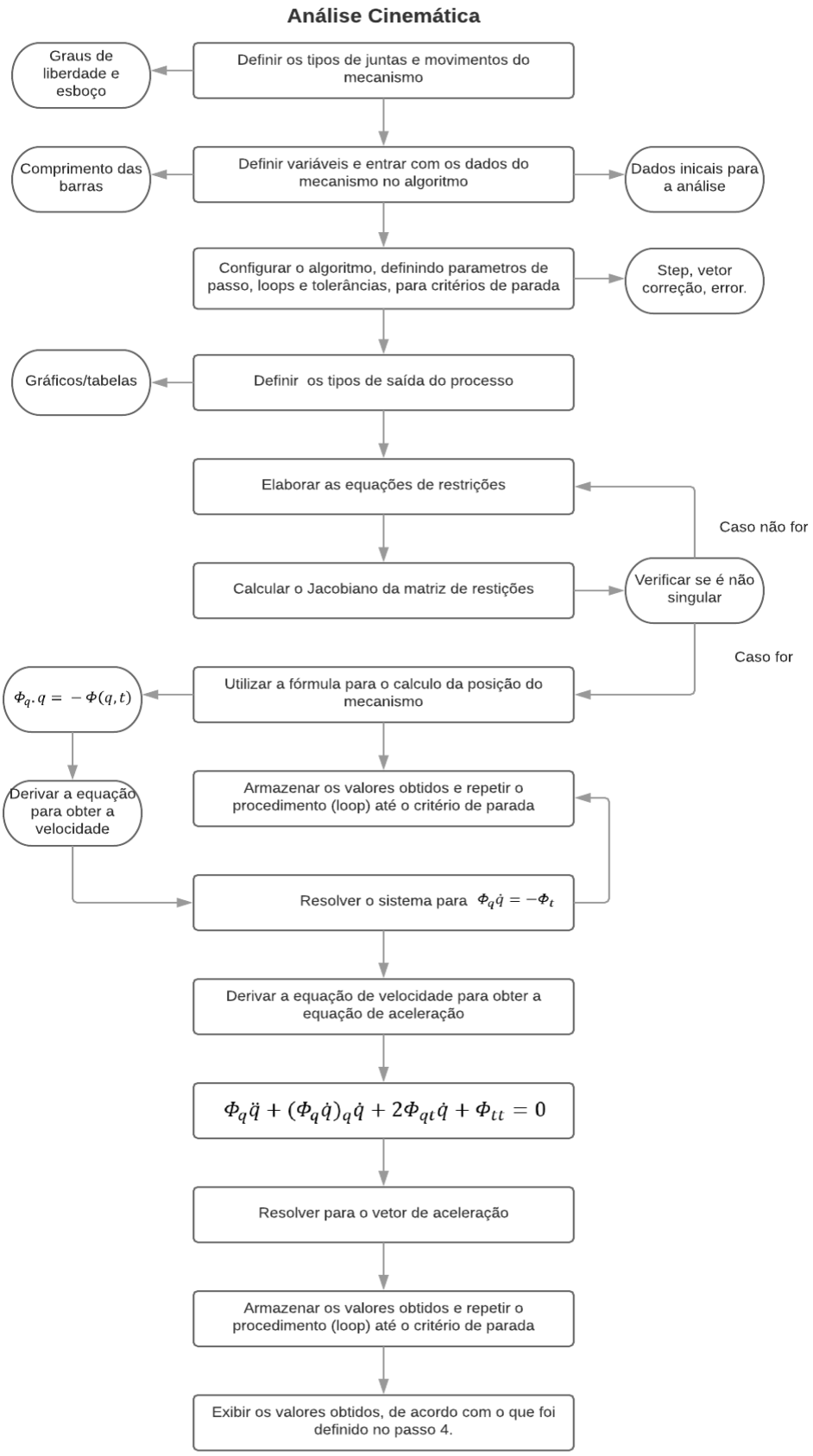
Figura 12 - Representação Gráfica do Método de Newton-Raphson



Fonte: Chapra e Canale (2015)

Haug (1989) complementa que o método numérico de Newton-Raphson pode ser utilizado para análise cinemática quando a matriz de restrições e seu Jacobiano são calculados, porém, quando o chute inicial não é próximo da solução, na maioria dos problemas o método pode não convergir, aproximando a solução para o sistema de equações a zero. Por fim a relação existente entre o teorema da função implícita com o método numérico de Newton-Raphson, é que se o teorema da função implícita não é satisfeito, ou seja, se o Jacobiano é singular, o método numérico não irá convergir, sendo este um dos critérios de convergência para o método. Dessa forma, o procedimento pode ser exemplificado no Fluxograma 4.

Fluxograma 4 - Procedimento para análise Cinemática



Fonte: Autoria Própria (2018)

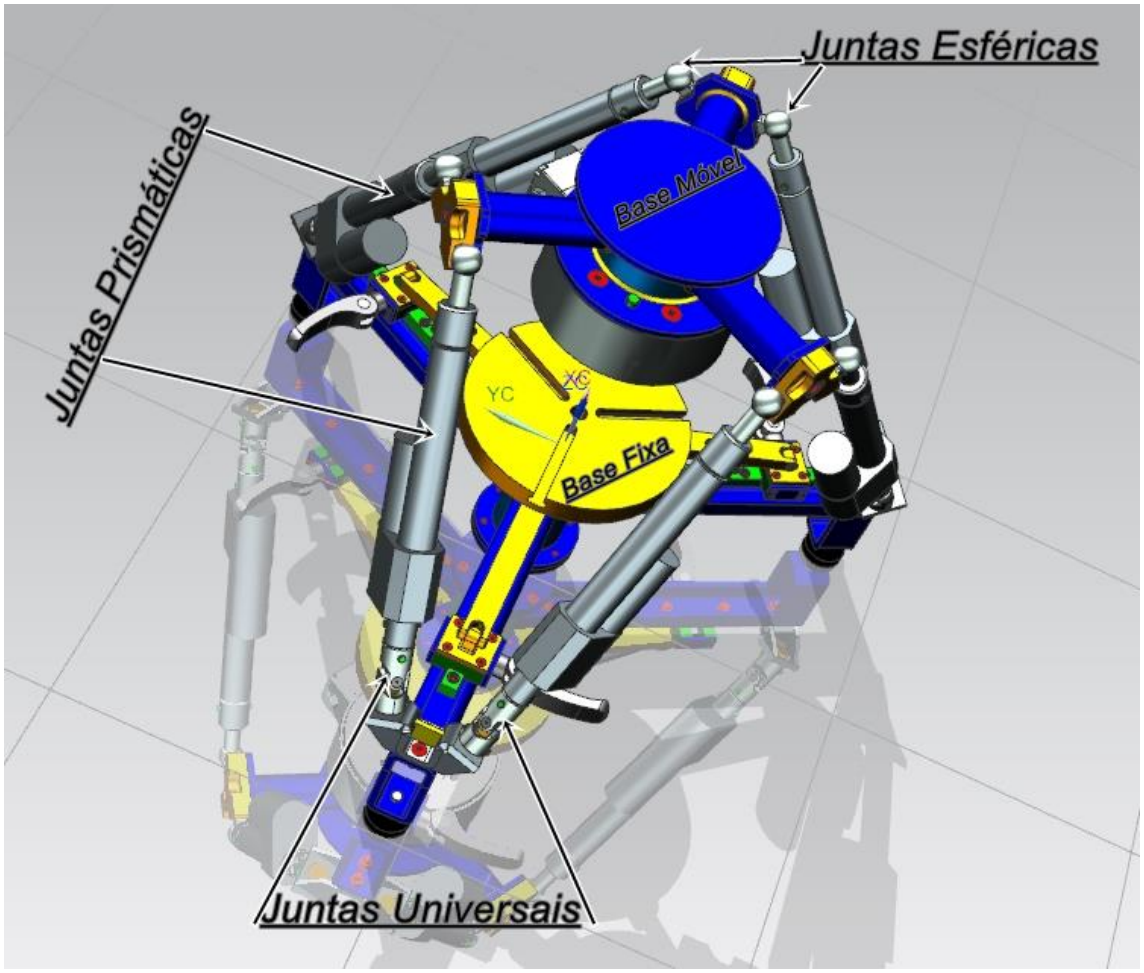
4. MATERIAIS E MÉTODOS

Para a realização da análise cinemática do modelo proposto da plataforma de Stewart, foram percorridas várias etapas, essas serão abordadas nesse tópico, detalhando as mesmas.

4.1. Modelo da plataforma

Como detalhado no Tópico 3.1, a plataforma escolhida segue o padrão observado nas literaturas, com o auxílio do software Siemens NX 12, o modelo da plataforma pode ser visto na Figura 13.

Figura 13 - Plataforma De Stewart utilizada



Fonte: Autoria Própria (2018)

Na Figura 13 pode-se observar os principais componentes, que são: duas plataformas, uma inferior fixa (amarela) e uma superior móvel (azul), bem como seis braços, compostos por juntas.

A partir da plataforma modelada, pode-se constatar a presença de juntas esféricas, juntas universais, juntas de translação, em uma quantidade de seis unidades de cada, além dos elementos citados anteriormente, como barras e bases.

O movimento é dado por meio das juntas de translação que se expandem e comprimem, presentes nas juntas de translação. A partir dessa analise simples, as equações de restrições podem ser modeladas.

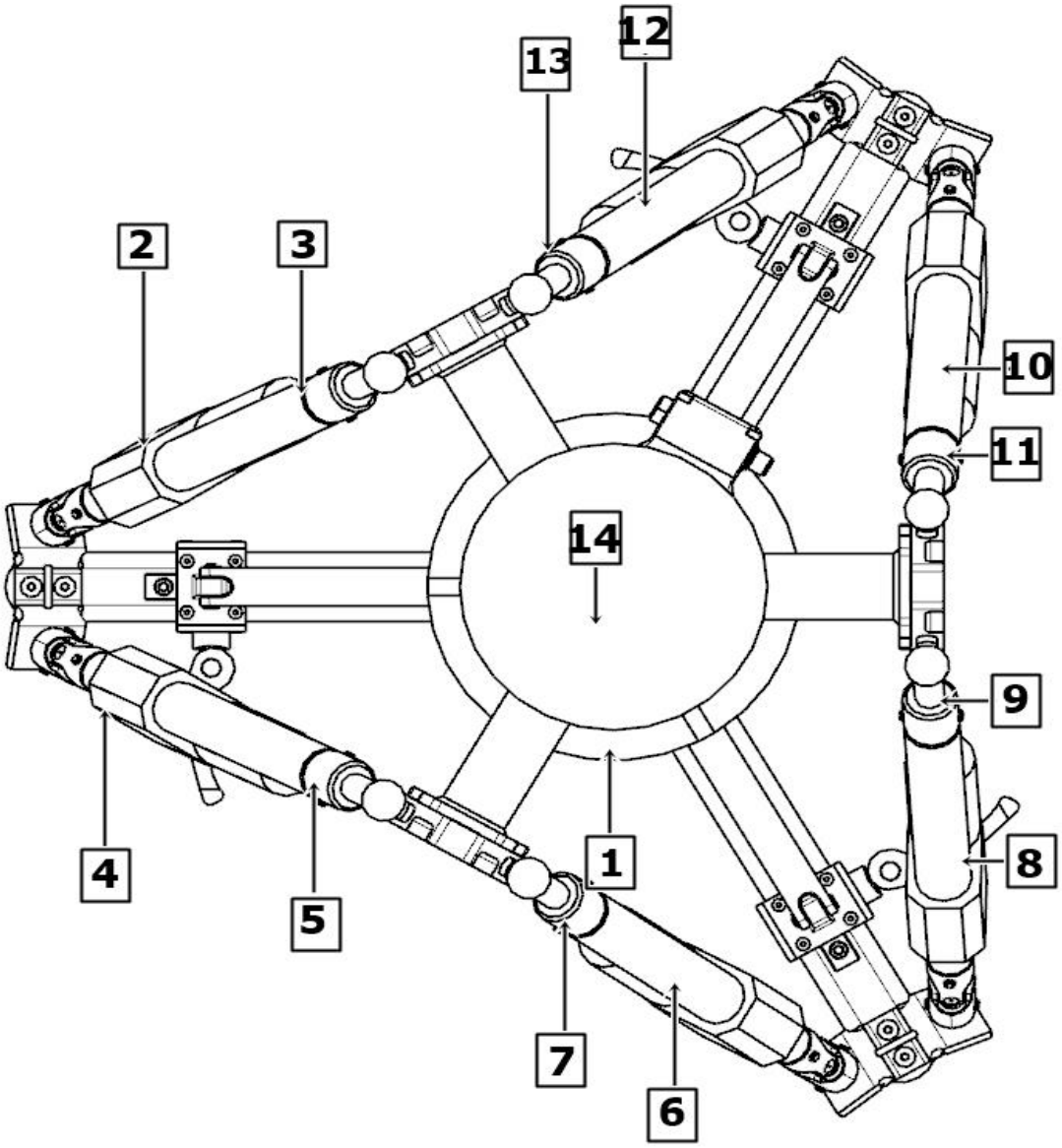
Com o auxílio do Software Siemens NX 12, também foi possível obter dados importantes para o início da análise cinemática. A partir de ferramentas inclusas no software mencionado anteriormente foi possível posicionar os sistemas de coordenadas global e locais em cada corpo.

Dado esse passo, propriedades como distância entre as origens dos sistemas de coordenadas, vetores de posição local, vetores unitários e orientações iniciais foram mensurados. Dessa forma, os parâmetros iniciais para a análise cinemática foram todos aferidos, podendo proceder com a modelagem das restrições e início da análise cinemática.

Para isso os corpos apresentados na Figura 13, foram enumerados, tal enumeração é de vital importância, para que as coordenadas generalizadas assumam seus valores corretamente, evitando assim uma confusão nos resultados da análise cinemática.

A enumeração é utilizada para a modelagem das restrições, visto que é necessário mais que uma equação por juntas, como foi dito anteriormente, existe mais de uma junta por tipo no modelo de plataforma usado. Assim, a Figura 14, apresenta a enumeração dos corpos.

Figura 14 – Numeração dos corpos



Fonte: Autoria própria (2019)

Dessa forma, têm-se que a base fixa ficou como o corpo 1, e a base móvel como o corpo 14. As juntas prismáticas unem dois corpos, um inferior e outro superior, os inferiores ficaram com as numerações pares, e os superiores com as numerações impares, ou seja, em outras palavras pode-se dizer que as juntas esféricas ficam como os números impares.

5. RESULTADOS E DISCUSSÕES

O inicio dos resultados se dá pela modelagem das equações de restrições do sistema. Essas equações, necessitam de parâmetros, para que sejam modeladas de forma correta. Tais parâmetros são apresentados nos Quadros 2, 3 e 4.

Quadro 2 - Distâncias entre as origens dos sistemas de coordenadas global e local

X	Y	Z	Corpo
0	0	0	1
177,528	167,4450	95	2
173,2830	106,3690	190	3
56,2480	237,4670	95	4
5,4470	203,2520	190	5
-233,776	70,0210	95	6
-178,760	96,8820	190	7
-233,776	70,0210	95	8
-178,760	-96,8820	190	9
56,2480	-237,467	95	10
5,4770	-203,252	190	11
177,528	-167,445	95	12
173,283	-106,369	190	13
0	0	285	14

Fonte: Autoria Própria (2018)

Quadro 3 - Parâmetros de Euler Iniciais

Parâmetros de Euler (4 por corpo)				Corpo
[1	0	0	0]	1
[0,531030903337024	0,446342042	-0,714397344	-0,0916623091903980]	2
[0,531030903337024	0,446342042	-0,714397344	-0,0916623091903980]	3
[0,344889104085633	0,395532718	-0,743774489	-0,414056678949998]	4
[0,344889104085633	0,395532718	-0,743774489	-0,414056678949998]	5
[0,186126322209598	0,841879426	-1,378202255	-0,505723267254085]	6
[0,186126322209598	0,841879426	-1,378202255	-0,505723267254085]	7
[0,186126322209598	-0,841879426	0,029348211	0,505723267254085]	8
[0,186126322209598	-0,841879426	0,029348211	0,505723267254085]	9
[0,344889104085633	-0,395532718	-0,743774489	0,414056678949998]	10
[0,344889104085633	-0,395532718	-0,743774489	0,414056678949998]	11
[0,531020905013085	-0,446350446	-0,714410795	0,0916528935330713]	12
[0,531020905013085	-0,446350446	-0,714410795	0,0916528935330713]	13
[1	0	0	0]	14

Fonte: Autoria Própria (2018)

Quadro 4- Vetores de posição local para cada junta

Junta Esférica			
Corpos i	X	Y	Z
i3	113,019	0	0
i5	113,019	0	0
i7	113,019	0	0
i9	113,019	0	0
i11	113,019	0	0
i13	113,019	0	0

Junta Prismatica			
Corpos i	X	Y	Z
i2	-100	0	0
i4	-100	0	0
i6	-100	0	0
i8	-100	0	0
i10	-100	0	0
i12	-100	0	0

Junta Universal			
Corpos i	X	Y	Z
i2	181.7740	228.5220	0
i4	107.0180	271.6820	0
i6	-288.7930	43.1600	0
i8	-288.7930	-43.1600	0
i10	107.0180	-271.6820	0
i12	181.7740	-228.5220	0

Corpos j	X	Y	Z
j14 - 3	169.0370	45.2930	0
j14 - 5	-45.2930	169.0370	0
j14 - 7	-123.7440	123.7440	0
j14 - 9	-123.7440	-123.7440	0
j14 - 11	-45.2930	-169.0370	0
j14 - 13	1.690.370	-452.930	0

Corpos j	X	Y	Z
j3	100	0	0
j5	100	0	0
j7	100	0	0
j9	100	0	0
j11	100	0	0
j13	100	0	0

Corpos j	X	Y	Z
j1 - 2	-113.0190	0	0
j1 - 4	-113.0190	0	0
j1 - 6	-113.0190	0	0
j1 - 8	-113.0190	0	0
j1 - 10	-113.0190	0	0
j1 - 12	-113.0190	0	0

Fonte: Autoria Própria (2018)

5.1. Analise Cinemática

Nesse tópico serão modeladas de forma literal as principais equações e funções que forem necessárias para a realização da análise cinemática.

Com o auxílio de Haug (1989), onde o mesmo informa as equações de restrições para cada tipo de junta, no Quadro 5 tem-se que:

Quadro 5 - Relações vetoriais das principais restrições

Corpo Fixo (Φ^C)	-	-
Esférica (Φ^s)	$\Phi^s = (P_i, P_j) = 0$	(40)
Universal (Φ^U)	$\Phi^s = (P_i, P_j) = 0$ $\Phi^{d1} = (h_i, h_j) = 0$	(41)
Translação (Φ^T)	$\Phi^{p1} = (h_i, h_j) = 0$ $\Phi^{p2} = (h_i, d_{ij}) = 0$ $\Phi^{d1} = (f_i, f_j) = 0$	(42)
Motora (Φ^M)	$\Phi^{d2} = (h_i, d_{ij}) - C(t) = 0$	(43)
Euler (Φ^P)	$P_i^T P_i - 1 = 0$	(44)

Fonte: Adaptado de Haug (1989)

Onde as Equações 40 a 44 são relações de paralelismo e ortogonalidade entre os vetores unitários e de localização presentes em cada sistema de coordenadas de cada corpo envolvido. Que quando expandidas tem a seguinte forma apresentadas no Quadro 6, pelas Equações 45 a 49.

Quadro 6 - Modelagem das restrições

Corpo Fixo (Φ^C)	$r_1 + A_1 * S_1'$ $\frac{1}{2}(\tilde{e}_1 + e_0 * \mathrm{I})$	(45)
Esférica (Φ^s)	$\Phi^s = (r_j + A * S_j) - (r_i + A * S_i) = 0$	(46)
Universal (Φ^U)	$\Phi^s = (r_j + A * S_j) - (r_i + A * S_i) = 0$ $\Phi^{d1} = h^T * A^T * A * h = 0$	(47)
Translação (Φ^T)	$\Phi^{p1} = \begin{bmatrix} f^T * A * A^T * h \\ g^T * A * A^T * h \end{bmatrix} = 0$ $\Phi^{p2} = \begin{bmatrix} f^T * A^T * (r_j + A * S_j - r_i) - f^T * S_i \\ g^T * A^T * (r_j + A * S_j - r_i) - g^T * S_i \end{bmatrix} = 0$ $\Phi^{d1} = f^T * A^T * A * f = 0$	(48)
Motora (Φ^M)	$\Phi^{d2} = (h^T * A^T * (r_j + A * S_j - r_i) - h^T * S_i) - C(t) = 0$	(49)
Euler (Φ^P)		

Fonte: Adaptado de Haug (1989)

De forma compacta, a Equação 50, apresenta o Quadro 6.

$$\Phi(q,t) = \begin{bmatrix} \Phi^K(q) \\ \Phi^D(q,t) \\ \Phi^P(q) \end{bmatrix} = 0 \tag{50}$$

Como dito anteriormente, o número de equações de restrições será 7 vezes o número de corpos presentes no sistema, como o sistema tem 14 corpos então terão 84 equações de restrições.

As equações de restrições são modeladas da seguinte maneira apresentada pela Equação 51:

$$\Phi(q,t) = \begin{bmatrix} \Phi^C \\ \Phi^S \\ \Phi^T \\ \Phi^U \\ \Phi^M \\ \Phi^{Pi} \end{bmatrix} = 0 \tag{51}$$

E assim o processo é repetido até abordar todas os braços e corpos presentes no sistema, levando a um total de 98 equações de restrições.

O próximo passo é o cálculo da matriz jacobiana, a mesma como explicado no referencial teórico é obtida por meio da derivação das equações de restrições por um vetor q, esse vetor armazena todas as variáveis que são necessárias para descrever a posição e orientação de todos os corpos. Como o sistema possui 14 corpos, serão necessárias 98 variáveis.

Como o sistema está sendo analisado no espaço, para cada corpo serão necessárias 3 variáveis de posição e 4 variáveis de orientação, ou seja, 7 variáveis por corpo, totalizando as 98 variáveis. As 4 variáveis de orientação são os parâmetros de Euler.

Para auxiliar na modelagem da matriz jacobiana, Haug (1989) fornece uma tabela contendo as derivadas das principais condições de posição entre os vetores unitários. Dessa forma a partir do Quadro 7, matriz jacobiana é montada.

Quadro 7 - Jacobiano das restrições

Condição	$\boldsymbol{\Phi_{ri}}$	$\boldsymbol{\Phi_{rj}}$	$\boldsymbol{\Phi_{\pi i}}$	$\boldsymbol{\Phi_{\pi j}}$
$\Phi^{d1}(a_i, a_j)$	0	0	$-a_j'^T A_j^T A_i \tilde{a}_i'$	$-a_i'^T A_i^T A_j \tilde{a}_j'$
$\Phi^{d2}(a_i, d_{ij})$	$-a_i'^T A_i^T$	$-a_i'^T A_i^T$	$(a_i'^T \tilde{s}_i'^P - d_{ij}{}^T A_i \tilde{a}_i')$	$-a_i'^T A_i^T A_j \tilde{s}_j'^P$
$\Phi^{s}(P_i, P_j)$	$-I$	I	$A_i \tilde{s}_i'^P$	$-A_j \tilde{s}_j'^P$

Fonte: Adaptado de Haug (1989)

Obtidas as restrições e o jacobiano, o início da análise de posição foi dado, utilizando uma variação no tempo de aproximadamente 6,14s, o algoritmo foi montado.

Com a matriz jacobiana e as restrições modeladas, utilizando a Equação (20) e o algoritmo de Newton Raphson é possível obter as posições para um determinado intervalo de tempo. Para fins organizacionais o algoritmo utilizado está localizado no Apêndice A.

Para realizar a análise de velocidade uma modificação foi feita na matriz jacobiana do sistema, sendo reduzida para uma matriz 84x84, isso pois as restrições de Euler nessa etapa foram excluídas como explicado no tópico de parâmetros de Euler 3.2.3.

Logo, modelado o jacobiano de velocidade é necessária a obtenção da função Nu, como mostrado na Equação (24), a mesma é obtida por meio da derivação temporal das equações de restrições usadas para a modelagem do jacobiano de velocidade, ou seja, as restrições cinemáticas e motoras, resultando na Equação 52:

$$\nu = \begin{bmatrix} -\Phi_t^K \\ -\Phi_t^D \end{bmatrix} \tag{52}$$

Utilizando-se novamente do algoritmo de Newton Raphson, de forma semelhante a análise de posição, a análise de velocidade é realizada para o mesmo intervalo de tempo, obtendo assim a variação de velocidade dos corpos do mecanismo.

Por fim a análise de aceleração é feita de forma análoga a de velocidade, utilizando-se a mesma matriz jacobiana, porém com uma função diferente, essa função, foi mostrada anteriormente na Equação (33) sendo a função gama.

Onde para um sistema espacial, as derivadas das principais relações vetoriais, que são a de restrição absoluta, esfericidade, ortogonalidade 1 e ortogonalidade 2, são apresentadas, respectivamente, pelas funções a seguir no intervalo de Equações da 53 a 57:

$$\Upsilon^C = -A_i * \tilde{\omega}_i' * s_i'^P * \omega_i' \tag{53}$$

$$\Upsilon^O = -\frac{1}{2}\left(\dot{\acute{e}}_i + \acute{e}_0 * I\right)\omega_i' \tag{54}$$

$$\Upsilon^S = -A_i * \tilde{\omega}_i' * \tilde{\omega}_i' * s_i'^P - A_j * \tilde{\omega}_j' * \tilde{\omega}_j' * s_j'^P \tag{55}$$

$$\Upsilon^{d1} = -a_j'^T\left[A_j^T * A_i * \tilde{\omega}_i' * \tilde{\omega}_i' + \tilde{\omega}_j' * \tilde{\omega}_j' * A_j^T * A_i\right]a_i + 2 * \omega_j'^T * \tilde{\alpha}_j' * A_j^T * A_i * \tilde{\alpha}_i' * \omega_i' \tag{56}$$

$$\begin{aligned}\Upsilon^{d2} = {} & 2 * \omega_i'^T * \tilde{\alpha}_i' * A_i^T\left(\dot{r}_i - \dot{r}_j\right) + 2 * s_j'^P * \tilde{\omega}_j' * A_j^T * A_i * \tilde{\omega}_i' * a_i - s_i'^P * \tilde{\omega}_i' * \tilde{\omega}_i' * a_i' \\ & -s_j'^{PT} * \tilde{\omega}_j' * \tilde{\omega}_j' * A_j^T * A_i * a_i' - d_{ij}^T * A_i * \tilde{\omega}_i' * \tilde{\omega}_i' * a_i'\end{aligned} \tag{57}$$

Dessa forma, seguindo a mesma ordem utilizada para a modelagem das restrições, a função gama é montada a fim de obter-se a Equação 58:

$$\Upsilon = \begin{bmatrix} -\Phi_{tt}^{K} \\ -\Phi_{tt}^{D} \end{bmatrix} \tag{58}$$

Que de forma expandida se torna a forma apresentada pela Equação 59:

$$\Upsilon = \begin{bmatrix} \Upsilon^{C} \\ \Upsilon^{O} \\ \Upsilon^{S} \\ \Upsilon^{T} \\ \Upsilon^{U} \\ \Upsilon^{M} \end{bmatrix} \tag{59}$$

Assim, a montagem da função gama é feita até que todas as 84 equações de restrições sejam modeladas para suas respectivas condições gama.

Aplicando no Newton-Raphson, para o mesmo intervalo de tempo usado anteriormente nas análises de posição e velocidade, é possível obter-se as acelerações para os 14 corpos do sistema.

Assim, nessa seção foram apresentados os meios e caminhos escolhidos para a realização da análise cinemática, explicando de forma literal como foram obtidos os resultados, a parte de resultados e cálculos, foi feita numericamente por meio de programação.

5.2. Discussões sobre os resultados da análise cinemática

Com o auxílio do software MATLAB R2016a, um compilado de funções fora modelado, esse método garante uma melhor organização dos scripts, fornecendo segurança e a possibilidade de futuras modificações e implementações sem muitas dificuldades.

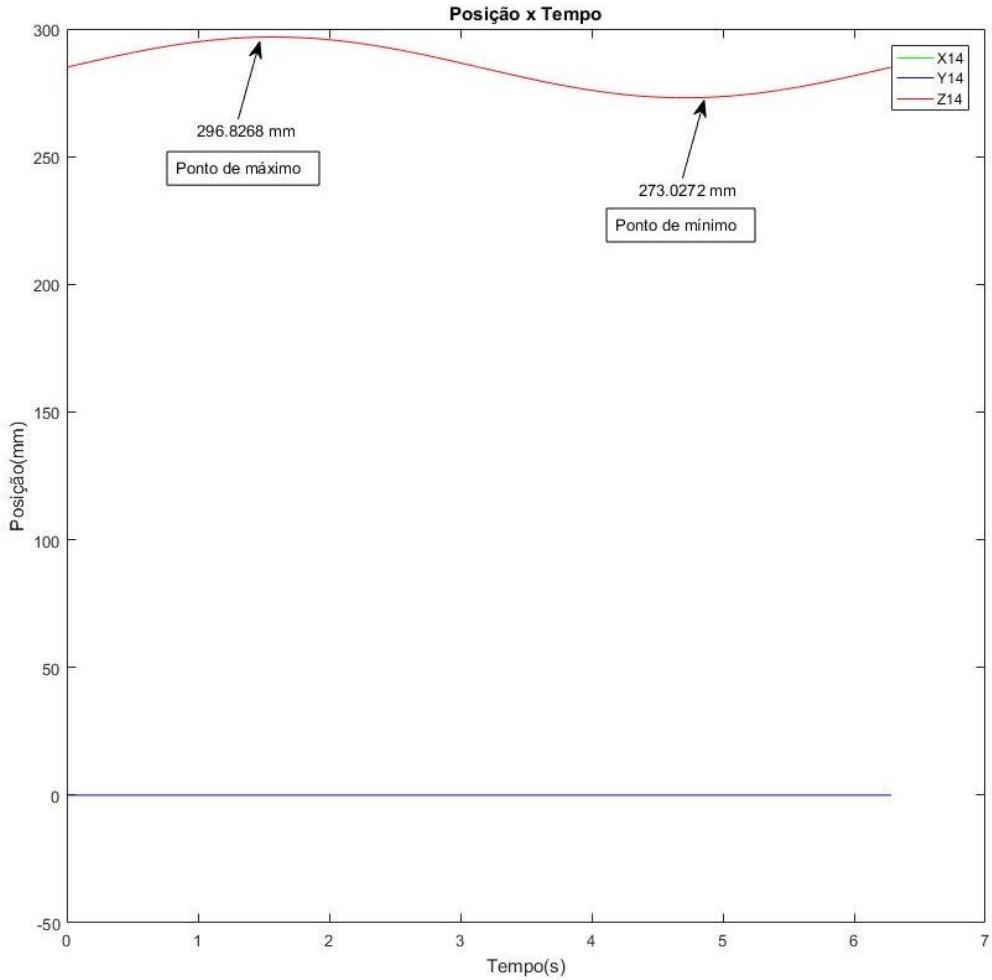
O método escolhido para expor os resultados das análises foi por meio de gráficos, aliás, diversos outros meios poderiam ser escolhidos, como por exemplo: a exibição numérica dos valores assumidos pelas variáveis durante a variação do tempo ou a animação em tempo real de um protótipo da plataforma usada.

Porém, a exibição por meio de gráficos foi escolhida, pois além de permitir uma análise trivial, é possível detectar erros de forma mais simples, pois um simples movimento da curva gerada, fora do padrão, gera suspeitas sobre um possível erro.

No sistema (plataforma) foi escolhido ser analisado ‘’apenas’’ o corpo 14, ou seja, a base móvel da plataforma. Ao invés de mostrar os 14 corpos variando sua posição, velocidade e aceleração, foi julgado mais coerente mostrar apenas a base móvel, visto que o movimento do corpo 14 é o resultado final dos movimentos de todos os outros 13 corpos do sistema. Além de garantir uma melhor organização, pois para mostrar a análise cinemática de 13 corpos ao invés de um só, haveria uma demanda muito grande de gráficos.

Utilizando uma função de movimento $C(t)$ do tipo: $l_0 + 10 * \sin(t)$ onde l_0 é o comprimento inicial dos atuadores da plataforma, têm-se que o a posição do corpo 14 é mostrada no Gráfico 1:

Gráfico 1 - Análise de Posição



Fonte: Autoria Própria (2019)

Pelo Gráfico 1, pode ser visto que praticamente a base móvel só possui variação na sua altura, ou variável Z14. Onde o comportamento das posições

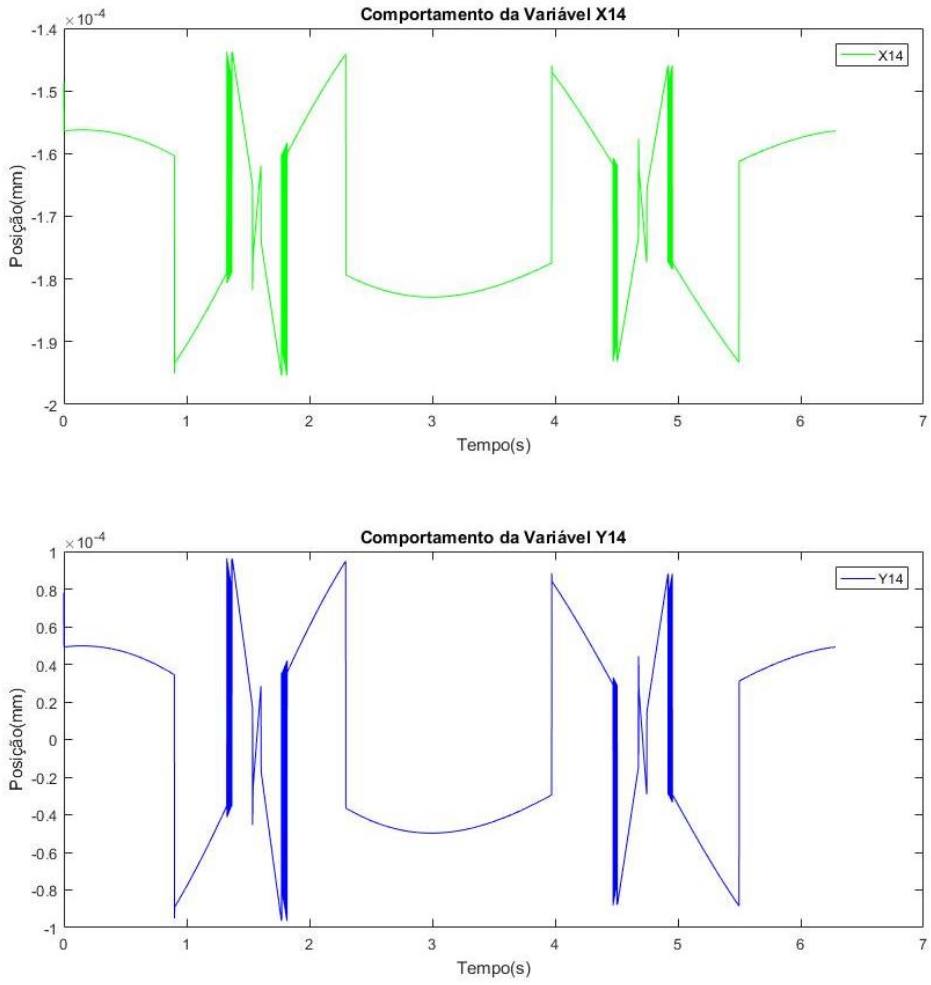
nas direções de X e Y estão sobrepostos, mostrando assim somente o a variável Y14, que foi a última a ser plotada.

Lembrando que no Gráfico 1, só estão sendo mostradas as variações das variáveis de posição (X, Y, Z), mais a frente será mostrada as variações das orientações desse corpo, regidas pelos parâmetros de Euler.

Porém, o movimento da variável X14 ou eixo X não está presente, isso porque seu movimento pode ser aproximado de zero, pois sua variação varia a precisão de 0,0001 mm.

Para isso, o Gráfico 2 foi plotado mostrando o comportamento das variáveis X14 e Y14.

Gráfico 2 - Comportamento das variáveis X14 e Y14



Fonte: Autoria Própria (2019)

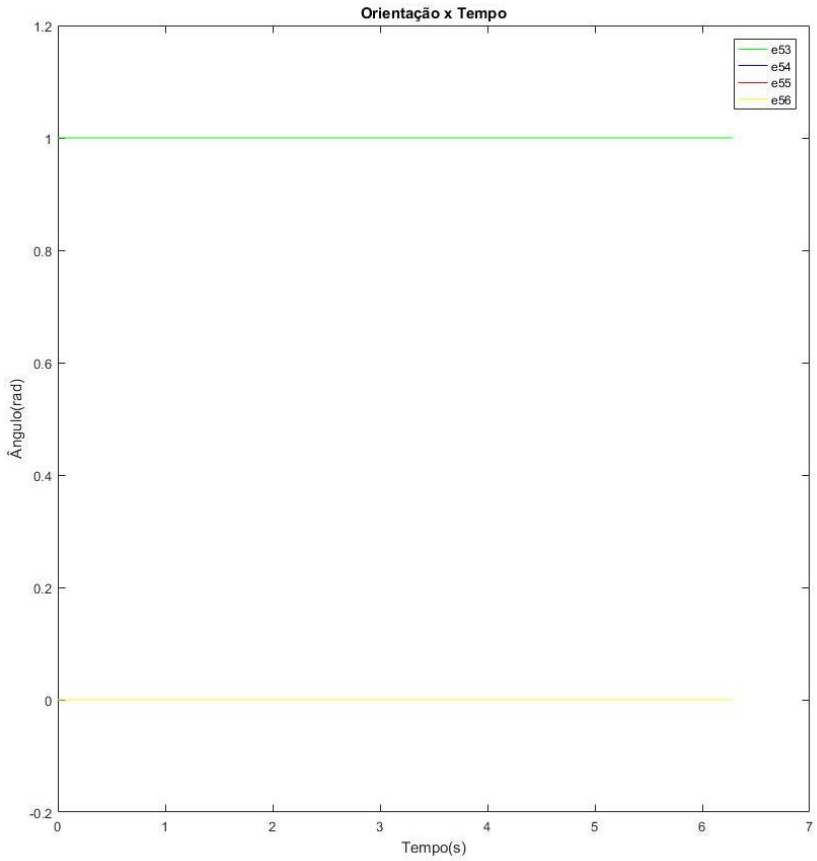
O pequeno movimento das variáveis X14 e Y14 é um pseudomovimento do corpo 14 nessas direções, esses são gerados por erros de arredondamento dos

dados de entrada da plataforma, ou seja, se dados com uma maior número de casas decimais fossem usados, esses movimentos iriam ser corrigidos, levando-os a tender a zero, como mostrado no Gráfico 1.

A magnitude desses erros mostrados no Gráfico 2, foi medida a partir dos máximos e mínimos dessas funções, que foram para a variável X14, um ponto de máximo de $-1{,}4373x10^{-4}mm$ e um ponto de mínimo de $-1{,}9504x10^{-4}mm$ e para a variável Y14, o ponto de máximo foi de $9{,}6355x10^{-5}mm$ e um o ponto de mínimo de $-9{,}6474x10^{-5}mm$ ou seja, erros muito pequenos, visto que a precisão para execução do método de Newton Raphson foi de 10^{-3}.

O Gráfico 3, mostra se houve variação na angulação do corpo 14, enquanto ele realizava essa variação de altura.

Gráfico 3 - Orientação do corpo 14



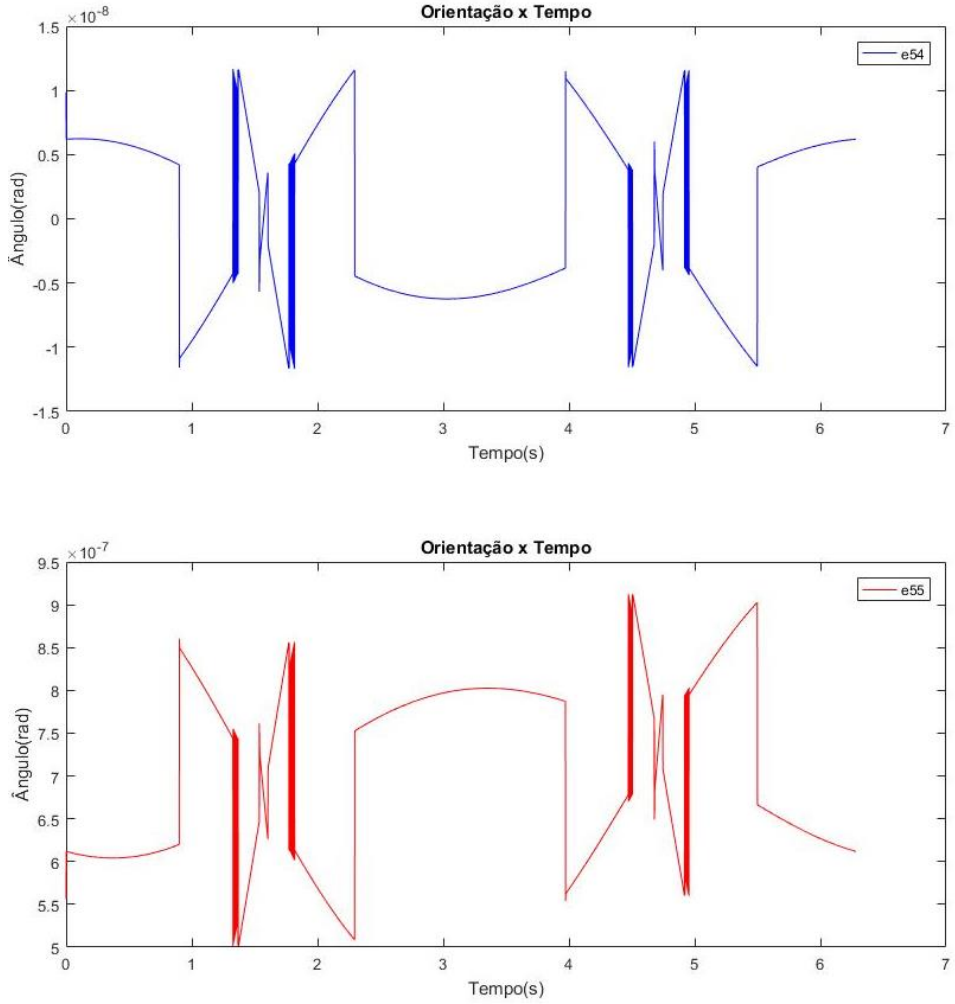
Fonte: Autoria Própria (2019)

Como pode ser visto, o corpo 14 da plataforma não sofre nenhuma variação angular significativa, durante seu movimento. Porém, ao se analisar com mais

precisão pode-se notar que há sim uma variação angular, nos parâmetros e54 e e55.

Mas, devido elas serem muito pequenas, a plotagem delas não é feita, por questões de escala do Gráfico 3 que gera sua escala com base nos maiores resultados, sendo necessário gerar um novo gráfico para visualizar tais. No Gráfico 4, elas são apresentadas.

Gráfico 4 - Comportamento das variáveis e54 e e55



Fonte: Autoria Própria (2019)

Constatando-se assim que o comportamento delas é similar ao da variável X14 e Y14, porém em uma escala bem menor. A justificativa dos mesmos é a mesma explicada a anteriormente, para as variáveis X14 e Y14, erros de arredondamento nos dados de entrada.

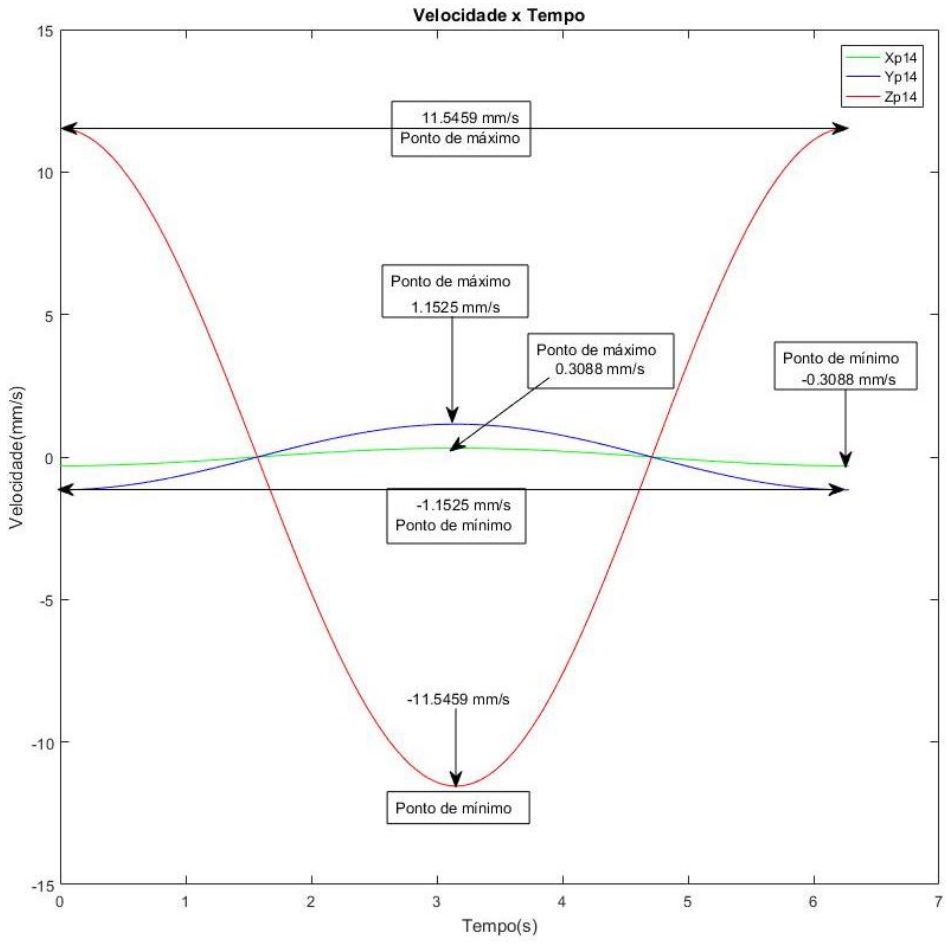
A magnitude desses erros também foi calculada, onde a variável e54 possui um ponto de máximo de $1{,}1674x10^{-8}rad$ e um ponto de mínimo de

$-1{,}1690x10^{-8}rad$. Para a variável e55 o ponto de máximo foi de $9{,}1243x10^{-7}rad$ e o ponto de mínimo foi de $5{,}0129x10^{-7}rad$.

Dessa maneira, ao analisar os deslocamentos e orientações do corpo 14 durante o intervalo de tempo proposto, a análise de posição do mecanismo está completa e assim a análise de velocidade pode ser iniciada.

A partir do Gráfico 5, pode ser visto a variação de velocidade linear para que tal deslocamento fosse realizado.

Gráfico 5 - Variação de Velocidade linear do corpo 14



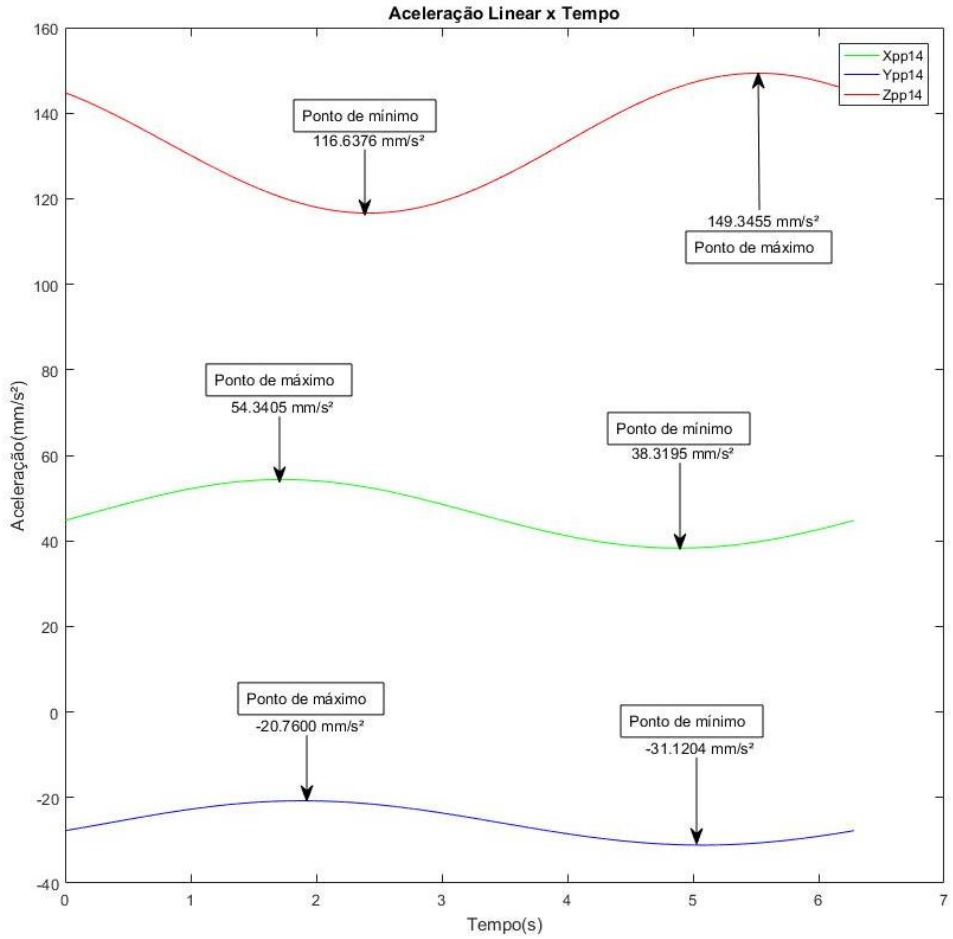
Fonte: Autoria Própria (2019)

No Gráfico 5, pode-se perceber que a variação de velocidade da variável Zp14 ou no eixo Z é a mais atuante, visto que o movimento ocorre de forma mais insinuante no mesmo. As baixas velocidades nos eixos X e Y, são contrastes dos pequenos movimentos realizados pelos mesmos apresentados nos gráficos anteriores.

Também é possível notar que os valores de obtidos pelas variáveis são razoáveis, ou seja, dentro de condições normais a plataforma se comporta de maneira correta.

No mais, as variáveis de velocidade do corpo 14 foram todas expostas nessa análise de velocidade e assim, a última parte da análise pode ser feita, que é a de aceleração, vinda a seguir no Gráfico 6.

Gráfico 6 - Aceleração Linear Corpo 14



Fonte: Autoria Própria (2019)

No gráfico apresentado é possível notar uma maior intensidade na variação da aceleração linear o eixo Z do corpo 14 ou em outras palavras da variável Z14. Obtendo os valores condizentes com as outras análises, visto que o movimento é predominante nessa direção.

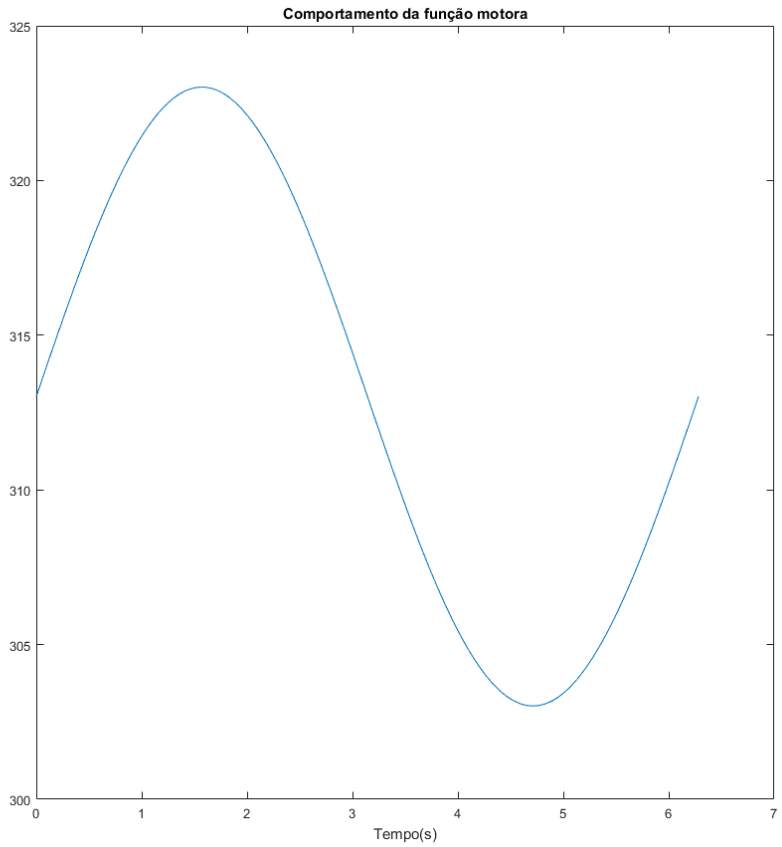
Os valores de aceleração para as outras duas variáveis estão ligadas aos movimentos de baixa escala explicados anteriormente, que por serem resultados

de erros de arredondamentos podem ter interferido de forma significativa nesse resultado.

Com isso, os módulos dos valores obtidos também estão coerentes, visto que os deslocamentos ocorrem de maneira suave, sem mudanças bruscas de direção ou altas inclinações, graficamente falando.

Uma forma de verificar a coerência da análise cinemática é comparando o comportamento das variáveis com a função motora escolhida. O movimento ondular apresentado pelas variáveis analisadas é gerado devido a função utilizada $l_0 + 10 * \sin(t)$. Que pode ser vista no Gráfico 7 a seguir.

Gráfico 7 - Comportamento da Função Motora



Fonte: Autoria Própria (2019)

Essa função também representa um ciclo de atuação da plataforma de forma completa, pois seus picos e vales indicam uma expansão e compressão dos atuadores, demonstrando o funcionamento correto dos mesmos.

Dessa maneira, a análise cinemática está completa, apresentando os valores assumidos pelas coordenadas generalizadas que representam as

variações de posição, orientação, velocidades e acelerações do corpo 14 ou a base móvel da plataforma.

Os scripts e funções para a realização da mesma estão no Apêndice A e o programa feito no software MATLAB levou um tempo de 231,881741 segundos para ser executado em um processador Intel Core i-5 6300HQ, com 16 GB de RAM DDR3L e uma GTX 960M 4 GB.

6. CONSIDERAÇÕES FINAIS

Com o conteúdo obtido das referências consultadas foi possível elaborar uma estrutura satisfatória de informações que foram usadas para introduzir, explicar e aplicar as teorias envolvidas em uma análise cinemática para um projeto de um mecanismo especifico, podendo assim obter as informações de posição, velocidade e aceleração do mesmo.

A modelagem das restrições, dos Jacobianos, das funções Nu, Gama e outras, tiveram a sua aplicação de forma satisfatória, mostrando que sua modelagem por meio dos conceitos de dinâmica de multicorpos foi feito de maneira coerente, permitindo a plataforma funcionar sem nenhum erro de singularidade.

Os dados referentes ao movimento da plataforma de Stewart mostraram pelos resultados da análise cinemática serem coerentes, ao comportamento da função motora escolhida para reger o movimento da plataforma, visto que obtiveram um movimento semelhante a mesma. Mostrando assim que as escolhas realizadas na fase de projeto são justificáveis, como por exemplo a escolha das juntas presente na plataforma.

Portanto, os resultados obtidos podem ser considerados suficientes e satisfatórios para a proposta do presente trabalho, pois de acordo com as informações obtidas da análise cinemática, é possível constatar o correto funcionamento do modelo da plataforma de Stewart, justificando os procedimentos de modelagem como o posicionamento dos sistema de coordenadas, a medição das orientações dos vetores unitários, bem como o cálculo das orientações inicias.

REFERÊNCIAS

BAUCHAU, O. A.. Flexible Multibody Dynamics. Atlanta (eua): Springer, 2011. 732p.

CAMPOS FILHO, Frederico Ferreira. ALGORITMOS NUMÉRICOS. 2. ed. Belo Horizonte: Ltc, 2007. 423 p.

CHAPRA, Steven C.; CANALE, Raymond P.. Numerical Methods for Engineers. 17. ed. New York (eua): Mcgraw-hill, 2015. 957 p

CHIN, Cheng Siong. COMPUTER-AIDED CONTROL SYSTEMS DESIGN: Pratical Applications Using MATLAB and Simulink. New York (eua): Crc, 2012. 357 p.

DAMIC, Vjekoslav; COHODAR, Maida. **Dynamic Analysis of Stewart Platform by Bond Graphs.** 2014. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877705815003896>. Acesso em: 27 dez. 2017.

DASGUPTA, Bhaskar; MRUTHYUNJAYA, T.s.. The Stewart platform manipulator: a review. **Mechanism And Machine Theory.** Guimarães, p. 15-40. 18 dez. 1998.

DUPLESSIS, Lukas Johannes. **DESIGN AND OPTIMUM OPERATION OF A RE-CONFIGURABLE PLANAR GOUGH-STEWART MACHINING PLATFORM.** 2001. 290 f. Tese (Doutorado) - Curso de Engenharia Mecânica, Universidade de Pretoria, Pretoria, 2001.

FLEMMING, Diva Marília; GONÇALVES, Mirian Buss. **Cálculo B:** Funções de várias varáveis, integrais múltiplas, integrais curvilíneas e de superfície. 2. ed. São Paulo: Pearson, 1999. 435 p.

GILAT, Amos. MATLAB: An Introduction With Applications. Danvers: John Wiley & Sons, 2004. 293 p.

HALLIDAY, David; RESNICK, Rpbert; WALKER, Jead. **Fundamentos da Física.** 8. ed. Rio de Janeiro: Ltc, 2008.

HAUG, Edward J.. **COMPUTE AIDED KINEMATICS AND DYNAMICS OF MECHANICAL SYSTEMS:** VOLUME I: BASIC MEHODS. Iowa: Allyn An Bacon, 1989. 1 v.

HAY, Alexander Morrison. **METHODS FOR THE DETERMINATION OF ACCESIBLE WORKSPACES OF PLANAR STEWART PLATFORMS OF GENERAL DESING.** 1999. 92 f. Dissertação (Mestrado) - Curso de Engenharia Mecânica, Universidade de Pretoria, Pretoria, 1999.

HIBBELER, R. C.. **Engenharia mecânica: Dinâmica.** 12. ed. São Paulo: Pearson Prentice Hall, 2011. 591 p.

HUNT, Brian R.; LIPSMAN, Ronald L.; ROSENBERD, Jonathan M.. A Guide to MATLAB: for Beginners and Experienced User. New York (eua): Cambridge University Press, 2001. 317 p.

JOSEPHS, Harold; HUSTON, Ronal L.. **DYNAMICS of MECHANICAL SYSTEMS.** Boca Raton: Crc Press, 2002. 777 p.

LUCIDCHART. 2018. Disponível em: <https://www.lucidchart.com>. Acesso em: 14 jan. 2018.

MABIE, Hamilton H.. **Mecanismos.** 2. ed. Rio de Janeiro: Sindicato Nacional dos Editores de Livros, 1980. 304 p.

MERIAM, J.l.; KRAIGE, L.g.. **Engineering Mechanics:** Dynamics. 6. ed. Eua: Wiley, 2006. 744 p.

MIKROLAR. **Universal Tyre-Testing Machine.** Disponível em: <http://mikrolar.com/hexapod.html>. Acesso em: 27 dez. 2017.

NEGRUT, Dan. **ME451.** 1ed. University of Wisconsin-Madison, 2014. 560p.

NORTON, Robert L.. **Cinemática e dinâmica dos mecanimos.** Porto Alegre: Mcgraw-hill, 2009. 812 p.

NORTON, Robert L.. **Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines.** 5. ed. New York: Mcgraw-hill, 2012. 857 p.

NIKRAVESH, Parvis E., **Computer-Aided Analysis of Mechanical Systems**, PrenticeHall, 1988.

SHABANA, Ahmed A.. **Computacional Dynamics:** Second Edition. Danvers: John Wiley & Sons, 2001. 522 p.

SMIT, Willem Jacobus. **The optimal design of a planar Stewart platform for prescribed machining tasks.** 2000. 98 f. Dissertação (Mestrado) - Curso de Engenharia Mecânica, Departamento de Engenharia Mecânica e Aeronautica, Universidade de Pretoria, Pretoria, 2000.

TECHNOLOGIES, Simge Simulataion. **6-DOF Motion Platform.** Disponível em: <http://www.simgesimulasyon.com/motion-platforms/>. Acesso em: 27 dez. 2017.

Zafer BINGUL and Oguzhan KARAHAN (2012). Dynamic Modeling and Simulation of Stewart Platform, Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization, Dr. Serdar Kucuk (Ed.), InTech, DOI: 10.5772/32470. Available from: https://www.intechopen.com/books/serial-and-parallel-robot-manipulators-kinematics-dynamics-control-and-optimization/dynamic-modelling-of-stewart-platform.

WIJCKMANS, P. M. E. J. (1996). Conditioning of differential algebraic equations and numerical solution of multibody dynamics Eindhoven: Technische Universiteit Eindhoven DOI: 10.6100/IR458459.

APÊNDICE A – Códigos e funções para a análise cinemática

Código Principal

```
close all
clear all
clc
%Chute Inicial
qo = getInitialGuess;
%Variação do Tempo
time = 0:1E-3:2*pi;
tic
%Análise de Posição
out = PositionAnalysis(qo,time);
%Análise de Velocidade
out_t = VelocityAnalysis(out,time);
%Análise de Aceleração
out_tt = AcelerationAnalysis(out,out_t,time);

%Plotagem dos resultados
figure('Name','"Posições" gráficas')

plot(time,out(92,:),'g')
hold on
 plot(time,out(93,:),'b')
hold on
plot(time,out(94,:),'r')
hold on
legend('X14','Y14','Z14')
title('Posição x Tempo')
 xlabel('Tempo(s)')
 ylabel('Posição(mm)')

figure('Name','Comportamento das variáveis X14 e Y14')
subplot(2,1,1)
plot(time,out(92,:),'g')
legend('X14')
title('Comportamento da Variável X14')
xlabel('Tempo(s)')
ylabel('Posição(mm)')
subplot(2,1,2)
plot(time,out(93,:),'b')
legend('Y14')
title('Comportamento da Variável Y14')
xlabel('Tempo(s)')
ylabel('Posição(mm)')

figure('Name','Velocidades Lineares')
plot(time,out_t(92,:),'g')
hold on
plot(time,out_t(93,:),'b')
hold on
plot(time,out_t(94,:),'r')
hold on
legend('Xp14','Yp14','Zp14')
title('Velocidade x Tempo')
xlabel('Tempo(s)')
ylabel('Velocidade(mm/s)')

figure('Name','Acelerações Lineares')
```

```
plot(time,out_tt(92,:),'g')
hold on
plot(time,out_tt(93,:),'b')
hold on
plot(time,out_tt(94,:),'r')
hold on
legend('Xpp14','Ypp14','Zpp14')
title('Aceleração Linear x Tempo')
xlabel('Tempo(s)')
ylabel('Aceleração(mm/s²)')


figure('Name','"Orietações" gráficas')

plot(time,out(95,:),'g')
hold on
plot(time,out(96,:),'b')
hold on
plot(time,out(97,:),'r')
hold on
plot(time,out(98,:),'y')
legend('e53','e54','e55','e56')
title('Orientação x Tempo')
xlabel('Tempo(s)')
ylabel('Ângulo(rad)')

figure('Name','Comportamento das variáveis e54 e e55')
subplot(2,1,1)
plot(time,out(96,:),'b')
legend('e54')
title('Orientação x Tempo')
xlabel('Tempo(s)')
ylabel('Ângulo(rad)')
subplot(2,1,2)
plot(time,out(97,:),'r')
legend('e55')
title('Orientação x Tempo')
xlabel('Tempo(s)')
ylabel('Ângulo(rad)')

toc
```

Análise de Posição

```
function out = PositionAnalysis(q,t)
out = zeros(length(q),length(t));
% q = qo;
%
for i=1:length(t)
 for ii = 1:50
   phi = getPhi(q,t(i));
   J = getJacobian(q,t(i));
   dx = -J\phi;
   q = q + dx;
   if norm(dx)<1E-3
    break
   end
 end
%Armazenamento do loop
```

```
out(:,i) = q;
end

end
```

Restrições

```
function phi = getPhi(q,t)
%% Constraint Modeling
%Load bodies parameters
B = BPar();
%% Fixed Joint Constraint(Parcial:06 equations, Global: 06 equations)
FixCons = [...
           q(1:3);...
           q(5:7)];
%% Spherical Joint Constraints(Parcial: 18 equations, Global: 24 equations)
%y=Si(ri,Pi,sri,rj,Pj,srj);
%                  ri       Pi       sri      rj       Pj        srj
 SphCons =  [Si(q(15:17),q(18:21),B(03).sr,q(92:94),q(95:98),B(14).sr1
);...
             Si(q(29:31),q(32:35),B(05).sr,q(92:94),q(95:98),B(14).sr2
);...
             Si(q(43:45),q(46:49),B(07).sr,q(92:94),q(95:98),B(14).sr3
);...
             Si(q(57:59),q(60:63),B(09).sr,q(92:94),q(95:98),B(14).sr4
);...
             Si(q(71:73),q(74:77),B(11).sr,q(92:94),q(95:98),B(14).sr5
);...
             Si(q(85:87),q(88:91),B(13).sr,q(92:94),q(95:98),B(14).sr6 )];
%% Slider Joint Constraints(Parcial: 30 Equations, Global: 54 equations)
%y=Transl(ri,si,pi,fi,gi,rj,sj,pj,fj,hj);
i = 1:7:98;
ii = 4:7:98;
%                          ri              si           pi               fi
gi              rj,             sj,          pj,            fj,
hj
  TraCons = [...
             Transl(q(i(02):i(02)+2), B(02).sq, q(ii(02):ii(02)+3),
B(02).Ad(:,1), B(02).Ad(:,2), q(i(03):i(03)+2), B(03).sq,
q(ii(03):ii(03)+3), B(03).Ad(:,1), B(03).Ad(:,3));...
             Transl(q(i(04):i(04)+2), B(04).sq, q(ii(04):ii(04)+3),
B(04).Ad(:,1), B(04).Ad(:,2), q(i(05):i(05)+2), B(05).sq,
q(ii(05):ii(05)+3), B(05).Ad(:,1), B(05).Ad(:,3));...
             Transl(q(i(06):i(06)+2), B(06).sq, q(ii(06):ii(06)+3),
B(06).Ad(:,1), B(06).Ad(:,2), q(i(07):i(07)+2), B(07).sq,
q(ii(07):ii(07)+3), B(07).Ad(:,1), B(07).Ad(:,3));...
             Transl(q(i(08):i(08)+2), B(08).sq, q(ii(08):ii(08)+3),
B(08).Ad(:,1), B(08).Ad(:,2), q(i(09):i(09)+2), B(09).sq,
q(ii(09):ii(09)+3), B(09).Ad(:,1), B(09).Ad(:,3));...
             Transl(q(i(10):i(10)+2), B(10).sq, q(ii(10):ii(10)+3),
B(10).Ad(:,1), B(10).Ad(:,2), q(i(11):i(11)+2), B(11).sq,
q(ii(11):ii(11)+3), B(11).Ad(:,1), B(11).Ad(:,3));...
             Transl(q(i(12):i(12)+2), B(12).sq, q(ii(12):ii(12)+3),
B(12).Ad(:,1), B(12).Ad(:,2), q(i(13):i(13)+2), B(13).sq,
q(ii(13):ii(13)+3), B(13).Ad(:,1), B(13).Ad(:,3))];
%% Universal Joint Constraints(Parcial:24 Equations, Global:78 Equations)
%y=Uni(rj,pj,spj,ri,pi,spi,hj,hi);
j = 1:7:98;
jj = 4:7:98;
%                   ri,     pi,    spi,      hi,                 rj,
pj,           spj,        hj
 UniCons = [...
```

```
            Uni(q(1:3), q(4:7), B(1).sp1, B(1).Au1(:,3), q(j(02):j(02)+2),
q(jj(02):jj(02)+3), B(02).sp, B(02).Au(:,3));...
            Uni(q(1:3), q(4:7), B(1).sp2, B(1).Au2(:,3), q(j(04):j(04)+2),
q(jj(04):jj(04)+3), B(04).sp, B(04).Au(:,3));...
            Uni(q(1:3), q(4:7), B(1).sp3, B(1).Au3(:,3), q(j(06):j(06)+2),
q(jj(06):jj(06)+3), B(06).sp, B(06).Au(:,3));...
            Uni(q(1:3), q(4:7), B(1).sp4, B(1).Au4(:,3), q(j(08):j(08)+2),
q(jj(08):jj(08)+3), B(08).sp, B(08).Au(:,3));...
            Uni(q(1:3), q(4:7), B(1).sp5, B(1).Au5(:,3), q(j(10):j(10)+2),
q(jj(10):jj(10)+3), B(10).sp, B(10).Au(:,3));...
            Uni(q(1:3), q(4:7), B(1).sp6, B(1).Au6(:,3), q(j(12):j(12)+2),
q(jj(12):jj(12)+3), B(12).sp, B(12).Au(:,3));...
            ];

 %% Driver Constraints(Parcial: 6 Equations, Global: 84 Equations )
 %y = Mot(ri,pi,si,hi,rj,pj,sj,t)
 %                    ri,                pi,            si,         hi,
rj,                pj,             sj,  time
 DriCons = [Mot(q(i(02):i(02)+2), q(ii(02):ii(02)+3), B(02).sq,
B(02).Ad(:,3),q(i(03):i(03)+2), q(ii(03):ii(03)+3),B(03).sq,  t) ;...
            Mot(q(i(04):i(04)+2), q(ii(04):ii(04)+3), B(04).sq,
B(04).Ad(:,3),q(i(05):i(05)+2), q(ii(05):ii(05)+3),B(05).sq,  t);...
            Mot(q(i(06):i(06)+2), q(ii(06):ii(06)+3), B(06).sq,
B(06).Ad(:,3),q(i(07):i(07)+2), q(ii(07):ii(07)+3),B(07).sq,  t);...
            Mot(q(i(08):i(08)+2), q(ii(08):ii(08)+3), B(08).sq,
B(08).Ad(:,3),q(i(09):i(09)+2), q(ii(09):ii(09)+3),B(09).sq,  t);...
            Mot(q(i(10):i(10)+2), q(ii(10):ii(10)+3), B(10).sq,
B(10).Ad(:,3),q(i(11):i(11)+2), q(ii(11):ii(11)+3),B(11).sq,  t);...
            Mot(q(i(12):i(12)+2), q(ii(12):ii(12)+3), B(12).sq,
B(12).Ad(:,3),q(i(13):i(13)+2), q(ii(13):ii(13)+3),B(13).sq,  t)];
 %% Euler Constraints(Parcial: 14 Equations, Global: 92 Equations )
 %y = Euler(p)
 j = 4:7:98;
 jj = j+3;
 EulerCons  = [Euler(q(j(1):jj(1)));...
               Euler(q(j(2):jj(2)));...
               Euler(q(j(3):jj(3)));...
               Euler(q(j(4):jj(4)));...
               Euler(q(j(5):jj(5)));...
               Euler(q(j(6):jj(6)));...
               Euler(q(j(7):jj(7)));...
               Euler(q(j(8):jj(8)));...
               Euler(q(j(9):jj(9)));...
               Euler(q(j(10):jj(10)));...
               Euler(q(j(11):jj(11)));...
               Euler(q(j(12):jj(12)));...
               Euler(q(j(13):jj(13)));...
               Euler(q(j(14):jj(14)));...
               ];
%% Constraint Equation Assembler
phi = [...
       FixCons;...%6,6
       SphCons;...%18,24
       TraCons;...%30,54
       UniCons;...%24,78
       DriCons;...% 6,84
     EulerCons;...%14,98
       ];


end
```

Armazenamento dos dados da plataforma

```
function BDs = BPar()

BDs(1).A = [1 0 0;0 1 0;0 0 1];
BDs(1).p = eulerParameters(BDs(1).A);
BDs(1).r = [0;0;0];
%Universal joints coordinate systems spj,  joint j, in relation to body 1
BDs(1).sp1 = [181.774273898;228.521581801;0.00000];
Px11 = [182.451179941;229.126602212;0.419223701];
Py11 = [181.736707223;227.981176778;0.840565974];
Pz11 = [182.509384063;227.936848773;-0.343074820];
BDs(1).sp2 = [107.018358203;271.681929851;0.0000];
Px12 = [107.203868228;272.570657885;0.419223701];
Py12 = [106.569137063;271.379193644;0.840565974];
Pz12 = [107.892306942;271.337672287;0.343074820];
BDs(1).sp3 = [-288.792632102;43.160348050;0.0000];
Px13 = [-289.655048169;43.444055673;0.419223701];
Py13 = [-288.305844286;43.398016866;0.840565974];
Pz13 = [-288.653793528;44.089338641;-0.343074820];
BDs(1).sp4 = [-288.792632102;-43.160348050;0.000];
Px14 = [-289.655048169;-43.444055673;0.419223701];
Py14 = [-288.305844286;-43.398016866;0.840565974];
Pz14 = [-288.931470676;-42.231357459;0.343074820];
BDs(1).sp5 = [107.018358203;-271.681929851;0];
Px15 = [107.203868228;-272.570657885;0.419223701];
Py15 = [106.569137063;-271.379193644;0.840565974];
Pz15 = [106.144409465;-272.026187414;-0.343074820];
BDs(1).sp6 = [181.774273898;-228.521581801;0];
Px16 = [182.451179941;-229.126602212;0.419223701];
Py16 = [181.736707223;-227.981176778;0.840565974];
Pz16 = [181.039163733;-229.106314829;0.343074820];
%Universal joints coordinate systems: Defined by matrix Auj, joint j, in
relation to
%body 1. Each colunm on matrix Aij = [f' g' h'] is a unit vector forming a
%triad to define coordinate systems.
BDs(1).Au1 = [Px11-BDs(1).sp1,Py11-BDs(1).sp1,Pz11-BDs(1).sp1];
BDs(1).Au2 = [Px12-BDs(1).sp2,Py12-BDs(1).sp2,Pz12-BDs(1).sp2];
BDs(1).Au3 = [Px13-BDs(1).sp3,Py13-BDs(1).sp3,Pz13-BDs(1).sp3];
BDs(1).Au4 = [Px14-BDs(1).sp4,Py14-BDs(1).sp4,Pz14-BDs(1).sp4];
BDs(1).Au5 = [Px15-BDs(1).sp5,Py15-BDs(1).sp5,Pz15-BDs(1).sp5];
BDs(1).Au6 = [Px16-BDs(1).sp6,Py16-BDs(1).sp6,Pz16-BDs(1).sp6];



%%
 O2 = [ 177.528522466;167.445498832;95.000000000];
 Px2 = [178.205428508;168.050519243;95.419223701];
 Py2 = [176.793412301;168.030231859;95.343074820];
 Pz2 = [177.490955790;166.905093809;95.840565974];
BDs(2).A = [Px2-O2,Py2-O2,Pz2-O2];
BDs(2).p = eulerParameters(BDs(2).A);
BDs(2).r = [177.528;167.445;95.000];
BDs(2).sp = [0;0;-3.390572648458574e+02/3];
BDs(2).sq = [0;0;-100.000];
BDs(2).Au = [ 0, 0,-1;...
              0, 1, 0;...
              1, 0, 0];
BDs(2).Ad = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(2).rp = [0;0;0];
```

```
%%
BDs(3).A = [Px2-O2,Py2-O2,Pz2-O2];
BDs(3).p = eulerParameters(BDs(3).A);
BDs(3).r = [173.283;106.369;190.000];
BDs(3).sq = [0;0;100];
BDs(3).sr = [0;0;3.390572648458574e+02/3];
BDs(3).Ad = [ 0,-1, 0;...
             -1, 0, 0;...
              0, 0,-1];
BDs(3).As = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(3).rp = [0;0;0];
%%
%Body 4
O4 = [ 56.247794505;237.466959767;95.000000000];
 Px4 = [56.433304529;238.355687802;95.419223701];
 Py4 = [55.373845766;237.811217331;94.656925180];
 Pz4 = [55.798573365;237.164223561;95.840565974];
BDs(4).A = [Px4-O4,Py4-O4,Pz4-O4];
BDs(4).p = eulerParameters(BDs(4).A);
BDs(4).r = [56.248;237.467;95.000];
BDs(4).sp = [0;0;-3.390572648458574e+02/3];
BDs(4).sq = [0;0;-100.000];
BDs(4).Au = BDs(2).Au ;
BDs(4).Ad = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(4).rp = [0;0;0];
%%
BDs(5).A = [Px4-O4,Py4-O4,Pz4-O4];
BDs(5).p = eulerParameters(BDs(5).A);
BDs(5).r = [5.477;203.252;190.000];
BDs(5).sq = [0;0;100.000];
BDs(5).sr = [0;0;3.390572648458574e+02/3];
BDs(5).Ad = [ 0,-1, 0;...
             -1, 0, 0;...
              0, 0,-1];
BDs(5).As = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(5).As = [1 0 0; 0 1 0; 0 0 1];
BDs(5).rp = [0;0;0];
%%
 %Body 6
 O6 = [ -233.776316970;70.021460936;95.000000000];
 Px6 = [-234.638733037;70.305168559;95.419223701];
 Py6 = [-233.915155544;69.092470345;95.343074820];
 Pz6 = [-233.289529155;70.259129752;95.840565974];

BDs(6).A = [Px6-O6,Py6-O6,Pz6-O6];
BDs(6).p = eulerParameters(BDs(6).A);
BDs(6).r = [-233.776;70.021;95.000];
BDs(6).sp = [0;0;-3.390572648458574e+02/3];
BDs(6).sq = [0;0;-100.000];
BDs(6).Au = BDs(2).Au;
BDs(6).Ad = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(6).rp = [0;0;0];
%%
```

```
BDs(7).A = [Px6-O6,Py6-O6,Pz6-O6];
BDs(7).p = eulerParameters(BDs(7).A);
BDs(7).r = [-178.760;96.882;190.000];
BDs(7).sq = [0;0;100];
BDs(7).sr = [0;0;3.390572648458574e+02/3];
BDs(7).Ad = [ 0,-1, 0;...
             -1, 0, 0;...
              0, 0,-1];
BDs(7).As = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(7).rp = [0;0;0];
%%
 %Body 8
O8 = [ -233.776316970;-70.021460936;95.000000000];
 Px8 = [ -234.638733037; -70.305168559; 95.419223701];
 Py8 = [-233.637478396;-70.950451527;94.656925180];
 Pz8 = [-233.289529155;-70.259129752; 95.840565974];

BDs(8).A = [Px8-O8,Py8-O8,Pz8-O8];
BDs(8).p = eulerParameters(BDs(8).A);
BDs(8).r = [-233.776;-70.021;95.000];
BDs(8).sp = [0;0;-3.390572648458574e+02/3];
BDs(8).sq = [0;0;-100.000];
BDs(8).Au = BDs(2).Au ;
BDs(8).Ad = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(8).rp = [0;0;0];
%%
BDs(9).A = [Px8-O8,Py8-O8,Pz8-O8];
BDs(9).p = eulerParameters(BDs(9).A);
BDs(9).r = [-178.760;-96.882;190.000];
BDs(9).sq = [0;0;100];
BDs(9).sr = [0;0;3.390572648458574e+02/3];
BDs(9).Ad = [ 0,-1, 0;...
             -1, 0, 0;...
              0, 0,-1];
BDs(9).As = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(9).rp = [0;0;0];
%%
%Body 10
O10 = [ 56.247794505;-237.466959767;95.000000000];

 Px10 = [56.433304529;-238.355687802;95.419223701];
 Py10 = [57.121743243;-237.122702204;95.343074820];
 Pz10 = [ 55.798573365;-237.164223561; 95.840565974];
BDs(10).A = [Px10-O10,Py10-O10,Pz10-O10];
BDs(10).p = eulerParameters(BDs(10).A);
BDs(10).r = [56.248;-237.467;95.000];
BDs(10).sp = [0;0;-3.390572648458574e+02/3];
BDs(10).sq = [0;0;-100.0];
BDs(10).Au = BDs(2).Au;
BDs(10).Ad = [1 0 0;...
             0 1 0;...
             0 0 1];
BDs(10).rp = [0;0;0];
%%
BDs(11).A = [Px10-O10,Py10-O10,Pz10-O10];
```

```
BDs(11).p = eulerParameters(BDs(11).A);
BDs(11).r = [5.477;-203.252;190.000];
BDs(11).sq = [0;0;100];
BDs(11).sr = [0;0;3.390572648458574e+02/3];
BDs(11).Ad = [ 0,-1, 0;...
              -1, 0, 0;...
               0, 0,-1];
BDs(11).As = [1 0 0;...
              0 1 0;...
              0 0 1];
BDs(11).rp = [0;0;0];
%%
 %Body 12
 O12 = [ 177.528522466; -167.445498832; 95.000000000];
 Px12 = [ 178.205428508;-168.050519243; 95.419223701];
 Py12 = [ 178.263632630;-166.860765804;94.656925180 ];
 Pz12 = [ 177.490955790;-166.905093809; 95.840565974];
BDs(12).A = [Px12-O12,Py12-O12,Pz12-O12];
BDs(12).p = eulerParameters(BDs(12).A);
BDs(12).r = [177.528;-167.445;95.000];
BDs(12).sp = [0;0;-3.390572648458574e+02/3];
BDs(12).sq = [0;0;-100.000];
BDs(12).Au = BDs(2).Au;
BDs(12).Ad = [1 0 0;...
              0 1 0;...
              0 0 1];
BDs(12).rp = [0;0;0];
%%
BDs(13).A = [Px12-O12,Py12-O12,Pz12-O12];
BDs(13).p = eulerParameters(BDs(13).A);
BDs(13).r = [173.283;-106.369;190.000];
BDs(13).sq = [0;0;100];
BDs(13).sr = [0;0;3.390572648458574e+02/3];
BDs(13).Ad = [ 0,-1, 0;...
              -1, 0, 0;...
               0, 0,-1];
BDs(13).As = [1 0 0;...
              0 1 0;...
              0 0 1];
BDs(13).rp = [0;0;0];
%%
BDs(14).A = [1 0 0;0 1 0;0 0 1];
BDs(14).p = eulerParameters(BDs(14).A);
BDs(14).r = [0;0;285];
BDs(14).sr1 = [169.037;45.293;0];
BDs(14).sr2 = [-45.293;169.037;0];
BDs(14).sr3 = [-123.744;123.744;0];
BDs(14).sr4 = [-123.744;-123.744;0];
BDs(14).sr5 = [-45.293;-169.037;0];
BDs(14).sr6 = [169.037;-45.293;0];
End
```

Modelagem dos Parâmetros de Euler

```
function p = eulerParameters(A)
%
p(1,:) = sqrt((trace(A)+1)/4);
p(2,:) = (A(3,2)-A(2,3))/(4*p(1,:));
p(3,:) = (A(1,3)-A(3,1))/(4*p(1,:));
p(4,:) = (A(2,1)-A(1,2))/(4*p(1,:));
end
```

Funções das restrições

```
%Função de Restrição Junta Esferica
function y = Si(ri,pi,sri,rj,pj,srj)
y = rj + getA(pj)*srj - ri - getA(pi)*sri;
end

%Função de Restrição Junta Prismatica
function y = Transl (ri,si,pi,fi,gi,rj,sj,pj,fj,hj)
d = dij(ri,si,pi,rj,sj,pj);

y = [fi'*getA(pi).'*getA(pj)*hj;...
     gi'*getA(pi).'*getA(pj)*hj;...
     fi'*getA(pi).'*d;...
     gi'*getA(pi).'*d;...
     fi'*getA(pi).'*getA(pj)*fj];

end

%Função de Restrição Junta Universal
function y  = Uni(ri,pi,spi,hi,rj,pj,spj,hj)

y = [rj + getA(pj)*spj - ri - getA(pi)*spi;...
     hi'*getA(pi).'*getA(pj)*hj];
end

%Função de Restrição Junta Motora
function y = Mot(ri,pi,si,hi,rj,pj,sj,t)
d = dij(ri,si,pi,rj,sj,pj);
y = (hi'*getA(pi).'*(d) - (313.0191+10*sin(t)));
end

%Função de Restrição Parametros de Euler
function y = Euler(p)
 y = p.'*p - 1;
end
```

Matriz Jacobiana

```
function j = getJacobian(q,t)
%% Load Body Parameters
B = BPar();
i = 1:7:98;
ii = 4:7:98;
JFi1 = [eye(3),zeros(3,4);...%corpo 1, na posição linha 1:6, coluna 1:7
        zeros(3,4),eye(3)];
%% Spherical Joints - Constraints
JSi1 = JSi(B(03).sr,q(ii(03):ii(03)+3));JSj1 =
JSj(B(14).sr1,q(ii(14):ii(14)+3));
JSi2 = JSi(B(05).sr,q(ii(05):ii(05)+3));JSj2 =
JSj(B(14).sr2,q(ii(14):ii(14)+3));
JSi3 = JSi(B(07).sr,q(ii(07):ii(07)+3));JSj3 =
JSj(B(14).sr3,q(ii(14):ii(14)+3));
JSi4 = JSi(B(09).sr,q(ii(09):ii(09)+3));JSj4 =
JSj(B(14).sr4,q(ii(14):ii(14)+3));
JSi5 = JSi(B(11).sr,q(ii(11):ii(11)+3));JSj5 =
JSj(B(14).sr5,q(ii(14):ii(14)+3));
```

```
JSi6 = JSi(B(13).sr,q(ii(13):ii(13)+3));JSj6 =
JSj(B(14).sr6,q(ii(14):ii(14)+3));
%% Slider Joints - Constraints
%Body i
JTi1 =  JTi( q(i(2):i(2)+2),  B(2).sq,  q(ii(2):ii(2)+3),  B(2).Ad(:,1),
B(2).Ad(:,2),  q(i(3):i(3)+2),  B(3).sq, q(ii(3):ii(3)+3),  B(3).Ad(:,1),
B(3).Ad(:,3));
JTi2 =  JTi( q(i(4):i(4)+2),  B(4).sq,  q(ii(4):ii(4)+3),  B(4).Ad(:,1),
B(4).Ad(:,2),  q(i(5):i(5)+2),  B(5).sq, q(ii(5):ii(5)+3),  B(5).Ad(:,1),
B(5).Ad(:,3));
JTi3 =  JTi( q(i(6):i(6)+2),  B(6).sq,  q(ii(6):ii(6)+3),  B(6).Ad(:,1),
B(6).Ad(:,2),  q(i(7):i(7)+2),  B(7).sq, q(ii(7):ii(7)+3),  B(7).Ad(:,1),
B(7).Ad(:,3));
JTi4 =  JTi( q(i(8):i(8)+2),  B(8).sq,  q(ii(8):ii(8)+3),  B(8).Ad(:,1),
B(8).Ad(:,2),  q(i(9):i(9)+2),  B(9).sq, q(ii(9):ii(9)+3),  B(9).Ad(:,1),
B(9).Ad(:,3));
JTi5 = JTi( q(i(10):i(10)+2), B(10).sq, q(ii(10):ii(10)+3), B(10).Ad(:,1),
B(10).Ad(:,2), q(i(11):i(11)+2), B(11).sq,q(ii(11):ii(11)+3),
B(11).Ad(:,1), B(11).Ad(:,3));
JTi6 = JTi( q(i(12):i(12)+2), B(12).sq, q(ii(12):ii(12)+3), B(12).Ad(:,1),
B(12).Ad(:,2), q(i(13):i(13)+2), B(13).sq,q(ii(13):ii(13)+3),
B(13).Ad(:,1), B(13).Ad(:,3));
%Body j
JTj1 = JTj(   q(ii(2):ii(2)+3),  B(2).Ad(:,1),  B(2).Ad(:,2),
B(3).Ad(:,1),  B(3).Ad(:,3),  B(3).sq, q(ii(3):ii(3)+3));
JTj2 = JTj(   q(ii(4):ii(4)+3),  B(4).Ad(:,1),  B(4).Ad(:,2),
B(5).Ad(:,1),  B(5).Ad(:,3),  B(5).sq, q(ii(5):ii(5)+3));
JTj3 = JTj(   q(ii(6):ii(6)+3),  B(6).Ad(:,1),  B(6).Ad(:,2),
B(7).Ad(:,1),  B(7).Ad(:,3),  B(7).sq, q(ii(7):ii(7)+3));
JTj4 = JTj(   q(ii(8):ii(8)+3),  B(8).Ad(:,1),  B(8).Ad(:,2),
B(9).Ad(:,1),  B(9).Ad(:,3),  B(9).sq, q(ii(9):ii(9)+3));
JTj5 = JTj(  q(ii(10):ii(10)+3), B(10).Ad(:,1), B(10).Ad(:,2),
B(11).Ad(:,1), B(11).Ad(:,3), B(11).sq,q(ii(11):ii(11)+3));
JTj6 = JTj(  q(ii(12):ii(12)+3), B(12).Ad(:,1), B(12).Ad(:,2),
B(13).Ad(:,1), B(13).Ad(:,3), B(13).sq,q(ii(13):ii(13)+3));
%% Universal Joints - Constraints
JUi1 =
JUi(B(1).sp1,B(1).Au1(:,3),B(2).Au(:,3),q(ii(1):ii(1)+3),q(ii(2):ii(2)+3));
JUj1 =
JUj(B(2).sp,B(1).Au1(:,3),B(2).Au(:,3),q(ii(1):ii(1)+3),q(ii(2):ii(2)+3));
JUi2 =
JUi(B(1).sp2,B(1).Au2(:,3),B(4).Au(:,3),q(ii(1):ii(1)+3),q(ii(4):ii(4)+3));
JUj2 =
JUj(B(4).sp,B(1).Au2(:,3),B(4).Au(:,3),q(ii(1):ii(1)+3),q(ii(4):ii(4)+3));
JUi3 =
JUi(B(1).sp3,B(1).Au3(:,3),B(6).Au(:,3),q(ii(1):ii(1)+3),q(ii(6):ii(6)+3));
JUj3 =
JUj(B(6).sp,B(1).Au3(:,3),B(6).Au(:,3),q(ii(1):ii(1)+3),q(ii(6):ii(6)+3));
JUi4 =
JUi(B(1).sp4,B(1).Au4(:,3),B(8).Au(:,3),q(ii(1):ii(1)+3),q(ii(8):ii(8)+3));
JUj4 =
JUj(B(8).sp,B(1).Au4(:,3),B(8).Au(:,3),q(ii(1):ii(1)+3),q(ii(8):ii(8)+3));
JUi5 =
JUi(B(1).sp5,B(1).Au5(:,3),B(10).Au(:,3),q(ii(1):ii(1)+3),q(ii(10):ii(10)+3
));JUj5 =
JUj(B(10).sp,B(1).Au5(:,3),B(10).Au(:,3),q(ii(1):ii(1)+3),q(ii(10):ii(10)+3
));
JUi6 =
JUi(B(1).sp6,B(1).Au6(:,3),B(12).Au(:,3),q(ii(1):ii(1)+3),q(ii(12):ii(12)+3
));JUj6 =
```

```
JUj(B(12).sp,B(1).Au6(:,3),B(12).Au(:,3),q(ii(1):ii(1)+3),q(ii(12):ii(12)+3
));
%% Driver Constraints
% i Bodies
% y=JDi(ri,si,pi,hi,rj,sj,pj)
%                ri          si         pi            hi             rj
sj        pj
JDi1 =
JDi(q(i(2):i(2)+2),B(2).sq,q(ii(2):ii(2)+3),B(2).Ad(:,3),q(i(3):i(3)+2),B(3
).sq,q(ii(3):ii(3)+3));
JDi2 =
JDi(q(i(4):i(4)+2),B(4).sq,q(ii(4):ii(4)+3),B(4).Ad(:,3),q(i(5):i(5)+2),B(5
).sq,q(ii(5):ii(5)+3));
JDi3 =
JDi(q(i(6):i(6)+2),B(6).sq,q(ii(6):ii(6)+3),B(6).Ad(:,3),q(i(7):i(7)+2),B(7
).sq,q(ii(7):ii(7)+3));
JDi4 =
JDi(q(i(8):i(8)+2),B(8).sq,q(ii(8):ii(8)+3),B(8).Ad(:,3),q(i(9):i(9)+2),B(9
).sq,q(ii(9):ii(9)+3));
JDi5 =
JDi(q(i(10):i(10)+2),B(10).sq,q(ii(10):ii(10)+3),B(10).Ad(:,3),q(i(11):i(11
)+2),B(11).sq,q(ii(11):ii(11)+3));
JDi6 =
JDi(q(i(12):i(12)+2),B(12).sq,q(ii(12):ii(12)+3),B(12).Ad(:,3),q(i(11):i(11
)+2),B(13).sq,q(ii(13):ii(13)+3));
% j Bodies
JDj1 = JDj(q(ii(2):ii(2)+3),B(2).Ad(:,3),B(3).sq,q(ii(3):ii(3)+3));
JDj2 = JDj(q(ii(4):ii(4)+3),B(4).Ad(:,3),B(5).sq,q(ii(5):ii(5)+3));
JDj3 = JDj(q(ii(6):ii(6)+3),B(6).Ad(:,3),B(7).sq,q(ii(7):ii(7)+3));
JDj4 = JDj(q(ii(8):ii(8)+3),B(8).Ad(:,3),B(9).sq,q(ii(9):ii(9)+3));
JDj5 = JDj(q(ii(10):ii(10)+3),B(10).Ad(:,3),B(11).sq,q(ii(11):ii(11)+3));
JDj6 = JDj(q(ii(12):ii(12)+3),B(12).Ad(:,3),B(13).sq,q(ii(13):ii(13)+3));
%% Euler Parameter Constraints
JPi01 = JPi(q(ii(01):ii(01)+3));JPi02 = JPi(q(ii(02):ii(02)+3));JPi03 =
JPi(q(ii(03):ii(03)+3));
JPi04 = JPi(q(ii(04):ii(04)+3));JPi05 = JPi(q(ii(05):ii(05)+3));JPi06 =
JPi(q(ii(06):ii(06)+3));
JPi07 = JPi(q(ii(07):ii(07)+3));JPi08 = JPi(q(ii(08):ii(08)+3));JPi09 =
JPi(q(ii(09):ii(09)+3));
JPi10 = JPi(q(ii(10):ii(10)+3));JPi11 = JPi(q(ii(11):ii(11)+3));JPi12 =
JPi(q(ii(12):ii(12)+3));
JPi13 = JPi(q(ii(13):ii(13)+3));JPi14 = JPi(q(ii(14):ii(14)+3));
%% Jacobian Assembler
j = zeros(98,98);
%Fixed
j(1:6,1:7) = JFi1;
%Spherical
j(7:9,15:21) = JSi1;     j(7:9,92:end) = JSj1;%Body 3 & Body 14
j(10:12,29:35) = JSi2; j(10:12,92:end) = JSj2;%Body 5 & Body 14
j(13:15,43:49) = JSi3; j(13:15,92:end) = JSj3;%Body 7 & Body 14
j(16:18,57:63) = JSi4; j(16:18,92:end) = JSj4;%Body 9 & Body 14
j(19:21,71:77) = JSi5; j(19:21,92:end) = JSj5;%Body 11 & Body 14
j(22:24,85:91) = JSi6; j(22:24,92:end) = JSj6;%Body 13 & Body 14
%Slider
j(25:29,8:14) = JTi1;  j(25:29,15:21) = JTj1; %Body 2 & Body 3
j(30:34,22:28) = JTi2;  j(30:34,29:35) = JTj2;%Body 4 & Body 5
j(35:39,36:42) = JTi3;  j(35:39,43:49) = JTj3;%Body 6 & Body 7
j(40:44,50:56) = JTi4;  j(40:44,57:63) = JTj4;%Body 8 & Body 9
j(45:49,64:70) = JTi5;  j(45:49,71:77) = JTj5;%Body 10 & Body 11
j(50:54,78:84) = JTi6;  j(50:54,85:91) = JTj6;%Body 12 & Body 13
%Universal
```

```
j(55:58,1:7) = JUi1;  j(55:58,8:14) = JUj1;%Body 1 & Body 2
j(59:62,1:7) = JUi2;  j(59:62,22:28) = JUj2;%Body 1 & Body 4
j(63:66,1:7) = JUi3;  j(63:66,36:42) = JUj3;%Body 1 & Body 6
j(67:70,1:7) = JUi4;  j(67:70,50:56) = JUj4;%Body 1 & Body 8
j(71:74,1:7) = JUi5;  j(71:74,64:70) = JUj5;%Body 1 & Body 10
j(75:78,1:7) = JUi6;  j(75:78,78:84) = JUj6;%Body 1 & Body 12
j(79,8:14) = JDi1;  j(79,15:21) = JDj1;%Body 2 & Body 3
j(80,22:28) = JDi2;  j(80,29:35) = JDj2;%Body 4 & Body 5
j(81,36:42) = JDi3;  j(81,43:49) = JDj3;%Body 6 & Body 7
j(82,50:56) = JDi4;  j(82,57:63) = JDj4;%Body 8 & Body 9
j(83,64:70) = JDi5;  j(83,71:77) = JDj5;%Body 10 & Body 11
j(84,78:84) = JDi6;  j(84,85:91) = JDj6;%Body 12 & Body 13
j(85,1:7) = JPi01;  j(86,8:14) = JPi02;%Body 1 & Body 2
j(87,15:21) = JPi03;  j(88,22:28) = JPi04;%Body 3 & Body 4
j(89,29:35) = JPi05;  j(90,36:42) = JPi06;%Body 5 & Body 6
j(91,43:49) = JPi07;  j(92,50:56) = JPi08;%Body 7 & Body 8
j(93,57:63) = JPi09;  j(94,64:70) = JPi10;%Body 9 & Body 10
j(95,71:77) = JPi11;  j(96,78:84) = JPi12;%Body 11 & Body 12
j(97,85:91) = JPi13;  j(98,92:98) = JPi14;%Body 13 & Body 14
end
```

Funções dos Jacobianos das restrições

```
%Jacobiano das juntas esfericas corpos i
function y = JSi(s,P)
sTilde = SkewSymmetric(s);
y = [-eye(3), 2*(getA(P)*sTilde)*getG(P)];
end

%Jacobiano das juntas esfericas corpos j
function y = JSj(s,P)
sTilde = SkewSymmetric(s);
y= [eye(3) 2*-getA(P)*sTilde*getG(P)];
end

%Jacobiano das juntas prismaticas corpos i
function y = JTi(ri,si,pi,fi,gi,rj,sj,pj,fj,hj)
%
d = dij(ri,si,pi,rj,sj,pj);
fiTilde = SkewSymmetric(fi);
giTilde = SkewSymmetric(gi);
siTilde = SkewSymmetric(si);
y = [zeros(1,3),    -2*hj'*getA(pj).'*getA(pi)*fiTilde*getG(pi);...
     zeros(1,3),    -2*hj'*getA(pj).'*getA(pi)*giTilde*getG(pi);...
     -fi'*getA(pi).', 2*(fi'*siTilde-d'*getA(pi)*fiTilde)*getG(pi);...
     -gi'*getA(pi).', 2*(gi'*siTilde-d'*getA(pi)*giTilde)*getG(pi);...
     zeros(1,3),    -2*fj'*getA(pj).'*getA(pi)*fiTilde*getG(pi)];
end

%Jacobiano das juntas prismaticas corpos j
function y = JTj(pi,fi,gi,fj,hj,sj,pj)
%
fjTilde = SkewSymmetric(fj);
hjTilde = SkewSymmetric(hj);
sjTilde = SkewSymmetric(sj);
y = [zeros(1,3),     -2*fi'*getA(pi).'*getA(pj)*hjTilde*getG(pj);...
     zeros(1,3),     -2*gi'*getA(pi).'*getA(pj)*hjTilde*getG(pj);...
```

```
     fi'*getA(pi).', -2*fi'*getA(pi).'*getA(pj)*sjTilde*getG(pj);...
     gi'*getA(pi).', -2*gi'*getA(pi).'*getA(pj)*sjTilde*getG(pj);...
     zeros(1,3),     -2*fi'*getA(pi).'*getA(pj)*fjTilde*getG(pj)];
end

%Jacobiano das juntas Universais corpos i
function y = JUi(si,hi,hj,Pi,Pj)
sTilde = SkewSymmetric(si);
hiTilde = SkewSymmetric(hi);
y = [  -eye(3) 2*getA(Pi)*sTilde*getG(Pi)
     zeros(1,3) -2*hj'*getA(Pj).'*getA(Pi)*hiTilde*getG(Pi)];
end

%Jacobiano das juntas Universais corpos j
function y = JUj(sj,hi,hj,Pi,Pj)
sTilde = SkewSymmetric(sj);
hiTilde = SkewSymmetric(hj);
y = [    eye(3) -2*getA(Pj)*sTilde*getG(Pj)
     zeros(1,3) -2*hi'*getA(Pi).'*getA(Pj)*hiTilde*getG(Pj)];
end

%Jacobiano das juntas Motoras corpos i
function y = JDi(ri,si,pi,hi,rj,sj,pj)
siTilde = SkewSymmetric(si);
hiTilde = SkewSymmetric(hi);
d = dij(ri,si,pi,rj,sj,pj);
y = [hi'*getA(pi).' 2*(hi'*siTilde-d'*getA(pi)*hiTilde)*getG(pi)] -
(0.001);
end

%Jacobiano das juntas Motoras corpos j
function y = JDj(pi,hi,sj,pj)
sjTilde = SkewSymmetric(sj);
y = [hi'*getA(pi).', -2*(hi'*getA(pi).'*getA(pj)*sjTilde)*getG(pi)] -
(0.001);
end

%Jacobiano dos parametros de Euler
function y = JPi(p)
y = [zeros(1,3), 2*p'];
end
```

Análise de velocidade

```
function y = VelocityAnalysis(q,t)
%
[n,m]=size(q);
y = zeros(n,m);
for i = 1:length(t)
    nu = getNu(q(:,i),t(i));
    j = getJacobian(q(:,i),t(i));
    j(85:98,:) = [];
    y(:,i) = j\nu;
end


end
```

Função Nu

```
%Função lado direito da equação de velocidade
function [Nu] = getNu(q,t)

Nu =...
  [zeros(78,1);...
    10*cos(t);...
    10*cos(t);...
    10*cos(t);...
    10*cos(t);...
    10*cos(t);...
    10*cos(t)];
end
```

Funções De Auxilio

```
%Vetor distância entre os pontos i e j
function y = dij(ri,si,pi,rj,sj,pj)
y = rj+getA(pj)*sj-ri-getA(pi)*si;
end

%Matrizes de rotação do sistema
function A = getA(p)
e0=p(1);
e = [p(2);p(3);p(4)];
eTilde = SkewSymmetric(e);

A = (2*e0^2-1)*eye(3)+2*(e*e'+e0*eTilde);


end

%Propriedade dos parâmetros de Euler
function [G] = getG(P)
G = [-P(2)  P(1)  P(4) -P(3)
     -P(3) -P(4)  P(1)  P(2)
     -P(4)  P(3) -P(2)  P(1)];
end

%Modelo de Matriz Anti-Símetrica
function Atilde = SkewSymmetric(q)
Atilde = [0 -q(3) q(2);...
          q(3) 0 -q(1);...
         -q(2) q(1) 0];
end

%Orientações Iniciais dos Corpos
%Body 2
O2 = [ 177.528522466;167.445498832;95.000000000];
 Px2 = [178.205428508;168.050519243;95.419223701];
 Py2 = [176.793412301;168.030231859;95.343074820];
 Pz2 = [177.490955790;166.905093809;95.840565974];
 A2 = [Px2-O2,Py2-O2,Pz2-O2];

%Body 4
O4 = [ 56.247794505;237.466959767;95.000000000];
 Px4 = [56.433304529;238.355687802;95.419223701];
 Py4 = [55.373845766;237.811217331;94.656925180];
```

```
 Pz4 = [55.798573365;237.164223561;95.840565974];
 A4 = [Px4-O4,Py4-O4,Pz4-O4];
 %Body 6
O6 = [ -233.776316970;70.021460936;95.000000000];

 Px6 = [-233.915155544;69.092470345;95.343074820];
 Py6 = [-232.913900903;69.737753312;94.580776299];
 Pz6 = [-233.289529155;70.259129752;95.840565974];
 A6 = [Px6-O6,Py6-O6,Pz6-O6];
 %Body 8
O8 = [ -233.776316970;-70.021460936;95.000000000];

 Px8 = [-233.637478396; -70.950451527;94.656925180];
 Py8 = [-232.913900903;-69.737753312;94.580776299];
 Pz8 = [-233.289529155;-70.259129752; 95.840565974];
 A8 = [Px8-O8,Py8-O8,Pz8-O8];
 %Body 10
O10 = [ 56.247794505;-237.466959767;95.000000000];

 Px10 = [57.12174324;-237.122702204 ;95.343074820];
 Py10 = [56.062284480;-236.578231733;94.580776299];
 Pz10 = [ 55.798573365;-237.164223561; 95.840565974];
 A10 = [Px10-O10,Py10-O10,Pz10-O10];

 %Body 12
 O12 = [ 177.528522466; -167.445498832; 95.000000000 ];
 Px12 = [ 178.263632630;-166.860765804 ;94.656925180];
 Py12 = [ 176.851616423;-166.840478421 ;94.580776299];
 Pz12 = [ 177.490955790;-166.905093809 ;95.840565974];
 A12 = [Px12-O12,Py12-O12,Pz12-O12];

%Obtenção do Chute Inicial
function qo = getInitialGuess()
B = BPar;
qo = [B(1).r;...
      B(1).p;...
      B(2).r;...
      B(2).p;...
      B(3).r;...
      B(3).p;...
      B(4).r;...
      B(4).p;...
      B(5).r;...
      B(5).p;...
      B(6).r;...
      B(6).p;...
      B(7).r;...
      B(7).p;...
      B(8).r;...
      B(8).p;...
      B(9).r;...
      B(9).p;...
      B(10).r;...
      B(10).p;...
      B(11).r;...
      B(11).p;...
      B(12).r;...
      B(12).p;...
      B(13).r;...
      B(13).p;...
      B(14).r;...
```

```
      B(14).p;...
      ];
end
```

Análise de Aceleração

```
function y = AcelerationAnalysis(q,out_t,t)
%
[n,m]=size(q);
y = zeros(n,m);
for i = 1:length(t)
    gama = GetGama(q(:,i),t(i),out_t(:,i));
    j = getJacobian(q(:,i),t(i));
    j(85:98,:) = [];
    y(:,i) = j\gama;
end

end
```

Função Gama

```
function gama = GetGama(q,t,out_t)
%Modelagem Da função Gama
%Carregando os parametros dos Corpos
B = BPar();
%% Gama Junta Fixa
% y = GamaBF(    pi    , si    ,ep    , ep0)
GamaBF1 = GamaBF(q(4:7),[0;0;0],q(5:7),q(4));
%% Gama Junta esferica
%[y] = GamaS(    pi      ,si      ,pj      ,sj       )
GamaSph = [GamaS(q(18:21),B(03).sr,q(95:98),B(14).sr1);...
           GamaS(q(32:35),B(05).sr,q(95:98),B(14).sr2);...
           GamaS(q(46:49),B(07).sr,q(95:98),B(14).sr3);...
           GamaS(q(60:63),B(09).sr,q(95:98),B(14).sr4);...
           GamaS(q(74:77),B(11).sr,q(95:98),B(14).sr5);...
           GamaS(q(88:91),B(13).sr,q(95:98),B(14).sr6)];
%% Gama Junta Prismatica
% y = GamaT     (fi           ,fj           ,gi           ,hj           ,pi
,pj                ,rpi          ,rpj            ,si      ,sj      ,ri
,rj)
i = 1:7:98;
ii = 4:7:98;
k = 1:length(t);
GamaTra =
[GamaT(B(02).Ad(:,1),B(03).Ad(:,1),B(02).Ad(:,2),B(03).Ad(:,3),q(ii(02):ii(
02)+3),q(ii(03):ii(03)+3),out_t(8:10,k),
out_t(15:17,k),B(02).sq,B(03).sq,q(i(02):i(02)+2),q(i(03):i(03)+2));...

GamaT(B(04).Ad(:,1),B(05).Ad(:,1),B(04).Ad(:,2),B(05).Ad(:,3),q(ii(04):ii(0
4)+3),q(ii(05):ii(05)+3),out_t(22:24,k),out_t(29:31,k),B(04).sq,B(05).sq,q(
i(04):i(04)+2),q(i(05):i(05)+2));...

GamaT(B(06).Ad(:,1),B(07).Ad(:,1),B(06).Ad(:,2),B(07).Ad(:,3),q(ii(06):ii(0
6)+3),q(ii(07):ii(07)+3),out_t(36:38,k),out_t(43:45,k),B(06).sq,B(07).sq,q(
i(06):i(06)+2),q(i(07):i(07)+2));...

GamaT(B(08).Ad(:,1),B(09).Ad(:,1),B(08).Ad(:,2),B(09).Ad(:,3),q(ii(08):ii(0
8)+3),q(ii(09):ii(09)+3),out_t(50:52,k),out_t(57:59,k),B(08).sq,B(09).sq,q(
i(08):i(08)+2),q(i(09):i(09)+2));...
```

```
GamaT(B(10).Ad(:,1),B(11).Ad(:,1),B(10).Ad(:,2),B(11).Ad(:,3),q(ii(10):ii(1
0)+3),q(ii(11):ii(11)+3),out_t(64:66,k),out_t(71:73,k),B(10).sq,B(11).sq,q(
i(10):i(10)+2),q(i(11):i(11)+2));...

GamaT(B(12).Ad(:,1),B(13).Ad(:,1),B(12).Ad(:,2),B(13).Ad(:,3),q(ii(12):ii(1
2)+3),q(ii(13):ii(13)+3),out_t(78:80,k),out_t(85:87,k),B(12).sq,B(13).sq,q(
i(12):i(12)+2),q(i(13):i(13)+2))];
%% Gama Junta Universal
%y = GamaU      (pi    ,si      ,pj                ,sj      ,hi
,hj)
jj =4:7:98;
GamaUni =
[GamaU(q(4:7),B(1).sp1,q(jj(02):jj(02)+3),B(02).sp,B(1).Au1(:,3),B(02).Au(:
,3));...

GamaU(q(4:7),B(1).sp2,q(jj(04):jj(04)+3),B(04).sp,B(1).Au2(:,3),B(04).Au(:,
3));...

GamaU(q(4:7),B(1).sp3,q(jj(06):jj(06)+3),B(06).sp,B(1).Au3(:,3),B(06).Au(:,
3));...

GamaU(q(4:7),B(1).sp4,q(jj(08):jj(08)+3),B(08).sp,B(1).Au4(:,3),B(08).Au(:,
3));...

GamaU(q(4:7),B(1).sp5,q(jj(10):jj(10)+3),B(10).sp,B(1).Au5(:,3),B(10).Au(:,
3));...

GamaU(q(4:7),B(1).sp6,q(jj(12):jj(12)+3),B(12).sp,B(1).Au6(:,3),B(12).Au(:,
3))];
%% Gama Junta Motora
%y = GamaM      (ri              ,rj              ,pi                ,pj
,si      ,sj      ,hi            ,rpi             ,rpj          ,t)
GamaMot =
[GamaM(q(i(02):i(02)+2),q(i(03):i(03)+2),q(ii(02):ii(02)+3),q(ii(03):ii(03)
+3),B(02).sq,B(03).sq,B(02).Ad(:,3),out_t(8:10,k), out_t(15:17,k),t);...

GamaM(q(i(04):i(04)+2),q(i(05):i(05)+2),q(ii(04):ii(04)+3),q(ii(05):ii(05)+
3),B(04).sq,B(05).sq,B(04).Ad(:,3),out_t(22:24,k),out_t(29:31,k),t);...

GamaM(q(i(06):i(06)+2),q(i(07):i(07)+2),q(ii(06):ii(06)+3),q(ii(07):ii(07)+
3),B(06).sq,B(07).sq,B(06).Ad(:,3),out_t(36:38,k),out_t(43:45,k),t);...

GamaM(q(i(08):i(08)+2),q(i(09):i(09)+2),q(ii(08):ii(08)+3),q(ii(09):ii(09)+
3),B(08).sq,B(09).sq,B(08).Ad(:,3),out_t(50:52,k),out_t(57:59,k),t);...

GamaM(q(i(10):i(10)+2),q(i(11):i(11)+2),q(ii(10):ii(10)+3),q(ii(11):ii(11)+
3),B(10).sq,B(11).sq,B(10).Ad(:,3),out_t(64:66,k),out_t(71:73,k),t);...

GamaM(q(i(12):i(12)+2),q(i(13):i(13)+2),q(ii(12):ii(12)+3),q(ii(13):ii(13)+
3),B(12).sq,B(13).sq,B(12).Ad(:,3),out_t(78:80,k),out_t(85:87,k),t)];
%% Montagem Função Gama
gama = [GamaBF1
        GamaSph
        GamaTra
        GamaUni
        GamaMot];
end
```

Funções Gama das Principais Restrições

```
function y = GamaBF(pi,si,ep,ep0)
%Função Gama Base Fixa
```

```
OMi = getOMi(pi);
Oi = [OMi(3,2); OMi(1,3); OMi(2,1)];
eTilde = SkewSymmetric(ep);

y = [-getA(pi)*OMi*OMi*si;
     -1/2*(eTilde + ep0*eye(3))*Oi];
end


function [y] = GamaS(pi,si,pj,sj)
%Função Gama Junta esférica
OMi = getOMi(pi);
OMj = getOMi(pj);
y = getA(pi)*OMi*OMi*si - getA(pj)*OMj*OMj*sj;

end


function y = GamaT(fi,fj,gi,hj,pi,pj,rpi,rpj,si,sj,ri,rj)
%Função Gama Junta de Translação
d = dij(ri,si,pi,rj,sj,pj);
OMi = getOMi(pi);
OMj = getOMi(pj);
Oi = [OMi(3,2); OMi(1,3); OMi(2,1)];
Oj = [OMj(3,2); OMj(1,3); OMj(2,1)];
fiTilde = SkewSymmetric(fi);
giTilde = SkewSymmetric(gi);
fjTilde = SkewSymmetric(fj);
hjTilde = SkewSymmetric(hj);


y = [-hj'*(getA(pj).'*getA(pi)*OMi*OMi + OMj*OMj*getA(pj).'*getA(pi))*fi +
2*Oj.'*hjTilde*getA(pj).'*getA(pi)*fiTilde*Oi
     -hj'*(getA(pj).'*getA(pi)*OMi*OMi + OMj*OMj*getA(pj).'*getA(pi))*gi +
2*Oj.'*hjTilde*getA(pj).'*getA(pi)*fiTilde*Oi
     2*Oi'*fiTilde*getA(pi).'*(rpi - rpj) +
2*sj'*OMj*getA(pj).'*getA(pi)*OMi*fi - si'*OMi*OMi*fi -
sj'*OMj*OMj*getA(pj).'*getA(pi)*fi - d'*getA(pi)*OMi*OMi*fi
     2*Oi'*giTilde*getA(pi).'*(rpi - rpj) +
2*sj'*OMj*getA(pj).'*getA(pi)*OMi*gi - si'*OMi*OMi*gi -
sj'*OMj*OMj*getA(pj).'*getA(pi)*gi - d'*getA(pi)*OMi*OMi*gi
     -fj'*(getA(pj).'*getA(pi)*OMi*OMi + OMj*OMj*getA(pj).'*getA(pi))*fi +
2*Oj.'*fjTilde*getA(pj).'*getA(pi)*fiTilde*Oi];
end


function y = GamaU(pi,si,pj,sj,hi,hj)
%Função Gama Junta Universal
hiTilde = SkewSymmetric(hi);
hjTilde = SkewSymmetric(hj);
OMi = getOMi(pi);
OMj = getOMi(pj);
Oi = [OMi(3,2); OMi(1,3); OMi(2,1)];
Oj = [OMj(3,2); OMj(1,3); OMj(2,1)];

y = [getA(pi)*OMi*OMi*si - getA(pj)*OMj*OMj*sj
    -hj'*(getA(pj).'*getA(pi)*OMi*OMi + OMj*OMj*getA(pj).'*getA(pi))*hi +
2*Oj.'*hjTilde*getA(pj).'*getA(pi)*hiTilde*Oi];
end
```

```
function y = GamaM(ri,rj,pi,pj,si,sj,hi,rpi,rpj,t)
%Função Gama Junta Motora
d = dij(ri,si,pi,rj,sj,pj);
OMi = getOMi(pi);
OMj = getOMi(pj);
Oi = [OMi(3,2); OMi(1,3); OMi(2,1)];
hiTilde = SkewSymmetric(hi);
y = (2*Oi'*hiTilde*getA(pi).'*(rpi - rpj) +
2*sj'*OMj*getA(pj).'*getA(pi)*OMi*hi - si'*OMi*OMi*hi -
sj'*OMj*OMj*getA(pj).'*getA(pi)*hi - d'*getA(pi)*OMi*OMi*hi) + 10*cos(t);
end


%Matriz de anti-simetrica Omega(OT)
function [y] = getOMi(pi)
e0 =pi(1);
e = [pi(2);pi(3);pi(4)];
eTilde = (e0*eye(3)) + SkewSymmetric(e);
diffA = eTilde*getA(pi);

y = (getA(pi).')*diffA;

end


%Matriz de anti-simetrica Omega(OT)
function [y] = getOMj(pj)
e0 =pj(1);
e = [pj(2);pj(3);pj(4)];
eTilde = (e0*eye(3)) + SkewSymmetric(e);
diffA = eTilde*getA(pj);

y = (getA(pj).')*diffA;
end
```