UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Implementação de Métodos Numéricos para a Resolução do Problema Cinemático Inverso de Robôs com Ênfase em Controle de Posição

Autor: **Cláudio Eduardo Aravéchia de Sá**
Orientador: **Prof. Dr. João Maurício Rosário**

Agosto / 1996

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Implementação de Métodos Numéricos para a Resolução do Problema Cinemático Inverso de Robôs com Ênfase em Controle de Posição

Autor: **Claudio Eduardo Aravéchia de Sá**
Orientador: **Prof. Dr. João Maurício Rosário**

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos

Dissertação de mestrado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requesito para a obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 1996
S.P. - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Sa11i

Sá, Cláudio Eduardo Aravéchia de
Implementação de métodos numéricos para a resolução do problema cinemático inverso de robôs com ênfase em controle de posição / Cláudio Eduardo Aravéchia de Sá.-- Campinas, SP: [s.n.], 1996.

Orientador: João Maurício Rosário.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Cinemática. 2. Robôs industriais. 3. Manipuladores mecânicos. I. Rosário, João Maurício. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

DISSERTAÇÃO DE MESTRADO

Implementação de Métodos Numéricos para a Resolução do Problema Cinemático Inverso de Robôs com Ênfase em Controle de Posição

Autor: **Claudio Eduardo Aravéchia de Sá**

Orientador: **Prof. Dr. João Maurício Rosário**

João Maurício Rosário

Prof. Dr. João Maurício Rosário, Presidente
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. José Manoel Balthazar
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. João Carlos Mendes Carvalho
Faculdade de Engenharia Mecânica - UFU

Campinas, 21 de Agosto de 1996

Dedicatória

Aos meus pais Eduardo e Ana,

e meus irmãos Rodrigo e Carla.

Agradecimentos

Este trabalho não poderia ter sido concluido sem o apoio e a ajuda de diversas pessoas às quais presto homenagem:

- Ao Professor João Maurício Rosário pela orientação, incentivo e confiança em todas as etapas deste projeto.

- Ao Professor José Manoel Balthazar, que fez despertar em min o gosto pela robótica.

- Aos meus amigos por proporcionarem inúmeros e indispensáveis momentos de descontração.

- A todos que, direta ou indiretamente, contribuíram para a conclusão deste projeto.

Sumário

Resumo

Sá, Claudio Eduardo Aravéchia de, *Implementação de Métodos Números para a Resolução do Problema Cinemático Inverso de Robôs com Ênfase em Controle de Posição*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1996. 114 p. Tese (Mestrado)

Um robô industrial pode ser modelado como uma cadeia articulada com diversos corpos rígidos conectados por juntas de revolução ou prismáticas que são movimentadas por atuadores. A partir de um dado conjunto de ângulos ou deslocamentos e da geometria do mesmo, torna-se possível o conhecimento da posição e orientação da garra em relação a um referencial solidário à base do manipulador (modelo cinemático direto). Entretanto, em grande parte das aplicações, a partir do conhecimento de uma dada posição e orientação da garra, é necessário calcular os ângulos ou deslocamentos necessários para movimentação das juntas do manipulador para atingir o objetivo desejado (modelo cinemático inverso). Neste trabalho é dado enfâse a robôs com juntas rotacionais, mas isto não impede que os resultados obtidos possam ser extendidos para robôs com outros tipos de juntas. Este trabalho tem como objetivo o estudo de algorítmos para a inversão da matriz Jacobiana necessária para a solução do problema cinemático inverso, pelo método recursivo, que possam ser implementados em tempo real e, ao mesmo tempo, garantir convergência e estabilidade do sistema. Os algorítmos foram implementados em linguaguem ADA em um microcomputador compatível com a linha IBM-PC-AT.

Palavras Chaves

- Cinemática Inversa, Robôs, Controle de Posição

Abstract

Sá, Claudio Eduardo Aravéchia de, *Implementação de Métodos Números para a Resolução do Problema Cinemático Inverso de Robôs com Ênfase em Controle de Posição*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1996. 114 p. Tese (Mestrado)

An industrial robot may be modeled by a series of links interconnected by either rotary or sliding joints driven by atuators. From a given angular configuration and the geometric's manipulator, it is possible to know the position and orientation of its end effector with respect to a coordinate system fixed in the base of the manipulator (kinematic direct model). However, in many aplications, from a knowlegd of a give position and orientation of end effector is necessary to calculate the angles or displacements necessary to the movement of the joint of the manipulator to achieve a given objective (kinematic inverse model). This work focuses robots with rotational joints, but the results may be extended for others types of robots. This work focus the study of algorithms for the inversion of Jacobian matrix necessary for the solution of inverse kinematic, through recursive method, that may be implemented in real time to achieve system convergence and stability. The algorithms will be implementing in ADA language in the computer compatible with IBM-PC-AT line.

Key Words

- Kinematic Inverse, Robots, Position Control

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

O problema do aumento da produtividade vêm atraindo a atenção mundial a muito tempo e uma das possíveis soluções, para isto, é a modernização dos meios de produção por meio da automação. Dentro deste escopo, os robôs industriais possuem grande importância, tendo em vista sua capacidade de realizar tarefas que exigem flexibilidade, rapidez e precisão.

Na realização de uma tarefa, o robô realiza um ciclo de operações sequênciais, que pode ser alterado devido a fatores externos. A maioria dos robôs utilizados no meio industrial possuem juntas rotativas, motivo que este trabalho será dirigido a esse tipo de robô, mas poderá ser facilmente extendido a outros tipos de robôs.

Para que a utilização de robôs na industria seja viável, é necessário que a sua programação seja simples e de fácil modificação, permitindo ao usuário rapidez e flexibilidade.

Os métodos de programação mais frequentemente utilizados em robôs industriais são:

- Aprendizagem ponto-a-ponto:

i) Movimento angular: Neste método são gravados pontos de referência fornecidos pelos transdutores de posição localizados em cada junta e a partir desses pontos são geradas trajetórias (angulares) através da utilização de algorítmos de interpolação. A simplicidade deste método não exige o conhecimento do modelo geométrico do robô;

ii) Movimento cartesiano: Idêntico ao procedimento descrito anteriormente mas a obtenção dos pontos de referência é realizada utilizando o modelo geométrico do manipulador;

- Programação off-line: A partir de um software para visualização gráfica do modelo geométrico de robôs, devem ser obtidos pontos de passagem correspondentes à trajetória do robô (expressos em coordenadas angulares). Esses pontos de passagem poderão ser obtidos a partir do movimento angular de cada junta, ou a partir do modelo geométrico. A partir de um conjunto de pontos correspondentes à trajetória a ser realizada pelo robô, torna-se possível a implementação de algoritmos off-line para interpolação e filtragem, levando-se em consideração aspectos dinâmicos e testes de colisão.

- Programação on-line: A partir do conhecimento do modelo geométrico e das características da trajetória desejada (posição final, velocidade e forma do caminho), pode-se implementar algorítmos para modelagem cinemática inversa e controle de posição.

Os métodos descritos anteriormente estão associados com o tipo de aplicação requerida. Isto permitirá:

- a realização de tarefas mais precisas e complexas;
- a necessidade da utilização de posições determinadas analíticamente, ou informações provenientes de sensores de percepção externa;
- ao uso de manipuladores em ambientes adversos à presença do homem;
- a necessidade do aumento do tempo útil de trabalho pois, durante a programação da tarefa pelo método de aprendizagem, o manipulador é utilizado, diminuindo o seu tempo útil.

Normalmente a programação de tarefas de robôs é realizada no espaço das juntas, não necessitando de um modelo geométrico, e a trajetória angular de mesma natureza dos sinais provenientes do trandutor de posição servirá como sinal de referência para o controlador de cada junta robótica (figura 1). Entretanto a realização de algumas tarefas em relação a um sistema de referência colocado na ferramenta (espaço cartesiano) exige o conhecimento

completo do modelo geométrico, e torna necessário a utilização de uma transformação de coordenadas, tendo em vista que o sinal de referência correspondente à trajetória, necessária para o controle das juntas, deverá ser angular.

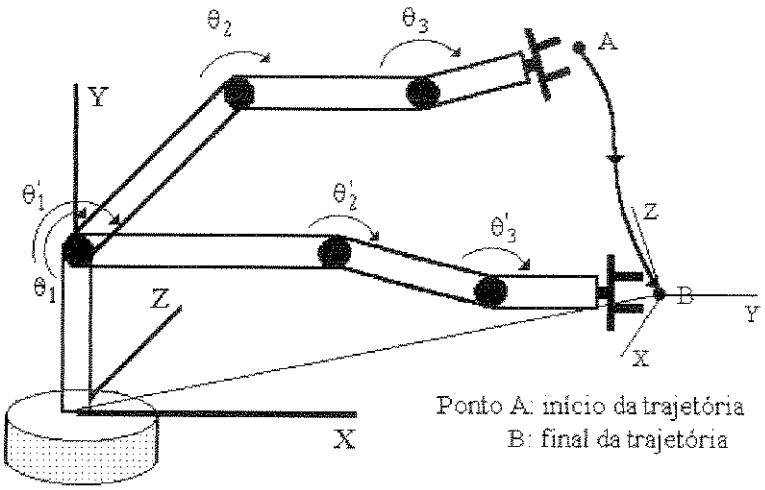


Figura 1: Espaço de Coordenadas de um Robô.

Em um robô, os diferentes graus de liberdade podem ser associados a diversos sistemas de coordenadas, cada um associado a um grau de liberdade e servindo para descrever o movimento de cada grau de liberdade associado. Mais especificamente, o modelo geométrico é aquele que expressa a posição e orientação da garra em relação a um sistema de coordenadas solidário à base do robô, em função de suas coordenadas generalizadas (angulares no caso de juntas rotacionais). Esta relação pode ser expressa matematicamente a partir de uma matriz que relaciona o sistema de coordenadas da base com o sistema de coordenadas do último elemento. Esta matriz é chamada de matriz de passagem homogênea do robô.

Embora a definição do modelo geométrico seja única, a maneira de obter a matriz de passagem homogênea do robô está associada ao sistema de referência utilizado. Existem diversas maneiras, e cada uma gera expressões diferentes que, embora equivalentes quantitativamente, podem diferir quanto ao número de operações aritméticas necessárias para fazer o cálculo numérico das mesmas. Tendo em vista a aplicação do modelo em sistemas de

controle em tempo real, figura 2, é importante que se obtenha um modelo com o menor número possível de operações matemáticas.

Entretanto, a estratégia de controle nos robôs é exercida sobre seus atuadores, sendo que, o sistema de controle só segue referências ou movimentos desejados definidos neste espaço (espaço das juntas). Contudo, o operador define as tarefas ou movimentos de referência no espaço operacional. Vemos assim que os movimentos desejados e as leis de controle estão em espaços diferentes. A obtenção de referências correspondentes às tarefas definidas no espaço operacional é denominada coordenação de movimentos, figura 2. A coordenação é expressa matematicamente pela inversão do modelo geométrico. Existem dois métodos para a solução do problema da inversão do modelo geométrico :

- Métodos analíticos. Estes métodos não são gerais, isto é, a inversão analítica não é trivial e não há garantia de que seja possível fazê-la para um robô qualquer. Além disso, caso seja encontrada a solução ela pode apresentar soluções multiplas (problema de redundâncias).

- Métodos numéricos. Estes métodos convergem para uma solução possível entre todas as existentes e com o atual desenvolvimento dos microcomputadores a utilização destes métodos em tempo real é viável. Os dois métodos mais utilizados são o método da linearização da matriz de passagem do robô e o método recursivo que utiliza o cálculo do modelo geométrico e da matriz Jacobiana inversa.

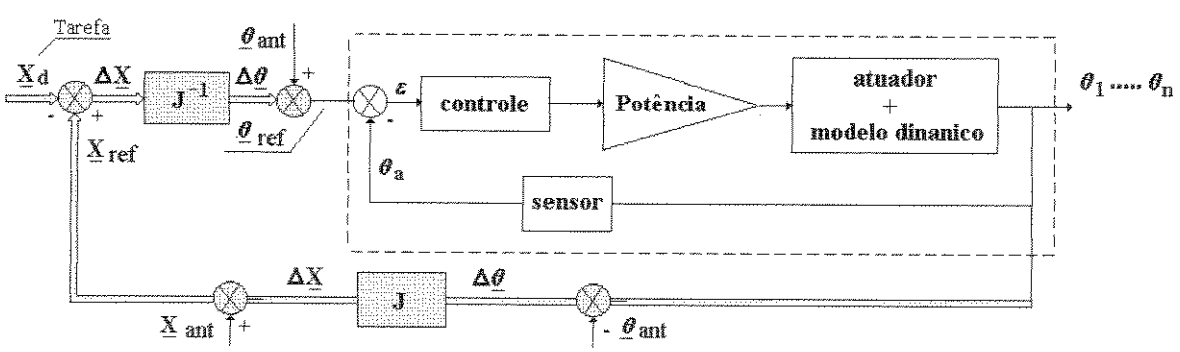


Figura 2: Coordenação de movimentos.

Diferentemente das soluções analíticas, as soluções numéricas podem ser combinadas com estratégias de anticolisão o que justifica a importância destes métodos. Entretanto, o uso

de métodos números podem ser um sucesso se ,e somente se, um algorítmo eficiente for desenvolvido.

Dentro deste escopo, este trabalho visa o desenvolvimento de um algorítmo de geração de trajetórias utilizando o modelo cinemático inverso, obtido através de método numérico, e o estudo de métodos para a inversão da matriz jacobiana (métodos de Gauss, de Greville e de Miss) que possam ser utilizados em tempo real e ao mesmo tempo possam garantir a convergência e estabilidade do sistema.

O algorítmo será implementado em linguaguem ADA, na forma de funções. Essa linguaguem apresenta alto grau de estruturação, o que permite simplificações na programação de tarefas com alto grau de complexidade.

O modelo geométrico será obtido para o manipulador submarino Kraft da Petrobrás, de seis graus de liberdade, apresentado no Anexo D, com juntas rotacionais e a partir disto serão realizadas simulações e estudados os comportamentos das trajetórias obtidas (oscilações) e um estudo qualitativo dos métodos implementados utilizados para a inversão da matriz Jacobiana (convergência, comportamento em relação a sistemas mal-condicionados entre outros).

No capítulo 2 faz-se uma breve análise dos métodos utilizados para a programação de robôs mostrando suas vantagens e desvantagens e discute-se o problema cinemático de um robô.

No capítulo 3 apresenta-se os métodos que serão implementados para a inversão da matriz Jacobiana necessários para a resolução do problema cinemático inverso e uma breve discusão sobre sistemas mal-condicionados.

No capítulo 4 os métodos serão descritos matematicamente e será apresentado o algorítmo para a geração de trajetórias off-line.

E, finalmente, no capítulo 6 apresenta-se as conclusões finais e perspectivas futuras.

CAPÍTULO 2

Posicionamento, Geração de Trajetórias e Controle

Os robôs são equipamentos multifuncionais reprogramáveis com grande flexibilidade de operação. Atualmente, a programação de tarefas é realizada através de uma "caixa de ensino" que é utilizada para conduzir o robô através das posições críticas do ciclo de operação. Este tipo de programação de tarefas posssui incovenientes, entre eles: utiliza o robô no período de programação e não permite um controle mais preciso sobre a trajetória da garra ou ferramenta.

Já a programação de tarefas off-line, não apresenta os incovenientes acima por ser realizada em computadores, necessitando apenas de um modelo matemático. Este modelo contém informações sobre a geometria do sistema (modelo cinemático). Neste capítulo serão mostradas as formas de programação de tarefas e as suas vantagens e desvantagens.

Como mencionado anteriormente, a tarefa de um robô é especificada em termos de coordenadas cartesianas, $\underline{x}$. Estas consistem na posição, descrita por um vetor posição, $\underline{p}$, e um vetor orientação, descritos por um vetor unitário de aproximação $\underline{a}$ e um vetor unitário de deslizamento $\underline{s}$. Todos esses vetores são definidos em relação ao sistema de coordenadas da base. Por conveniência, um vetor unitário normal é definido como $\underline{n} = \underline{s} \times \underline{a}$, onde $\times$ denota o produto vetorial, figura 2.1.

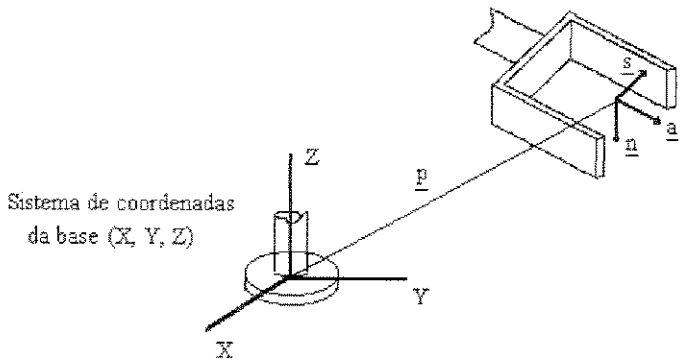


Figura 2.1: Vetores de posição e de orientação de um robô.

Contudo, o elemento terminal é acionado por atuadores ligados às juntas que necessitam de referências angulares. Deste modo torna-se necessária a determinação do conjunto de ângulos (espaço das juntas) correspondentes ao vetor $\underline{x}$ (espaço cartesiano). Isto requer a inversão do modelo cinemático do robô (modelo cinemático inverso), isto pode ser obtido analiticamente ou numericamente. A solução, caso exista, pode não ser única como poderá ser visto neste capítulo. Além disso será apresentada uma breve discussão sobre a matriz Jacobiana.

2.1 - Programação de tarefas

2.1.1 - Programação por aprendizagem

Programar visa o estabelecimento de uma sequência de operações a serem executadas pelo robô. A programação das tarefas pode ser realizada através de uma programação por aprendizagem ou a partir de uma linguagem de programação de computadores.

A programação por aprendizagem pode ser realizada pelos seguintes métodos:

- *Aprendizagem direta*. Neste método, figura 2.2a, o operador guia fisicamente o robô por seu órgão terminal. Enquanto isto, os sensores de posição de cada junta do robô são utilizados para memorizar os pontos importantes da tarefa a ser executada. Este método é

- *Aprendizagem por simulação física.* Neste método, figura 2.2b, o operador guia um simulador físico, que tem a geometria e os sensores idênticos ao robô original. Uma vez memorizada a tarefa, esta é transferida para o sistema de controle do robô. Este método é indicado para robôs de grande porte ou com estruturas mecânicas não reversíveis, bem como a programação em ambientes que exijam distância. Este método necessita de um modelo preciso e conhecido.

- *Aprendizagem por telecomando.* Neste método, figura 2.2c, um dispositivo de telecomando (teach pendant ou teach-in box) é utilizado para mover cada junta do robô isoladamente ou fornecer a posição e orientação da garra. Este método é adequado para qualquer tipo de robô e resulta mais barato que o método anterior, caso o telecomando seja feito a partir de um dispositivo de programação, que permita a atuação sobre cada ligação independentemente.

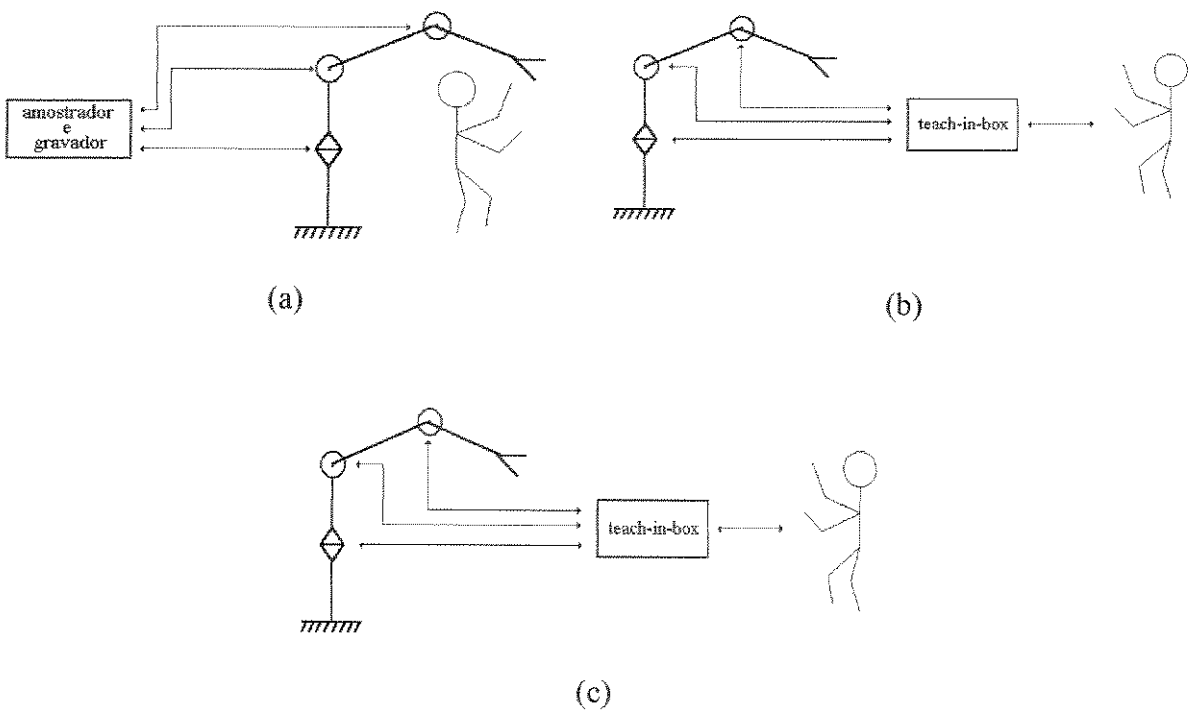


Figura 2.2: Programação por aprendizagem: o operador guia o robô diretamente (a), utilizando um simulador físico (b) e por telecomando (c).

2.1.1.1 - Memorização de tarefas

A memorização de tarefas durante a fase de aprendizagem pode ser feita de duas maneiras:

- *Programação ponto a ponto*: É gravado apenas os pontos essenciais da trajetoria do orgão terminal, de modo que o movimento entre dois pontos consecutivos gravados fica determinado pelos algorítmos de controle utilizados nos servo-mecanismos das diversas ligações. Este método é adequado para aplicações que não requeiram um controle preciso da trajetória e da velocidade intermediários aos pontos essenciais da tarefa.

- *Programação por caminhos contínuos*: Os pontos da trajetória, na aprendizagem, são amostrados com uma taxa elevada e armazenados na memória. Estes pontos podem ser amostrados no controle segundo uma taxa programada, conduzindo o robô, contínuamente por inércia ou por "arrasto", de maneira precisa e em uma velocidade que é função da taxa de amostragem.

2.1.2 - Programação por linguagens

A programação usando linguagens pode ser considerada como o processo pelo qual os programas são desenvolvidos sem a necessidade do uso do robô propriamente dito mas através da utilização de uma linguagem de programação de computador. Com o atual avanço na tecnologia de desenvolvimento de hardware e software, a programação de robôs por linguagens, está cada vez mais próxima da realidade industrial. Estes avanços incluem uma maior sofisticação dos controladores, melhor precisão de posicionamento e incremento no número e tipos de sensores.

2.1.2.1 - Níveis de programação

As linguagens de programação de robôs podem ser classificadas nos seguintes níveis:

- Nível de junta. As linguagens classificadas neste nível requerem a programação individual de cada junta do robô para que uma dada posição seja alcançada.

- Nível de manipulador. Neste nível é necessário apenas fornecer a posição e orientação do orgão terminal e o sistema se encarrega de obter, através do modelo geométrico inverso do robô, as posições de cada junta.

- Nível de objeto. Neste nível é necessário apenas as especificações relativas ao posicionamento de objetos no interior do volume de trabalho do robô. Deste modo, é necessária a existência de um modelo matemático que represente o ambiente de trabalho no qual o robô se encontra.

- Nível de objetivo. Neste nível a tarefa não é realmente descrita, mas definida como, por exemplo : montar as peças A, B e C. Neste caso é necessário, além do conhecimento do modelo do ambiente, um conjunto de dados relativos a uma determinada tarefa

2.1.2.2 - Vantagens da utilização de programação por linguagens

Entre as vantagens da utilização da programação por linguaguens é possível citar:

- Redução do tempo em que o robô está fora da linha de produção, pois ele pode continuar operando enquanto a sua próxima tarefa está sendo programada;

- O programador não necessita entrar em contato com o ambiente de trabalho;

- Integração com sistemas CAD-CAM permitindo uma maior integração entre as fases de projeto e de produção e, consequentemente, reduzindo o tempo do processo de produção;

- Simplificação da programação de tarefas com alto grau de complexidade. As linguagens de programação facilitam a elaboração de tarefas complexas pois, possuem formas analíticas de geracão de trajetórias e análise de dados provenientes de sensores externos, além de várias estruturas de controle etc;

- Segurança na geração de trajetorias. As linguagens podem internamente gerar um modelo geométrico do ambiente de trabalho do robô e, através de algortimos de detecção de colisão, produzir trajetórias seguras e simulá-las por software antes da execução final.

2.2 - Modelagem

2.2.1 - Modelo cinemático

O modelo geométrico de um robô é definido como a função vetorial $\underline{f}$, que exprime um vetor $\underline{x}$ (espaço cartesiano) em função do vetor $\underline{\theta}$ (coordenadas angulares):

$$\underline{x} = \underline{f}(\underline{\theta}) \qquad (2.1)$$

Como já visto, para se determinar o modelo geométrico é necessário apenas o calculo da matriz de passagem homogênea que relaciona o deslocamento espacial do sistema de coordenadas da garra ao sistema de coordenadas de referência, figura 2.3, equação (A.3) Anexo A.

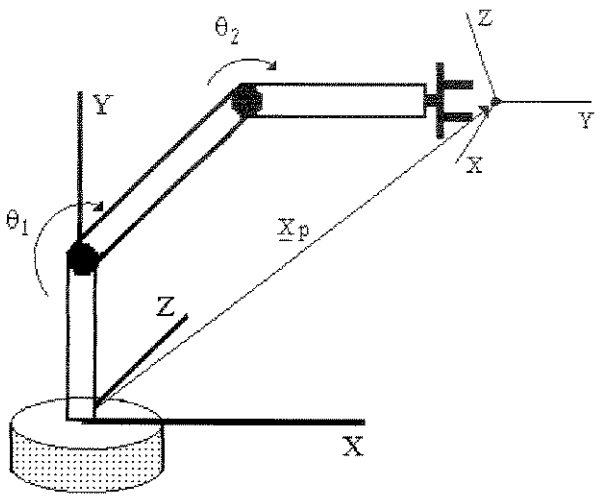


Figura 2.3: O Modelo geométrico descreve a posição e orientação da garra em função das variaveis de juntas.

O modelo geométrico é um modelo básico para a obtenção do modelo dinâmico, para a implementação de estratégias de controle de trajétorias e para simulação cinemática.

2.2.2 - Modelo cinemático inverso

Como anteriormente dito, a coordenação de movimentos, que consiste na obtenção de um movimento de referência (angular), para cada junta, para um dado movimento de referência da garra (cartesiano), é expressa matematicamente pela inversão do modelo geométrico, isto é :

$$\underline{\theta} = \underline{f}(\underline{\theta})^{-1}(\underline{x}) \qquad (2.2)$$

A função $\underline{f}$ é não linear e composta de soma de produtos das coordenadas generalizadas das ligações de translação e de somas e produtos de senos e cossenos das coordenads generalizadas das ligações de rotação. Por isso, a sua inversão é em geral não trivial.

Como $\underline{f}$ é não linear, não se pode garantir a existência e/ou a unicidade de uma função inversa $\underline{f}^{-1}$. No caso geral, só se pode determinar o número máximo de prováveis soluções.

Os métodos de solução do problema da inversão do modelo geométrico são:

- *Métodos analíticos* Estes métodos conduzem à obtenção de todas as soluções. Estes métodos não são gerais, isto é, a inversão analítica não é trivial e além disso não há garantia de que seja possível fazê-la para um robô qualquer. Os métodos analíticos são adequados para robôs simples, isto é, aqueles que possuem um grande número de parâmetros de Denavit-Hartemberg nulos.

- *Métodos numéricos iterativos* Estes métodos convergem para uma solução possível entre todas as existentes, são de caráter geral e, com o atual desenvolvimento dos microcomputadores, a utilização destes métodos em tempo real é viável.

Existem diversos métodos numéricos iterativos, entre eles o *método recursivo*, figura 2.4, que utiliza ao cálculo do modelo geométrico direto e da matriz Jacobiana inversa.

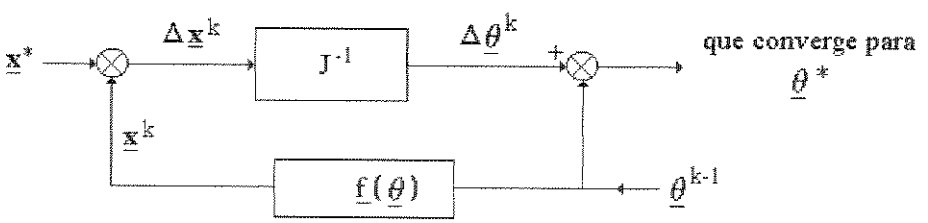


Figura 2.4: Utilização do modelo geométrico direto e da matriz Jacobiana inversa para determinar uma configuração $\underline{\theta}^*$ correspondente a uma situação desejada $\underline{x}^*$.

Neste trabalho optou-se pela utilização do método recursivo para a implementação de um software para a geração de trajetórias pois, é de fácil implementação.

2.2.2.1 - Existencia ou não da solução do modelo geométrico inverso

Para uma tarefa qualquer, definida no espaço operacional (cartesiano) de um robô, são necessários no mínimo seis graus de liberdade para a sua execução. Contudo se existirem obstáculos a se contornar, o número de graus de liberdade pode aumentar. Denotando-se m o número de graus de liberdade necessários à execução de uma tarefa, se o manipulador tem n graus de liberdade, pode-se encontrar, com relação às soluções do modelo inverso, três situações:

i - se $n < m$ não existe solução

ii - se $n = m$ pode ocorrer dois casos

- Existe um número finito de soluções fora das configurações singulares, figura 2.5.

- Existe um número infinito de soluções nas configurações singulares.

iii - Se $n > m$, têm-se uma redundância e existe um número infinito de soluções. Para se obter uma solução única, introduz-se restrições no espaço operacional, ou usa-se um método que minimize determinado critério de desempenho.

Obviamente, se $\underline{x}$ está fora do volume de trabalho do manipulador, não existe solução.

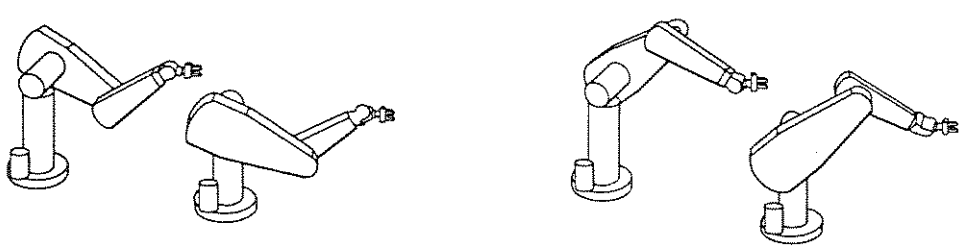


Figura 2.5: Quatro configurações, $\underline{\theta}$, que constituem uma solução para a situação $\underline{x}$ correspondente.

2.3 - Matriz Jacobiana

A matriz Jacobiana, utilizada no método recursivo para o cálculo do modelo cinemático inverso, é uma forma multimensional da derivada e relaciona a velocidade no espaço de juntas à velocidade no espaço cartesiano isto é:

$$\Delta \underline{x} = J \, \Delta \underline{\theta} \qquad (2.3)$$

Uma técnica para determinar o Jacobiano de um robô de seis graus de liberdade é fazendo uso das matrizes de transformações que definem a geometria do robô. Esta técnica é apresentada no Anexo B.

2.3.1 - Matriz Jacobiana inversa

A matriz Jacobiana inversa pode ser obtida através de dois métodos:

- *Inversão simbólica.* Se deixarmos a matriz Jacobiana em sua forma original, forma simbólica, pode-se encontrar a inversa usando a algebra. Mas a complexidade do Jacobiano torna este procedimento muito difícil ou até mesmo impossível.

- *Inversão numérica.* Para cada instante, a configuração do robô define um conjunto de variáveis de juntas, deste modo então a matriz Jacobiana, em cada instante, é uma matriz número ca. A literatura em análise número ca apresenta diversas técnicas para a inversão de matrizes. Tais técnicas não devem apenas serem rápidas, mas também, retornarem respostas precisas.

Pode-se notar que o modelo geométrico do robô é de grande importância pois, através dele pode-se calcular a matriz Jacobiana do robô e, além disso, ele é utilizado diretamente na malha de controle para a geração de uma trajetória ponto a ponto em tempo real.

Neste trabalho optou-se pela utilização do método recursivo para a geração de uma trajetória por ser de facil implementação.

No capítulo seguinte serão apresentados os métodos que serão estudados para a inversão da matriz Jacobiana. Estes métodos foram escolhidos devido ao baixo número de operações aritméticas que utilizam, reduzindo deste modo o tempo computacional gasto, e por serem amplamente usados na literatura mas, não impede que outros métodos sejam utilizados.

CAPÍTULO 3

Métodos de Resolução do Problema Inverso

No capítulo anterior mostrou-se a necessidade da inversão da matriz Jacobiana para a obtenção do modelo cinemático inverso, através do método numérico recursivo.

Deste modo existe a necessidade da utilização de métodos numéricos para a inversão da matriz Jacobiana. Neste trabalho foram implementados os métodos de Gauss, utilizado para o calculo da matriz inversa de J; Greville, utilizado para o calculo da matriz pseudoinversa de J, e Miss no qual não há necessidade da inverão da matriz Jacobiana para se obter $\underline{\theta}$.

Estes métodos são ditos métodos diretos e em princípio, produzirão uma solução exata (desprezando os erros de arredondamento), se houver, em um número finito de operações. Entretanto as soluções podem perder o significado caso o sistema linear seja mal-condicionado.

Neste capítulo serão apresentados os métodos e uma breve discusão sobre sistemas lineares mal-condicionados.

3.1 - Métodos

3.1.1 - Método de Eliminação Gauss

É bem conhecido que se A (n × n) é não singular (Kreysizig, 1983), então existe uma única matriz B, que é chamada de inversa de A, tal que A B = B A = I, onde I é a matriz identidade. Se A é singular ou retangular, não existe a inversa de A. Um sistema é dito singular quando:

$$\text{Det}\,(A) = 0 \tag{3.1}$$

Para determinar a matriz inversa de A será utilizado o método de eliminação de Gauss, que consiste em transformar um sistema linear $A\underline{x} = \underline{b}$ em um sistema triangular superior.

O procedimento para a determinação da inversa, utilizando o método de Gauss, é apresentado no Anexo C. A matriz inversa de J será utilizada para determinar o sistema:

$$\Delta\underline{\theta} = J^{-1}\,\Delta\underline{x} \tag{3.3}$$

3.1.3 - Pseudoinversa

Em muitos casos, a solução de um sistema de equações lineares existe, mesmo quando a inversa da matriz não existe. Este problema e muitos outros podem ser resolvidos através do conceito da pseudoinversa, ou inversa generalizada (Nashed, 1976), da matriz.

A inversa generalizada deve atender a algumas das seguintes propriedades para obter sucesso:

i) deve reduzir-se a A^{-1} se A é não singular;

ii) deve sempre existir;

iii) deve possuir algumas das propriedades da inversa (ou modificações destas)

iv) quando usadas no lugar da inversa, deve ser capaz de proporcionar respostas sensíveis para questões importantes tal como; consistência das equações, ou soluções de mínimos quadrados.

Moore e Penrose definiram a pseudoinversa de uma matriz, A^+,como a única solução para:

$$\begin{aligned} A \times A^+ \times A &= A \\ A^+ \times A \times A^+ &= A^+ \\ (A \times A^+)^t &= A \times A^+ \\ (A \times A^+)^t &= A^+ \times A \end{aligned} \qquad (3.3)$$

Neste trabalho a matriz pseudoinversa foi obtida através do método de Greville e será utilizada para determinar o sistema:

$$\Delta\underline{\theta} = J^+ \Delta\underline{x} \qquad (3.4)$$

3.1.3 - Métodos de Miss

Miss (Miss, 1991), em seu primeiro método, fez uso da norma Euclidiana, equação 3.5, e de sua derivada para garantir que o erro na solução, $\underline{x}$, de um sistema linear seja mínimo.

$$S = (\Sigma (A_{ij} x_j - b_i)^2)^{1/2} \qquad (3.5)$$

Em seu segundo método, para se obter o mínimo absoluto foi utilizado o gradiente pois, a mais curta distância entre dois pontos de uma superficie é definida pelo seu gradiente.

A diferenciação da função S necessária para a avaliação do gradiente, em que para o caso da primeira versão foi usado para determinar a solução, é empregado na segunda versão como a condição anterior para o ajuste do vetor $\underline{x}$.

3.2 - Número de operações aritméticas

O número de operações aritméticas necessárias para cada método é apresentado na tabela 3.1.

Método	**Número de operações aritméticas**
Gauss	$(4n^3 + 9n^3 - 7n) / 6$
Greville	$3mn + 4n$
Miss (método 1)	$5mn + 3n$
Miss (método 2)	$4mn + 6m + 3n$

Tabela 3.1: Número de operações aritméticas para os métodos apresentados.

onde **m** é igual ao número de linhas e **n** é igual ao número de colunas de uma matriz.

Estes números de operações foram obtidos das bibliografias citadas anteriormente.

3.3 - Sistemas Mal-Condicionados

Como foi dito no inicio do capítulo, para sistemas mal-condicionados os erros de arredondamento podem levar a obtenção de soluções não exatas do sistema linear, mesmo se o método utilizado para a resolução for exato. Deste modo é recomendavel o refinamento da solução obtida.

Um sistema linear é dito bem condicionado (Kreysizig, 1983) se pequenos erros nos coeficientes, ou arredondamentos, durante o processo de solução causam pequenas pertubações na solução final, e é dito mal-condicionado se as pertubações sobre a solução final é grande.

Considere um sistema linear com duas equações e duas variáveis desconhecidas. O mal condicionamento, figura 3.1 (a), ocorre se, e somente se, as retas (soluções) que representam as equações são quase paralelas, deste modo o ponto de intersecção das retas (a solução do sistema) não é bem definido. Já em um sistema bem-condicionado o ponto de intersecção das retas é nítido, figura 3.3 (b).

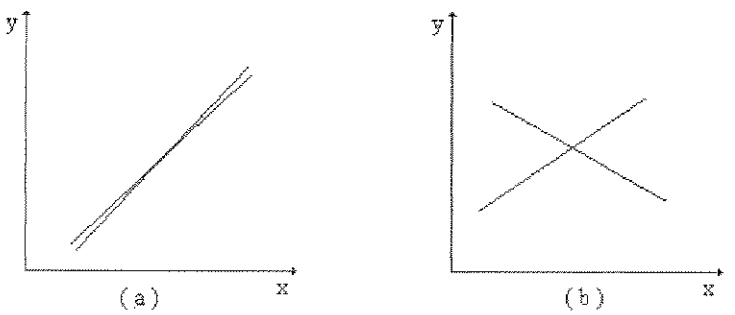


Figura 3.1 : Solução de um sistema linear com duas equações e duas variáveis desconhecidas: sistema mal-condicionado (a) e bem-condicionado (b).

Para grandes sistemas, a situação é similar, mas geometricamente não há intersecção de retas. No espaço tridimencional existe a intersecção de planos, e em espaços de outras dimensões existe a intersecção de hiperplanos.

3.3.1 - Número de Condição

Uma maneira simples de detectar o mal-condicionamento em um sistema linear pode ser feita mudando-se deliberadamente alguns coeficientes e determinar o grau de mudança na solução.

Outra forma de determinar o mal-condicionamento de um sistem é através do calculo do número de condição (Atkinson, 1988) da matriz A que é definido como:

$$\text{Cond}\,(\,A\,) = \| A \| \, \| A^{-1} \| \qquad (3.6)$$

onde $\| \ \|$ é a norma da matriz A.

Não existe uma linha de divisão nítida entre um sistema bem-condicionado e um mal-condicionado. Em geral para valores do número de condição compreendidos entre 1 e 90 (Kreysizig, 1983), não há razão para preocupação e portanto, o sistema é bem-condicionado. Mas, para valores maiores o sistema pode ser considerado mal-condicionado e deste modo, o vetor solução obtido não é confiavel. No capítulo 4 será apresentado um algorítmo para a correção do vetor solução de um sistema mal condicionado.

Os métodos apresentados neste capítulo serão implementados juntamente com o algorítmo para a geração de trajetórias e após isto, será estudado o comportamento (número de iterações e oscilações entre outros) do método utilizado sobre a trajetória gerada.

Estes métodos, como já mencionado no capítulo anterior, foram escolhidos devido ao baixo número de operações aritméticas que utilizam e por serem amplamente usados na literatura mas, não impede que outros métodos sejam utilizados.

No capítulo seguinte serão descritos matematicamente os métodos aqui apresentados.

CAPÍTULO 4

Algoritmos Implementados

No capítulo anterior foi apresentada uma breve introdução dos métodos que serão implementados no programa de geração de trajetórias devido a necessidade da obtenção da matriz inversa da matriz Jacobiana. Estes métodos foram escolhidos devido ao baixo número de operações aritméticas que utilizam, reduzindo deste modo o tempo computacional gasto, e por serem amplamente usados na literatura mas, não impede que outros métodos sejam utilizados.

Neste capítulo esses métodos serão descritos matematicamente e apresentados os fluxogramas para implementação. Além disso, será apresentado o algorítmo proposto para a geração de uma trajetória ponto a ponto.

4.1 - Algorítmos para resolução de sistemas lineares

4.1.1 - Método de Eliminação de Gauss

O método de eliminação de Gauss (Dorn, 1972) consiste em transformar a matriz do sistema em uma matriz triangular superior. Desta forma o sistema:

$$
\begin{array}{l}
a_{11}x_1 + a_{12}x_2 + \ldots\ldots\ldots + a_{1j}x_j + \ldots\ldots\ldots + a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots\ldots\ldots + a_{2j}x_j + \ldots\ldots\ldots + a_{2n}x_n = b_2 \\
\quad\vdots \qquad\qquad\qquad\qquad\quad \vdots \qquad\qquad\qquad\qquad\quad \vdots \\
a_{i1}x_1 + a_{i2}x_2 + \ldots\ldots\ldots + a_{ij}x_j + \ldots\ldots\ldots + a_{in}x_n = b_i \\
\quad\vdots \qquad\qquad\qquad\qquad\quad \vdots \qquad\qquad\qquad\qquad\quad \vdots \\
a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots\ldots + a_{nj}x_j + \ldots\ldots\ldots + a_{nn}x_n = b_n
\end{array}
\tag{4.1}
$$

é transformado em:

$$
\begin{array}{l}
x_1 + a_{12}^{(1)}x_2 + \ldots\ldots\ldots + a_{1j}^{(1)}x_j + \ldots\ldots\ldots + a_{1n}^{(1)}x_n = b_1^{(1)} \\
\qquad\quad x_2 + \ldots\ldots\ldots + a_{2j}^{(2)}x_j + \ldots\ldots\ldots + a_{2n}^{(2)}x_n = b_2^{(2)} \\
\quad\vdots \qquad\qquad\qquad\qquad\quad \vdots \qquad\qquad\qquad\qquad\quad \vdots \\
\qquad\qquad\qquad\quad x_j + \ldots\ldots\ldots + a_{in}^{(i)}x_n = b_i^{(i)} \\
\quad\vdots \qquad\qquad\qquad\qquad\quad \vdots \qquad\qquad\qquad\qquad\quad \vdots \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad x_n = b_n^{(n)}
\end{array}
\tag{4.2}
$$

onde os índices superiores dos coeficientes correspondem ao número de modificações efetuadas em cada elemento.

A solução do sistema (4.1) é obtida de modo imediato mediante uma substituição regressiva, pois:

o valor de x_n é obtido diretamente da última equação

o valor de x_{n-1} é obtido levando o valor de x_n na penúltima equação

$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$

o valor de x_2 é obtido levando os valores $x_3, x_4, \ldots, x_n$ na segunda equação

o valor de x_1 é obtido levando os valores $x_2, x_3, \ldots, x_n$ na primeira equação

O problema então consiste em levar o sistema (4.1) à forma do sistema (4.2) por meio de uma sucessiva e conveniente soma de equações que não altera o sistema inicial devido as propriedades da algebra linear.

Como essas somas só modificarão a matriz A e o vetor b, podemos fazê-las na matriz ampliada (n x (n + 1)):

$$A_E = \begin{bmatrix} a_{11} & a_{12} & \ldots\ldots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \ldots\ldots & a_{2n} & a_{2,n+1} \\ . & & & & \\ . & & & & \\ a_{n1} & a_{n2} & \ldots\ldots & a_{nn} & a_{n,n+1} \end{bmatrix} \qquad (4.3)$$

cuja última coluna é o vetor b.

Começa-se as transformações anulando os elementos de g_1 (elementos da coluna 1) abaixo de a_{11}. A isto chama-se de primeira eliminação, o elemento a_{11} é chamado de pivô dessa eliminação, que é realizada da seguinte forma:

-- O primeiro passo da primeira eliminação é dividir l_1 (primeira linha da matriz A_E) pelo pivô, o que dá a nova linha $l_1^{(1)}$.

-- Assim, para anular o elemento a_{21} basta somar l_2 a $l_1^{(1)}$ premultiplicada por $-a_{21}$.

-- De um modo geral, para anular o elemento a_{i1}, (i = 2, 3, ..., n), executamos a operação vetorial:

$$l_i^{(1)} = l_i - a_{i1} l_1^{(1)} \qquad (i = 2, 3, \ldots, n) \qquad (4.4)$$

que, desenvolvida para os elementos de tôdas as colunas de l_i, equivale a:

$$a_{ij}^{(1)} = a_{ij} - a_{i1} a_{1j}^{(1)} \qquad (4.5)$$

onde i = 2,3, ... ,n

j = 1,2, ... ,n+1.

Ao fim da primeira eliminação, a matriz A_E está transformada em:

$$A_E^{(1)} = \begin{bmatrix} 1 & a_{12}^{(1)} & \ldots\ldots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ 0 & a_{22}^{(1)} & \ldots\ldots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ . & & & & \\ . & & & & \\ 0 & a_{n2}^{(1)} & \ldots\ldots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{bmatrix} \qquad (4.6)$$

A segunda eliminação consiste em anular os elementos de $g_2^{(1)}$ (elementos da coluna 2) abaixo de $a_{22}^{(1)}$ denominado pivô dessa eliminação. Para realizá-la fazemos operações análogas da primeira :

-- Divide-se $l_2^{(1)}$ (segunda linha da matriz $A_E^{(1)}$) pelo pivô, obtendo $l_2^{(2)}$.

-- Para anular o coeficiente $a_{32}^{(1)}$ basta somar $l_3^{(1)}$ a $l_2^{(2)}$ premultiplicada por $-a_{32}^{(1)}$.

-- De um modo geral, para anular o elemento $a_{i2}^{(1)}$, (i = 3, ..., n), executamos a operação vetorial

$$l_i^{(2)} = l_i^{(1)} - a_{i2}^{(1)} l_2^{(2)} \qquad (i = 3, ..., n) \qquad (4.7)$$

que, desenvolvida para os elementos de tôdas as colunas de $l_I^{(1)}$, equivale a

$$a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)} a_{2j}^{(1)} \qquad (4.8)$$

onde i = 3, ... ,n

j = 2, ... ,n+1.

Ao fim da segunda eliminação, a matriz A_E está transformada em:

$$A_E^{(2)} = \begin{bmatrix} 1 & a_{12}^{(1)} & \ldots\ldots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ 0 & 0 & \ldots\ldots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ . & & & & \\ . & & & & \\ 0 & 0 & \ldots\ldots & a_{nn}^{(2)} & a_{n,n+1}^{(2)} \end{bmatrix} \qquad (4.9)$$

Podemos agora generalizar o procedimento da primeira e segunda eliminação para uma eliminação genérica k, que consiste em anular os elementos de $g_k^{(k-1)}$ (elementos da coluna k) abaixo de $a_{kk}^{(k-1)}$, denominado pivô da k-ésima eliminação.

-- Inicialmente faz-se a divisão

$$l_k = l_k^{(k-1)} / a_{kk}^{(k-1)}, \qquad (4.10)$$

que desenvolvida elemento por elemento de $l_k^{(k)}$ (k-ésima linha da matriz $A_E^{(k)}$), equivale a:

$$a_{kj}^{(k)} = a_{kj}^{(k-1)} / a_{kk}^{(k-1)} \qquad (4.11)$$

onde $j = k, \ldots\ldots, k+1$

$k = 1, 2, \ldots\ldots, n.$

-- Se k for menor do que n, fazemos as operações vetoriais:

$$l_i^{(k)} = l_i^{(k-1)} - a_{ik}^{(k-1)} l_k^{(k)} \quad (i = k+1, \ldots, n), \qquad (4.12)$$

que desenvolvidas elemento por elemento, e em analogia com a equação (4.7) e (4.8), dá:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} a_{kj}^{(k)} \qquad (4.13)$$

onde $j = k, \ldots\ldots, n+1$

$i = k+1, \ldots\ldots, n$

$k = 1, 2, \ldots\ldots, n-1.$

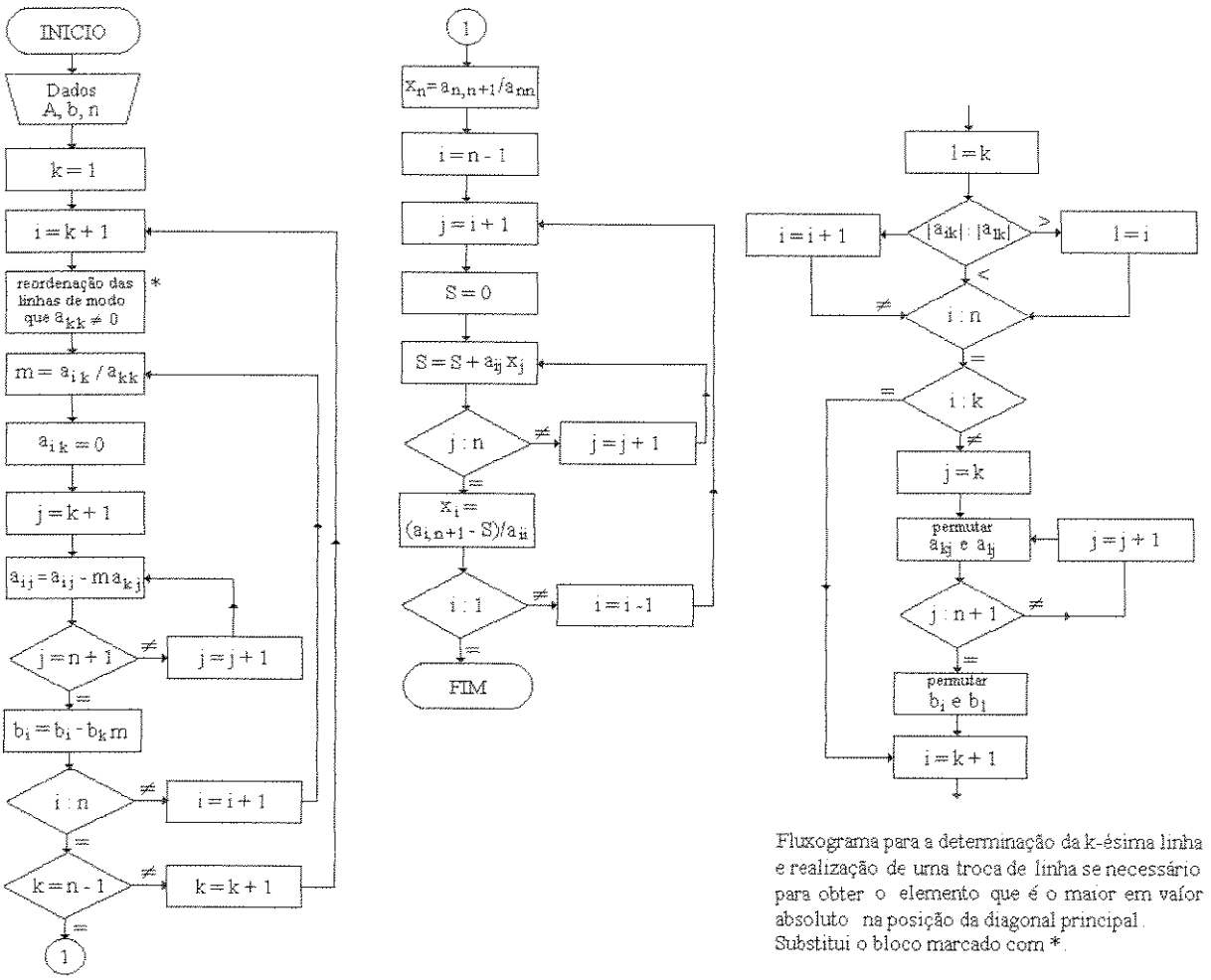
-- Se k for igual a n, o sistema (4.3) está reduzido ao sistema (4.9).

4.1.1.1 - Condensação Pivotal

A princípio o método de Gauss produz soluções exatas, já que o método é exato. No entanto, quando se trabalha com computadores, é utilizado a representação dos números em ponto flutuante (Dorn, 1972). Os inevitáveis erros de arredondamento que ocorrem nas operações com esses números podem comprometer seriamente a solução obtida.

Para minimizar a influência destes erros de arredondamento, utiliza-se a estratégia da condensação pivotal, que consiste em escolher o elemento do pivô no início de cada etapa como sendo o número de maior valor absoluto dentre os elementos da i-ésima coluna que estão na diagonal principal ou abaixo dela.

O fluxograma do método de eliminação de Gauss é apresentado na figura 4.1 juntamente com o fluxograma para a condensação pivotal.



Fluxograma para a determinação da k-ésima linha e realização de uma troca de linha se necessário para obter o elemento que é o maior em valor absoluto na posição da diagonal principal. Substitui o bloco marcado com *.

Figura 4.1: Fluxograma para a determinação da solução do sistema $A\underline{x} = \underline{b}$ pelo método de Gauss.

4.1.2 - Determinação da matriz Pseudoinversa pelo método de Greville

Seja a_k a matriz formada pela k-ésima coluna de A e A_k a matriz formada pelas k primeiras colunas de A.

A matriz pseudoinversa de A_k denotada por A_k^+, é obtida (Gorla, 1984) pela equação:

$$A_k^+ = \begin{bmatrix} A_{k-1}^+ - d_k b_k \\ \hline b_k \end{bmatrix} \tag{4.14}$$

onde

$$\underline{d}_k = A_{k-1}^+ \underline{a}_k \tag{4.15}$$

$$\underline{b}_k = \begin{cases} (\underline{c}_k^t \underline{c}_k)^{-1} \underline{c}_k^t & \text{se } \underline{c}_k = 0 \\ (1 - \underline{d}_k^t \underline{d}_k)^{-1} \underline{d}_k^t A_{k-1}^+ & \text{se } \underline{c}_k \neq 0 \end{cases} \tag{4.16}$$

$$\text{onde } \underline{c}_k = \underline{a}_k - A_{k-1} \underline{d}_k \tag{4.17}$$

$k = 1, 2, 3, \ldots, n.$

A inicialização das iterações depende da primeira coluna de A:

$$A_1^+ = \begin{cases} \vec{0}^t & \text{se } \underline{a}_1 = 0 \\ \underline{a}_1^t / (\underline{a}_1^t \underline{a}_1) & \text{se } \underline{a}_1 \neq 0 \end{cases} \tag{4.18}$$

Este algorítmo é independente do rank e das dimensões da matriz. O fluxograma deste método é apresentado na figura 4.2.

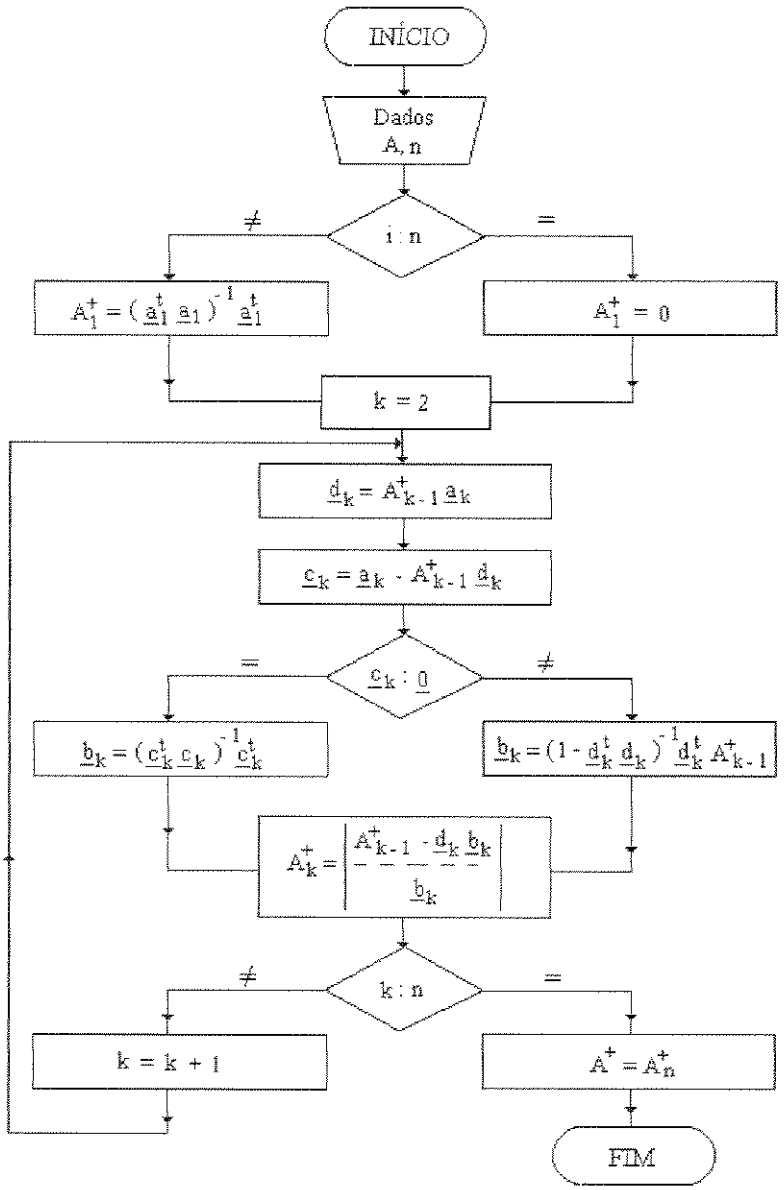


Figura 4.2: Fluxograma para a determinação da matriz Pseudoinversa pelo método de Greville.

4.1.3 - Métodos de Miss

4.1.3.1 - Método 1

Para garantir que o erro na solução, $\underline{x}$, de um sistema linear seja mínimo, isto é:

$$\| A\underline{x} - \underline{b} \| \rightarrow 0 \qquad (4.19)$$

pode-se utilizar a norma Euclidiana (Miss, 1981):

$$(\sum (A_{ij} x_j - b_i)^2)^{1/2} \qquad (4.20)$$

Para isto, seja a função S dada por:

$$S = \sum (A_{ij} x_j - b_i)^2 \qquad (4.21)$$

Derivando-se S, equação 4.21, em função de x_j e igualando a zero, isto é:

$$\delta S / \delta x_j = 0 \qquad (4.22)$$

pode-se obter a solução ótima. Assim, após a derivação, tem-se:

$$2 \sum_i (A_{ij}^2 x_j - A_{ij} b_i) = 0 \qquad (4.23)$$

Resolvendo-se a equação (4.23) para x_j , tem-se :

$$x_j = \sum_i A_{ij} b_i / \sum_i A_{ij}^2 \qquad (4.24)$$

que é o valor ótimo para a solução do sistema linear.

4.1.3.2 - Método 2

Outra maneira de se obter o mínimo absoluto (Miss, 1981) pode ser obtida utilizando o gradiente. Considere uma função V de *n* variáveis que define uma superfície em um espaço dimensional (*n* + 1). A mais curta distância entre dois pontos desta superfície é definida pelo seu gradiente, que é igual a soma das diferenciais parciais $\delta V / \delta r_j$ multiplicado pelos vetores unitários k_j .

A diferenciação da função S necessária para a avaliação do gradiente, em que para o caso da primeira versão foi usado para determinar a solução, é empregado para determinar a derivada parcial $\delta S / \delta x_j$ como a condição anterior para o ajuste do vetor $\underline{x}$. Deste modo, obtém-se:

$$x_j = \tau (\delta S / \delta x_j) = \tau (- 2 \sum_i b_i A_{ij}) \qquad (4.25)$$

$$\Delta \underline{b} \Leftarrow [\Delta b_i - \Sigma (A_{ij} x_j)] \qquad (4.26)$$

$j = 1 (1) n$

$i = 1 (1) m.$

O algorítmo definido por este procedimento apenas consegue a convergência requerida se a constante de proporcionalidade (para acelerar a convergência), τ, é ajustado em cada passo por um valor particular que é dado por:

$$\tau = [\Sigma_j (\Delta b_i \Sigma_i (A_{ij} \delta S / \delta x_j)) / \Sigma_j ((\Sigma_i (A_{ij} \delta S / \delta x_j))^2)] 0.95 \qquad (4.27)$$

onde 0.95 fator de segurança para o qual o algoritmo desenvolve ótimas qualidades de convergência.

4.2 - Refinamento da solução para sistemas Mal-Condicionados

O residuo ($\underline{r}$) do vetor solução é definido como a diferença entre os vetores $\underline{b}$ e $A \underline{x}_o$, então:

$$\underline{r} = \underline{b} - A\underline{x}_o \qquad (4.28)$$

onde $\underline{x}_o$ é a solução não exata do sistema.

Seja

$$\underline{e} = \underline{x} - \underline{x}_o \tag{4.29}$$

onde $\underline{x}$ é a solução exata do sistema e $\underline{e}$ o vetor erro do sistema. Desde que $A\,\underline{x} = \underline{b}$, e usando as equações (4.28) e (4.29), tem-se:

$$\underline{r} = A\,\underline{e} \tag{4.30}$$

Utilizando a equação (4.30) pode-se calcular o vetor erro do sistema:

$$\underline{e} = A^{-1}\,\underline{r} \tag{4.31}$$

Utilizando as equações (4.28), (4.29) e (4.31) em um processo iterativo, figura 4.3, pode-se obter um vetor solução $\underline{x}_o$ tão próxima da solução exata quanto se desejar.

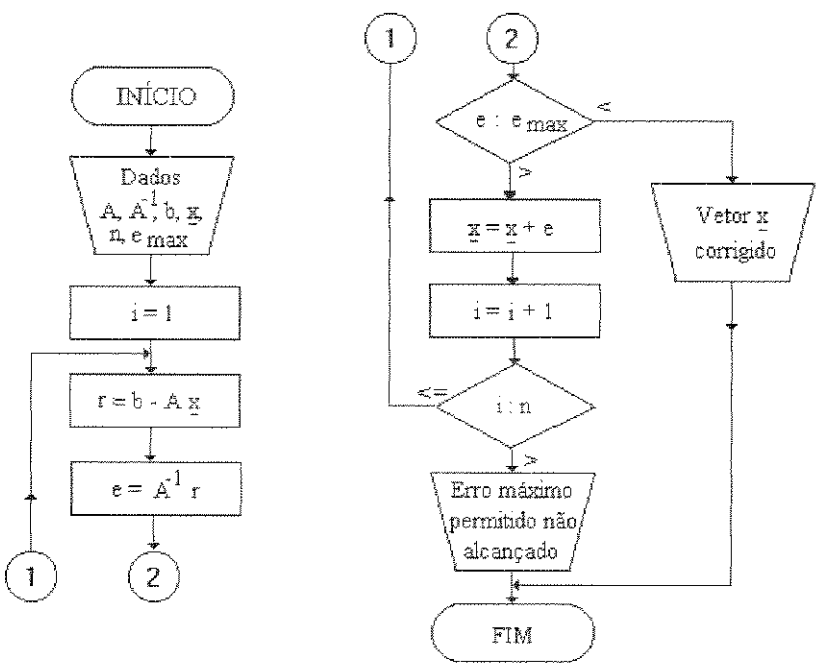


Figura 4.3: Fluxograma para a correção do vetor solução de um sistema mal condicionado.

4.3 - Algoritmo proposto para a geração de uma trajetória ponto a ponto

O algoritmo proposto é apresentado na figura 4.4, existindo quatro critérios para os quais as iterações param. São eles:

Erro máximo permitido. Este critério utiliza um erro máximo permitido para a posição e para a orientação. O erro de posição ($e_{r\,p}$)é obtido através da expressão:

$$e_{r\,p} = \Sigma \left(p_d (i) - p_a (i) \right) \qquad (4.32)$$

onde p_d é a posição final desejada e p_a é a posição atual do elemento terminal do robô. O erro de orientação ($e_{r\,p}$)é obtido utilizando-se o conceito do produto escalar entre dois vetores e é dado por:

$$\begin{aligned} e_{r\,o} = \; & \cos^{-1}((\underline{n}_d * \underline{n}_a) / (\| \underline{n}_d \| * \| \underline{n}_a \|)) + \\ & \cos^{-1}((\underline{s}_d * \underline{s}_a) / (\| \underline{s}_d \| * \| \underline{s}_a \|)) + \\ & \cos^{-1}((\underline{a}_d * \underline{a}_a) / (\| \underline{a}_d \| * \| \underline{a}_a \|)). \end{aligned} \qquad (4.33)$$

onde os vetore $\underline{n}$, $\underline{s}$ e $\underline{a}$ são os vetores ortonormais que descrevem a orientação do elemento terminal do robô

Número de iterações. Este critério utiliza um número máximo de iterações, N, no caso do sistema não convergir para a posição e orientação desejada.

Final do limite físico da junta. Este critério utiliza o máximo percurso para o qual uma junta pode atuar (cada junta possui um limite físico próprio).

Teste do "Rank" da matriz. Utilizado apenas para o metodo de inversão de Gauss. Caso o "rank" da matriz seja menor que o número de linhas da mesma as iterações param pois o sistema é indeterminado.

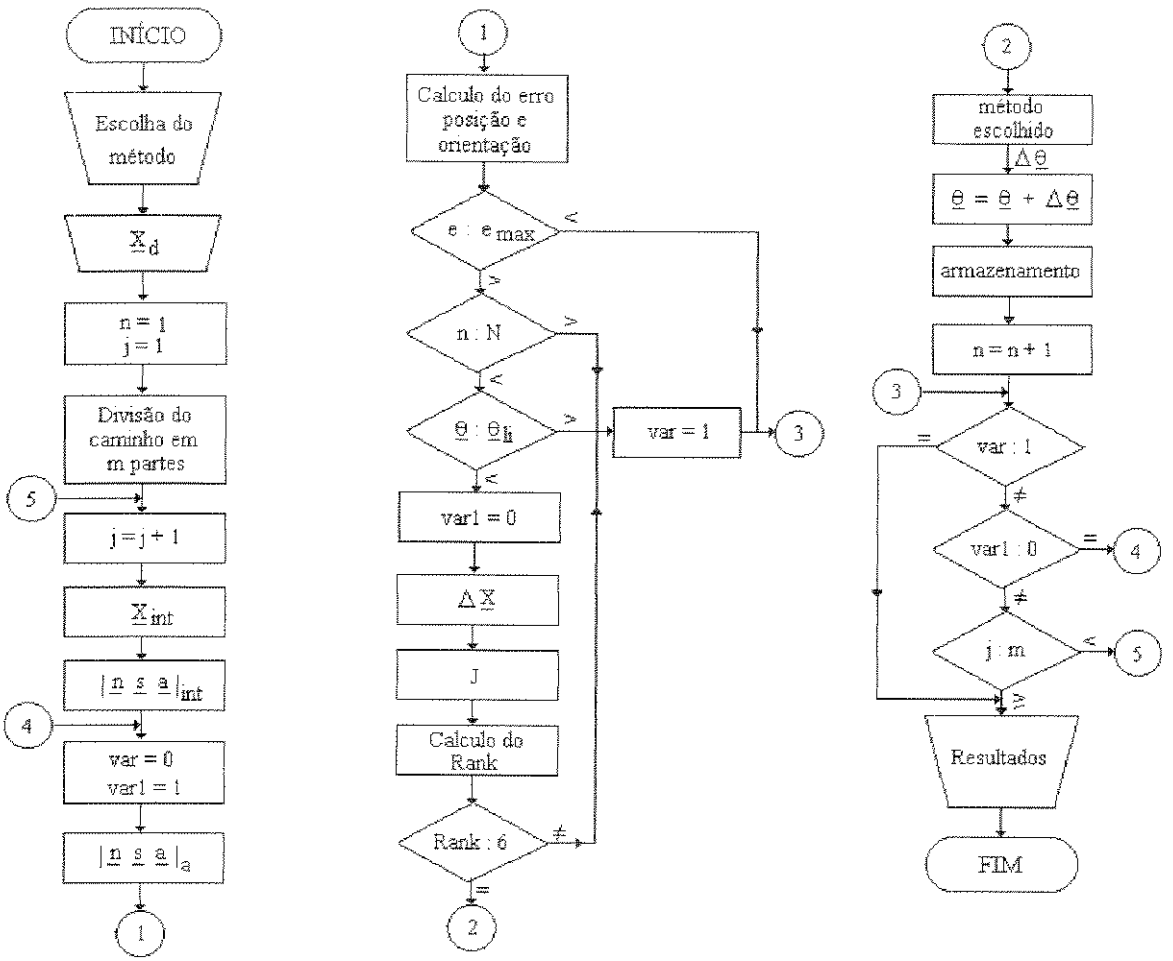


Figura 4.4: Algorítmo proposto para a geração de uma trajetória ponto a ponto.

4.3.1 - Divisão do Caminho

Neste algorítmo utiliza-se uma variável, *m*, cujo objetivo é a de dividir o caminho desejado em *m* partes, figura 4.5.

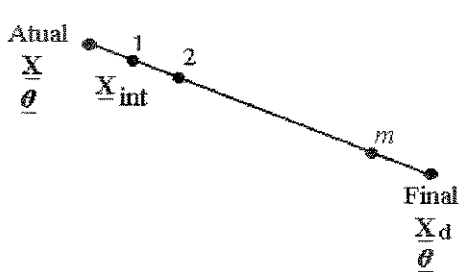


Figura 4.5: Divisão do caminho requerido em *m* partes.

Onde $\underline{X}_{int}$ representa a posição intermediaria.

Isto é realizado porque este método utilizado apenas funciona para pequenos deslocamentos.

Deste modo então, o algorítmo calcula a trajétoria da posição inicial até a posição intermediaria e após ela ser atingida inicia-se o calculo da trajétoria desta posição até a posição intermediaria posterior. Isto é realizado até que a posição final desejada seja alcançada.

Neste capítulo foram desenvolvidos matematicamente os métodos que serão implementados para a inversão da matriz Jacobiana. Estes métodos foram escolhidos, como já mencionado anteriormente, devido ao baixo número de operações aritméticas necessários e por serem amplamente usados na literatura mas, não impede que outros métodos sejam utilizados.

No capítulo seguinte são realizadas simulações com o objetivo de testar o algoritmo de geração de trajetórias, apresentados neste capítulo.

Além disso, será estudado o comportamento das curvas obtidas em relação ao aumento do valor da variável *m*.

CAPÍTULO 5

Implementação Final e Resultados Obtidos

5.1 - Implementação

Com o objetivo de testar o algorítmo de geração de trajetórias e estudar a influência da variável interna *m*, descrito no capítulo 4, foram realizadas simulações e observado o comportamento das curvas geradas. Os valores da posição e orientação (config.) desejadas utilizadas são apresentadas na tabela 5.1.

Além disso, foram plotados os caminhos que a garra do manipulador descreveu no espaço para cada simulação com o objetivo de observar se o caminho seguido pela garra apresentava oscilações, o que não é desejável.

config.	X (mm)	Y (mm)	Z (mm)	Psi (graus)	Teta(graus)	Phi(graus)
1	800.0	0.0	933.1	21.0	58.0	90.0
2	776.9	0.0	700.0	25.0	63.0	75.0
3	776.9	456.0	933.1	85.0	62.0	14.0
4	250.0	-45.0	450.0	45.0	62.0	14.0
5	800.0	0.0	933.1	21.0	58.0	90.0
6	800.0	0.0	600.0	21.0	58.0	91.0
7	776.9	0.0	600.0	21.0	58.0	90.0
8	776.9	0.0	933.1	21.0	58.0	91.0
9	458.0	658.0	521.0	52.0	14.0	62.0

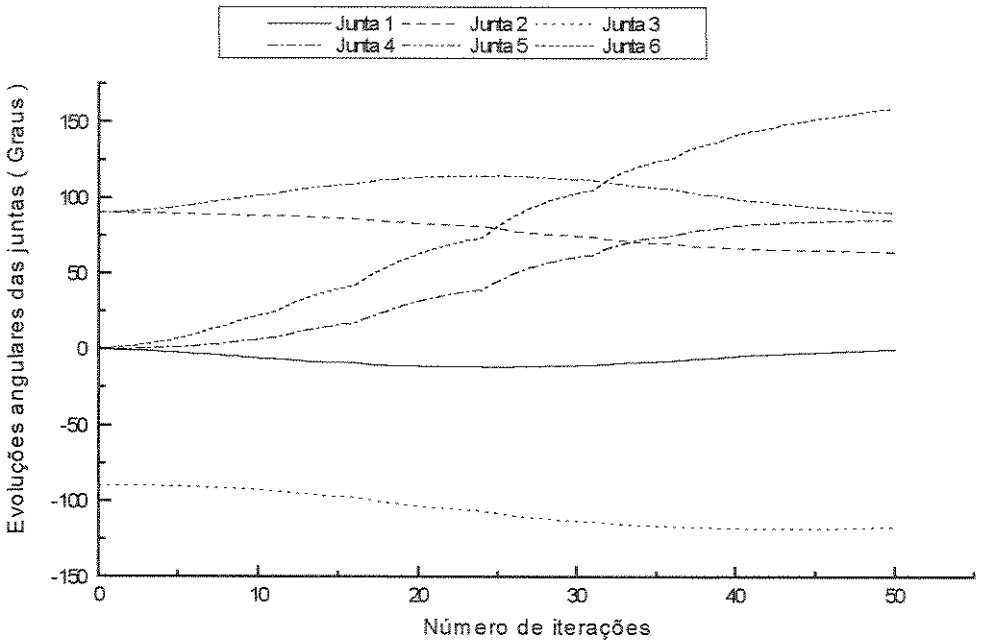
Tabela 5.1 : Posição e orientação desejadas, $\underline{X}_d$.

As configurações de 1 a 4 e 9 representam trajetórias lineares enquanto as configurações de 5 a 7 representam um movimento composto para a geração de um retângulo.

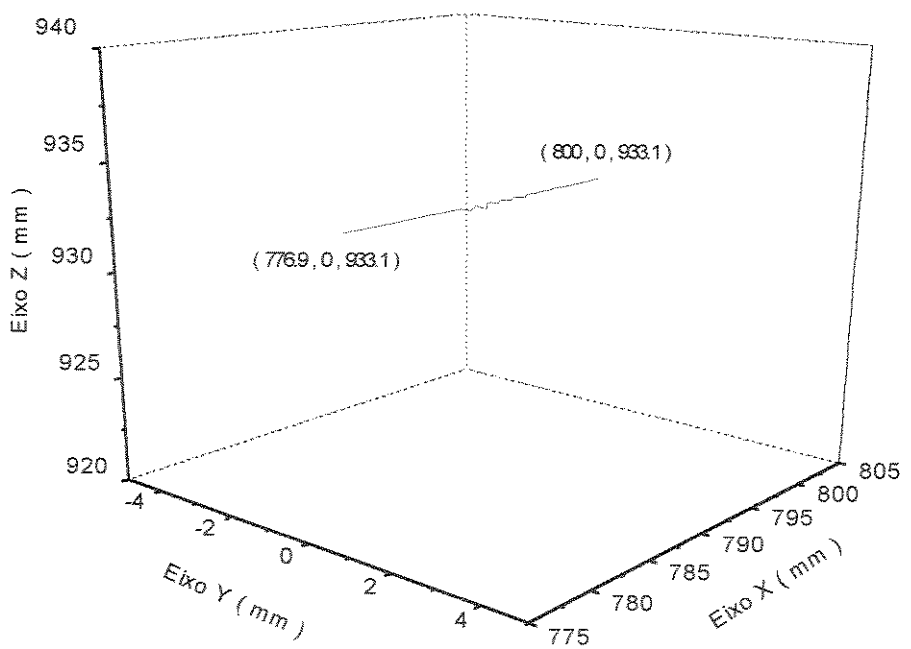
5.2 - Apresentação gráfica dos resultados obtidos

Os valores da variável *m* utilizados nas simulações foram 20, 50 e 70 divisões. Serão apresentados a seguir os gráficos obtidos para cada uma destas configurações e posteriormrnte serão apresentadas, no item 5.3, uma interpretação dos resultados obtidos e um estudo do erro é apresentado no Anexo E.

A posição e orientação inicial (coordenadas cartesianas), para as simulações apresentadas a seguir, são dadas por (776.9, 0, 933.1, 0, 90 ,0) que corresponde a uma posição angular (coordenadas angulares), em graus, de (0, 90, -90, 0, 90, 0).

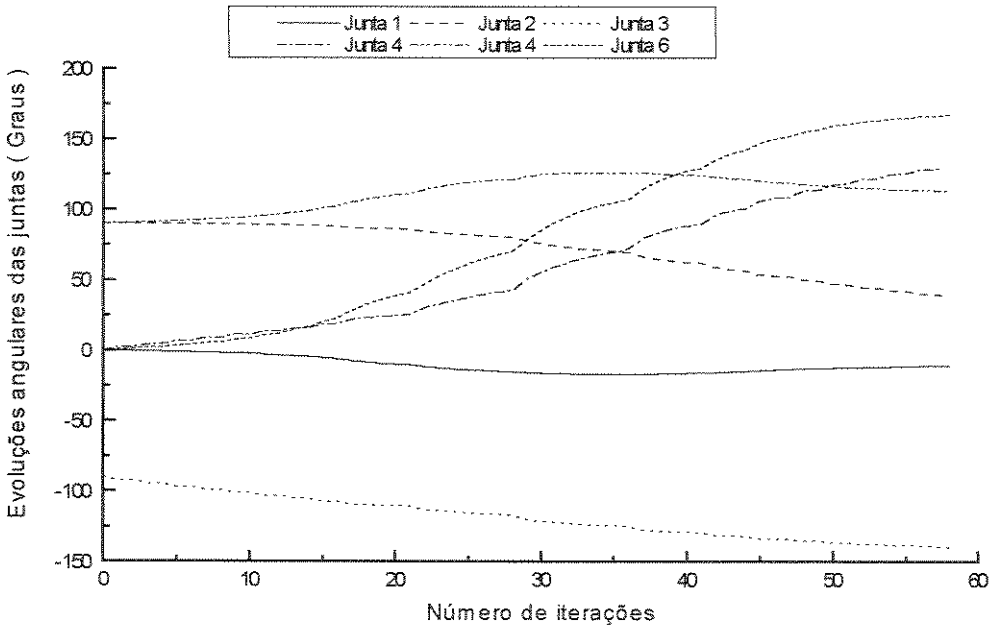


(a) Evoluções angulares das juntas.

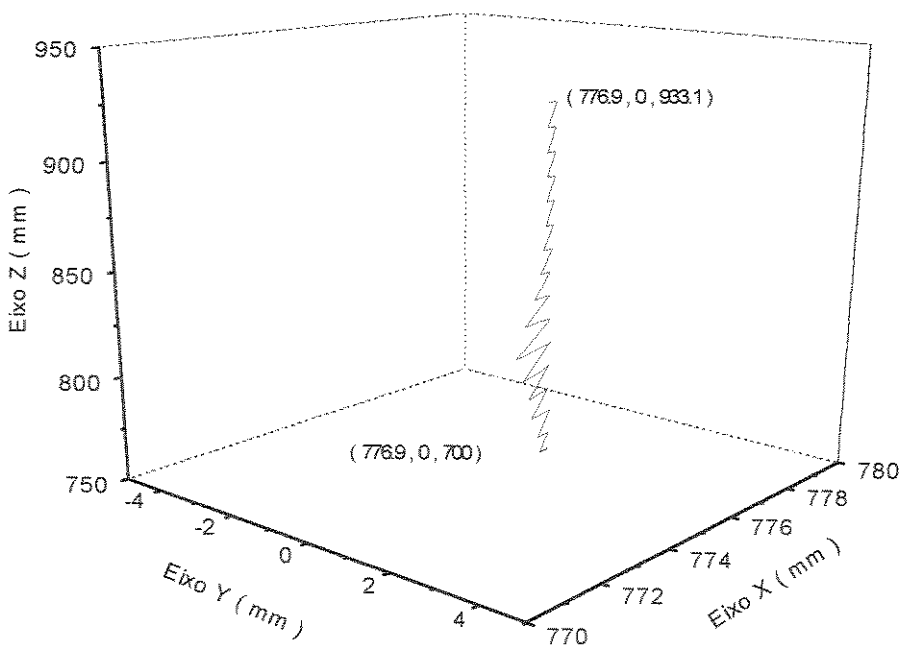


(b) Evolução espacial da garra.

Figura 5.1 : Configuração 1 (800,0,933.1,21,58,90) utilizando os métodos de Gauss e Greville para m igual a 20 divisões.

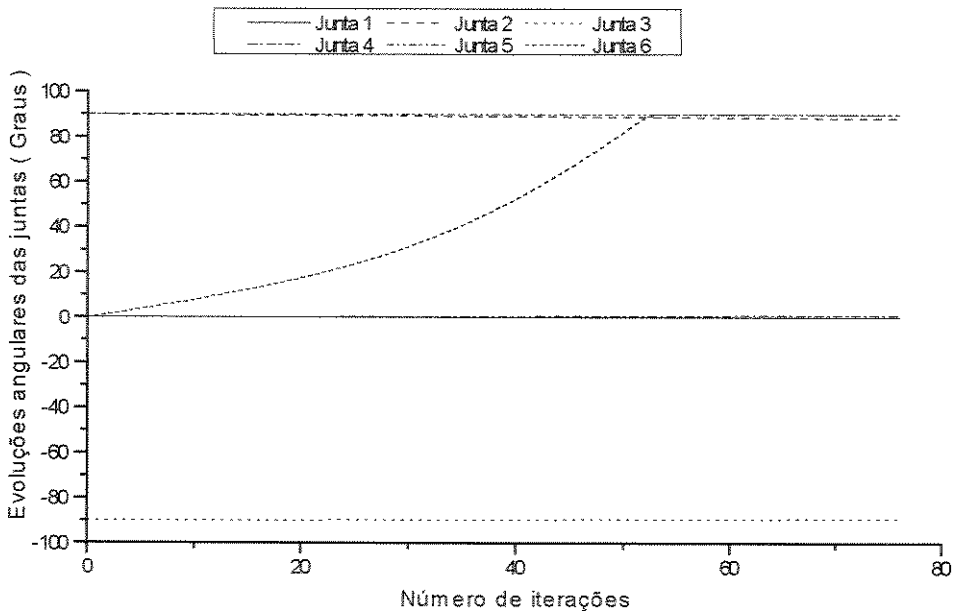


(a) Evoluções angulares das juntas.

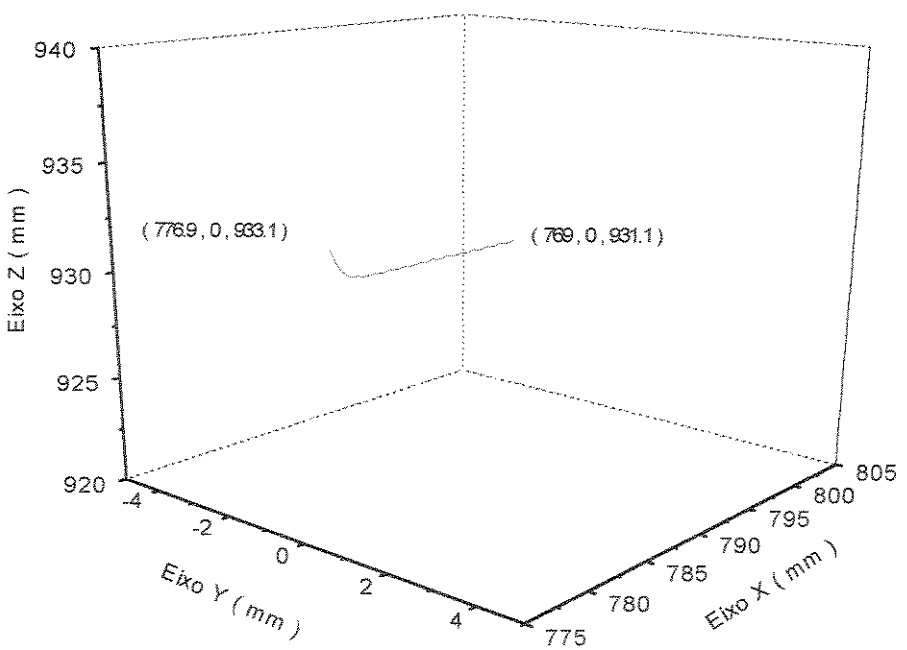


(b) Evolução espacial da garra.

Figura 5.2 : Configuração 2 (776.9,0,700,25,63,75) utilizando os métodos de Gauss e Greville para *m* igual a 20 divisões.

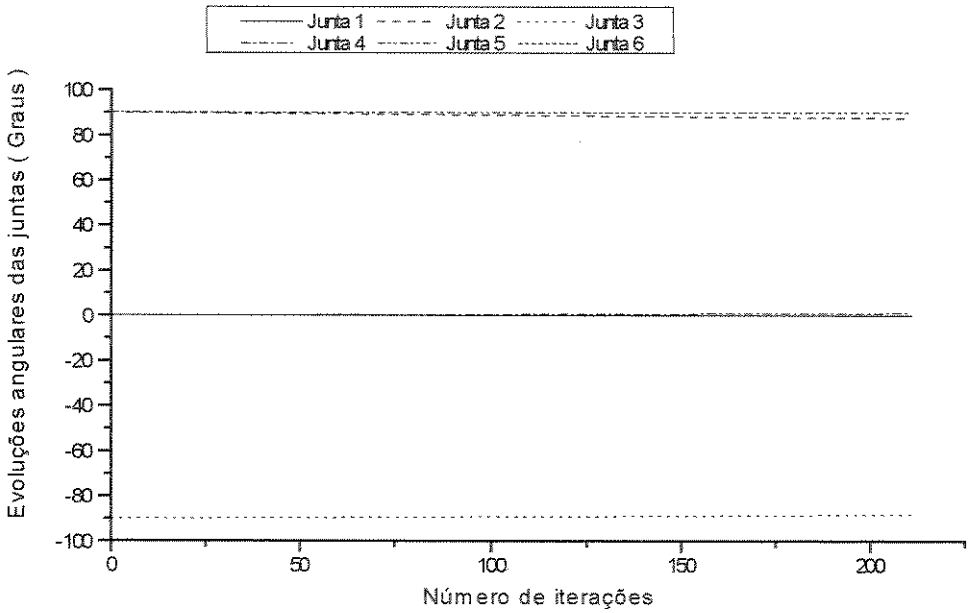


(a) Evoluções angulares das juntas.

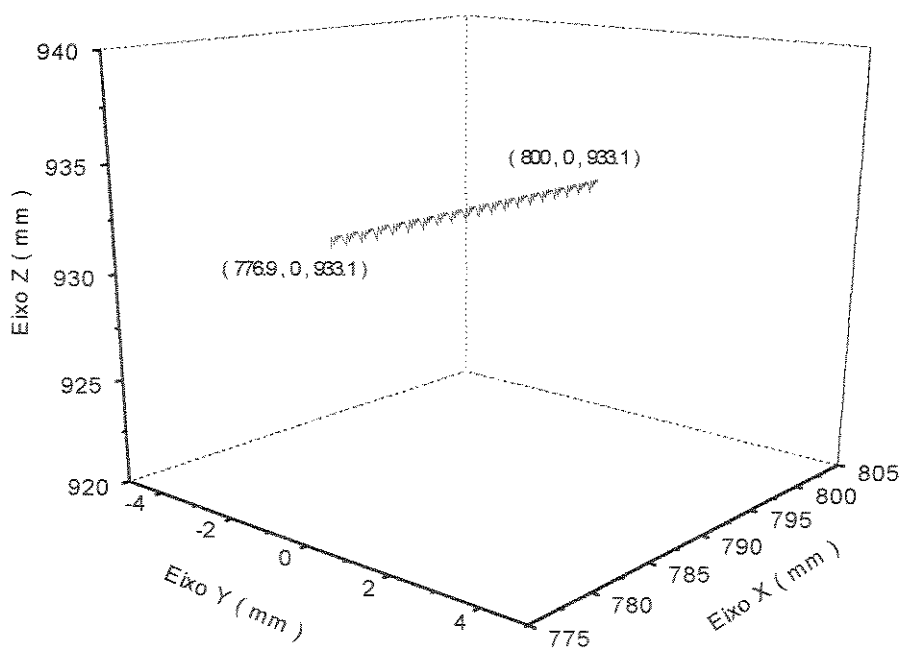


(b) Evolução espacial da garra.

Figura 5.3 : Configuração 1 (800,0,933.1,21,58,90) utilizando o primeiro método de Miss para *m* igual a 20 divisões.

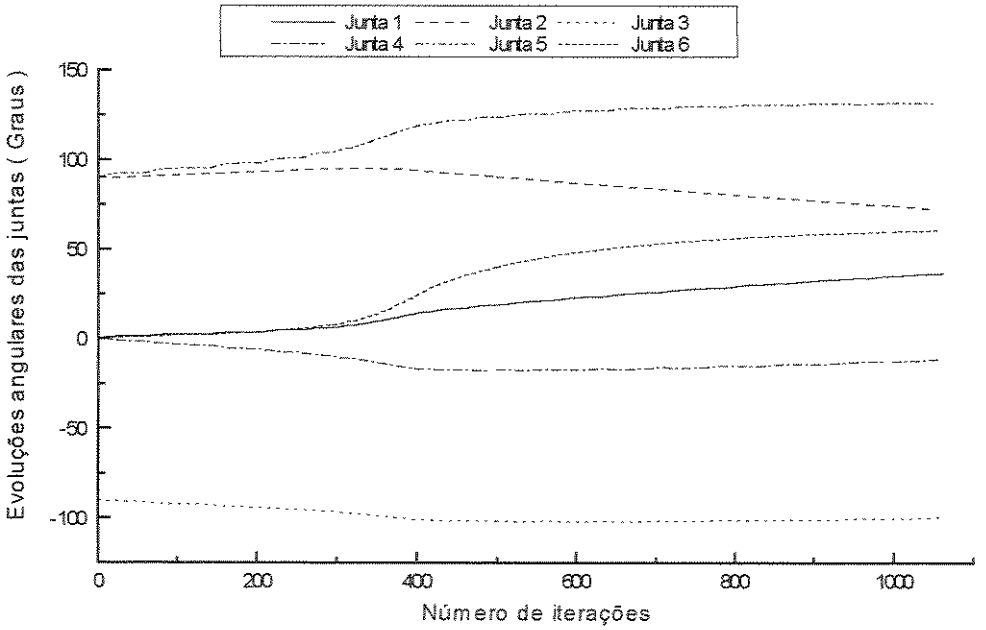


(a) Evoluções angulares das juntas.

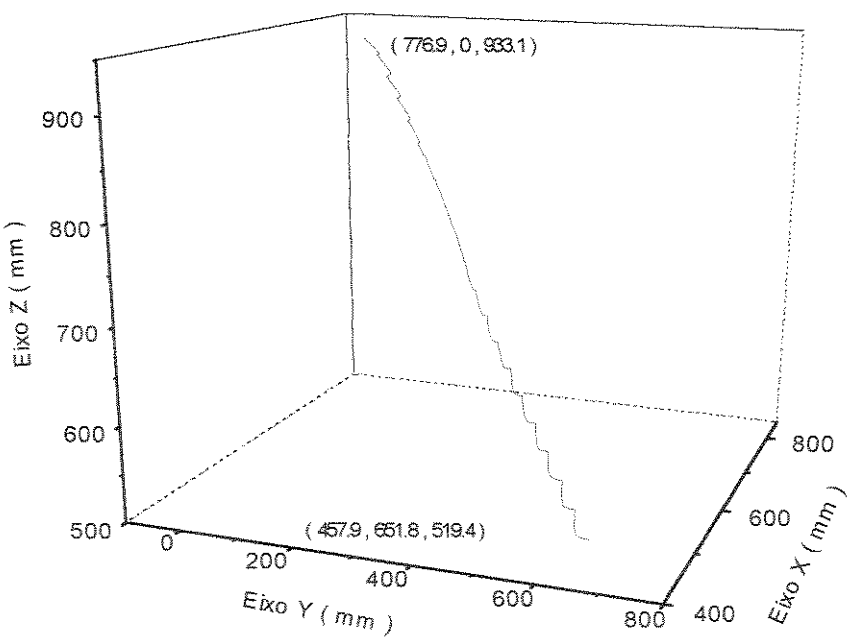


(b) Evolução espacial da garra.

Figura 5.4 : Configuração 1 (800,0,933.1,21,58,90) utilizando o segundo método de Miss para *m* igual a 20 divisões.

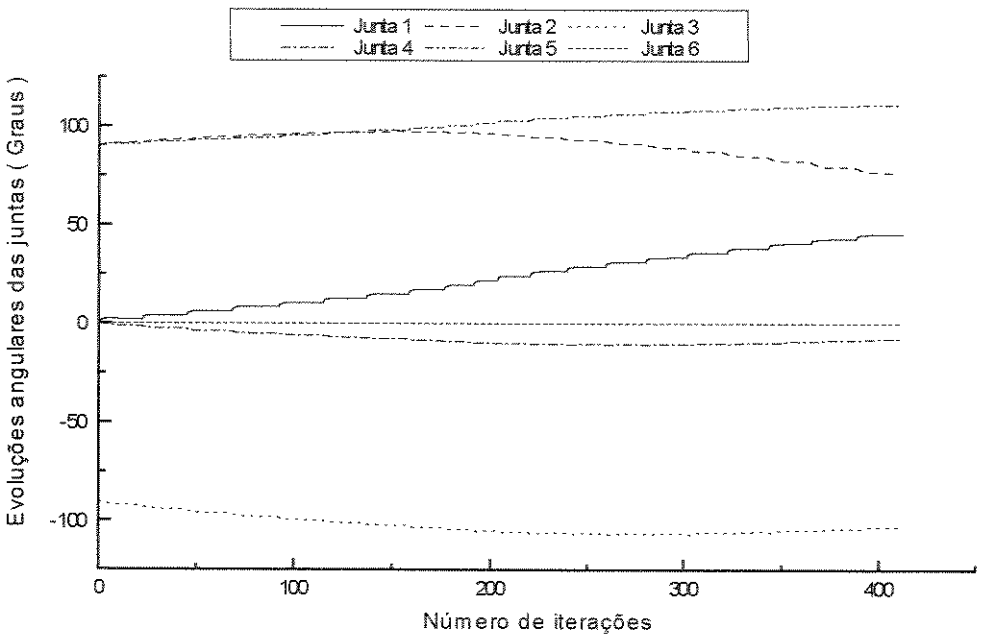


(a) Evoluções angulares das juntas.

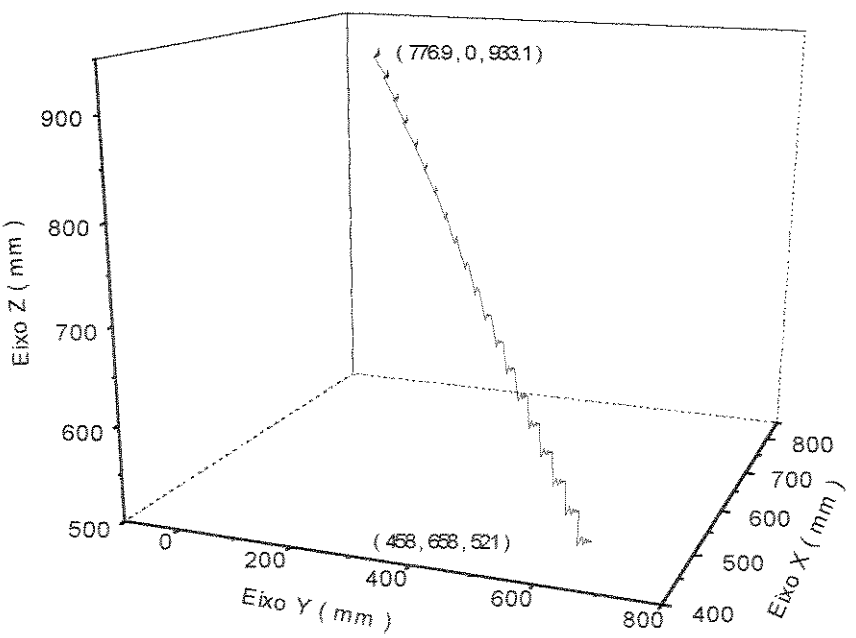


(b) Evolução epacial da garra.

Figura 5.5 : Configuração 6 (458,658,521,52,14,62) utilizando o primeiro método de Miss para *m* igual a 20 divisões.

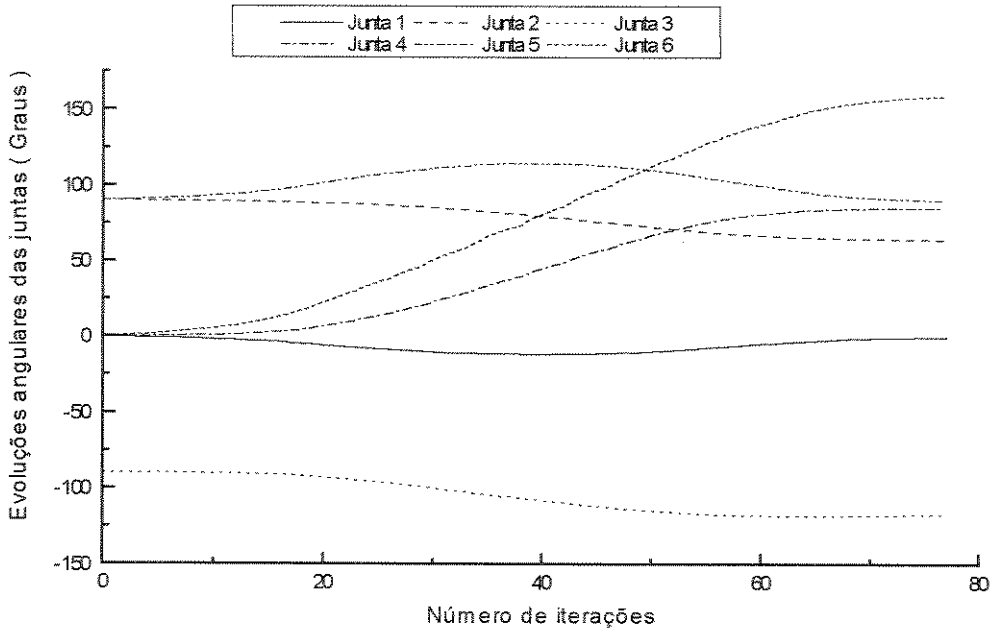


(a) Evoluções angulares das juntas.

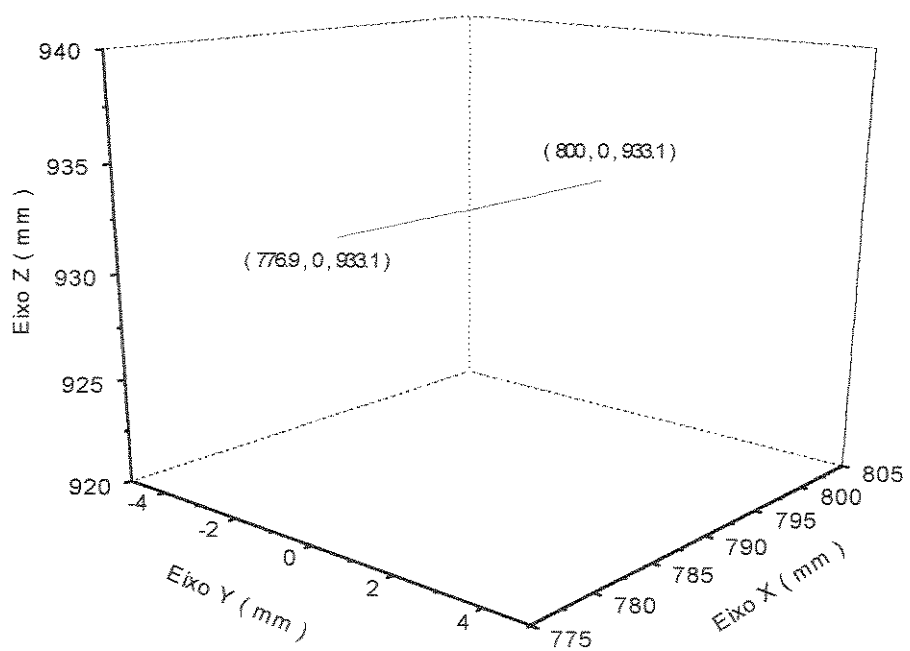


(b) Evolução espacial da garra.

Figura 5.6 : Configuração 6 (458,658,521,52,14,62) utilizando o segundo método de Miss para *m* igual a 20 divisões.

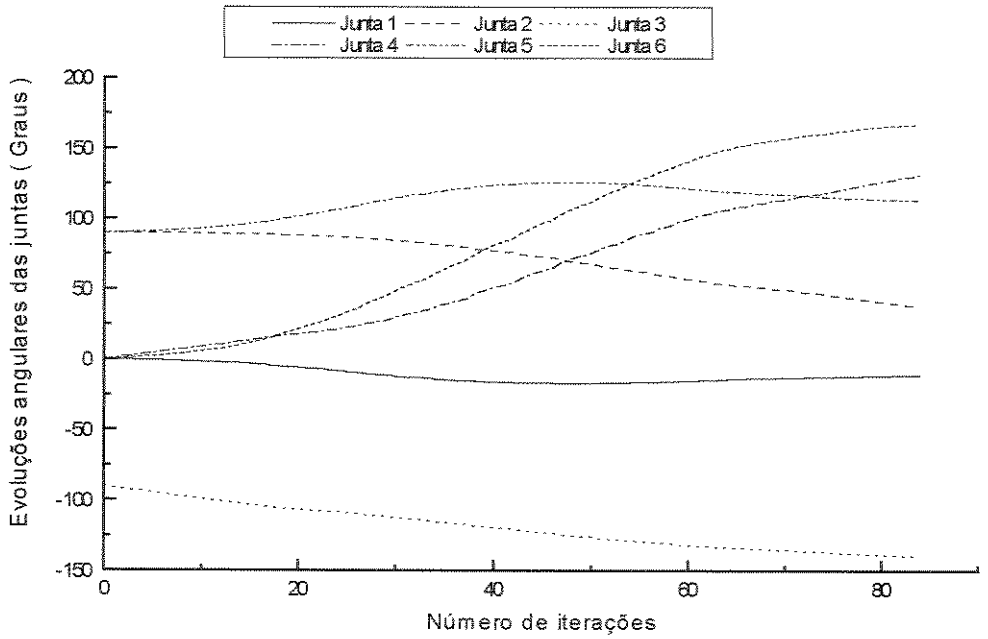


(a) Evoluções angulares das juntas.

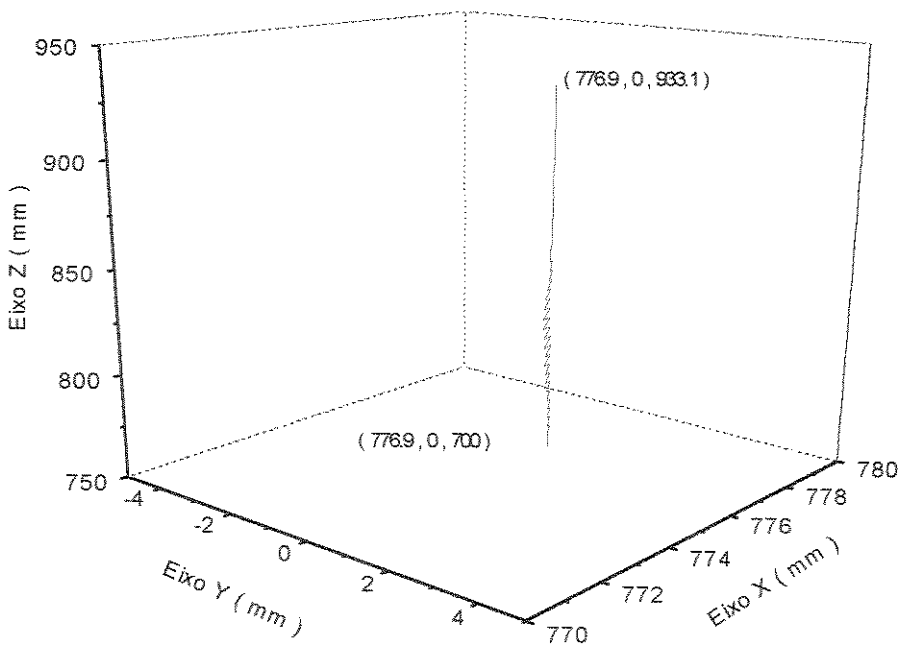


(b) Evolução espacial da garra.

Figura 5.7 : Configuração 1 (800,0,933.1,21,58,90) utizando os métodos de Gauss e Greville para *m* 50 divisões.

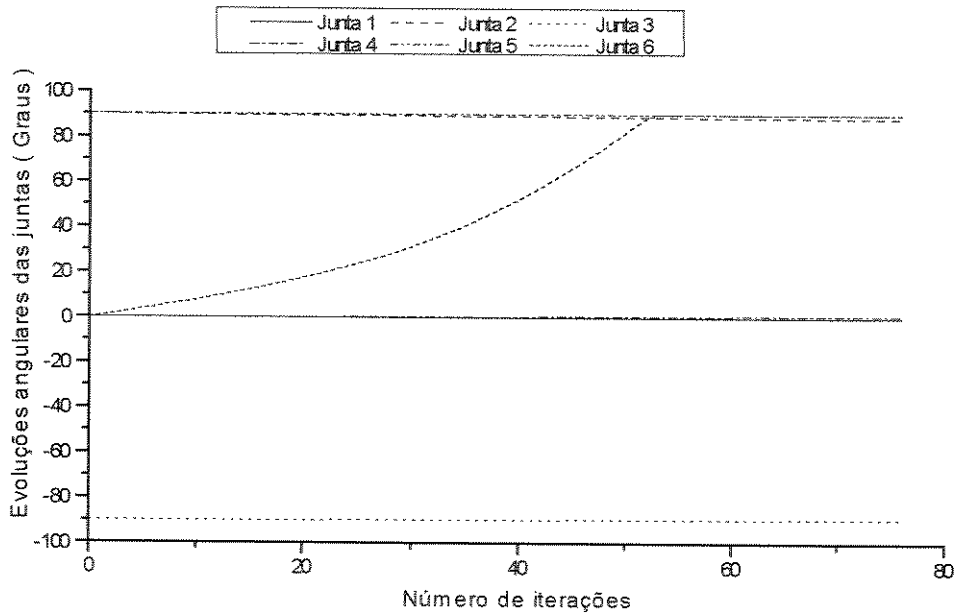


(a) Evoluções angulares das juntas.

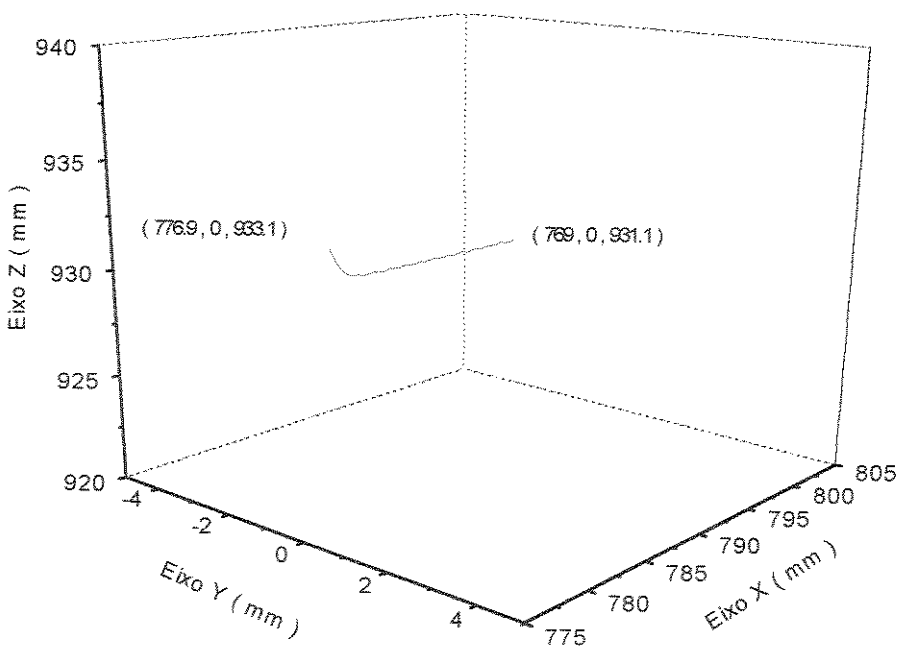


(b) Evolução espacial da garra.

Figura 5.8 : Configuração 2 (776.9,0,700,25,63,75) utilizando os métodos de Gauss e Greville para *m* igual a 50 divisões.

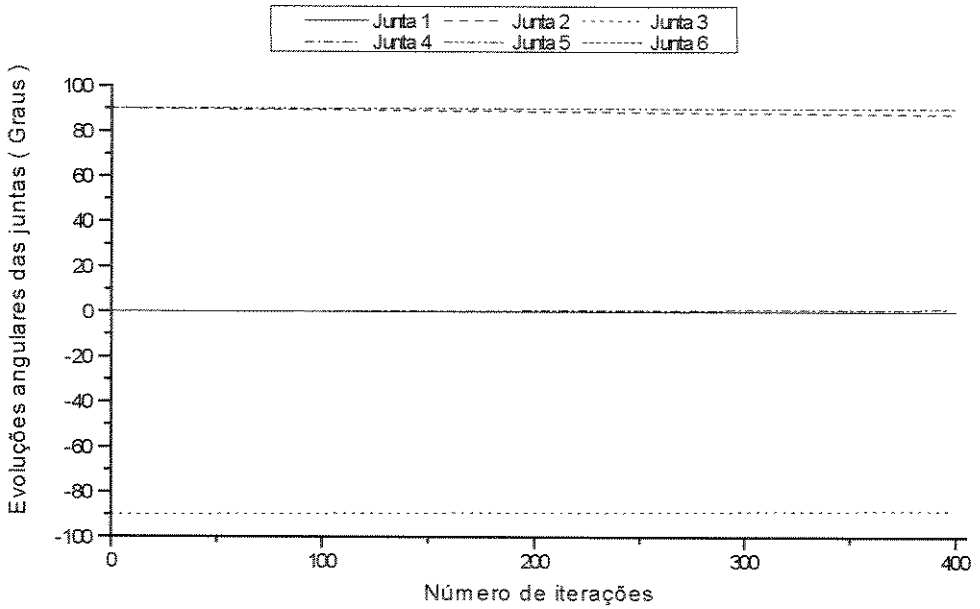


(a) Evoluções angulares das juntas.

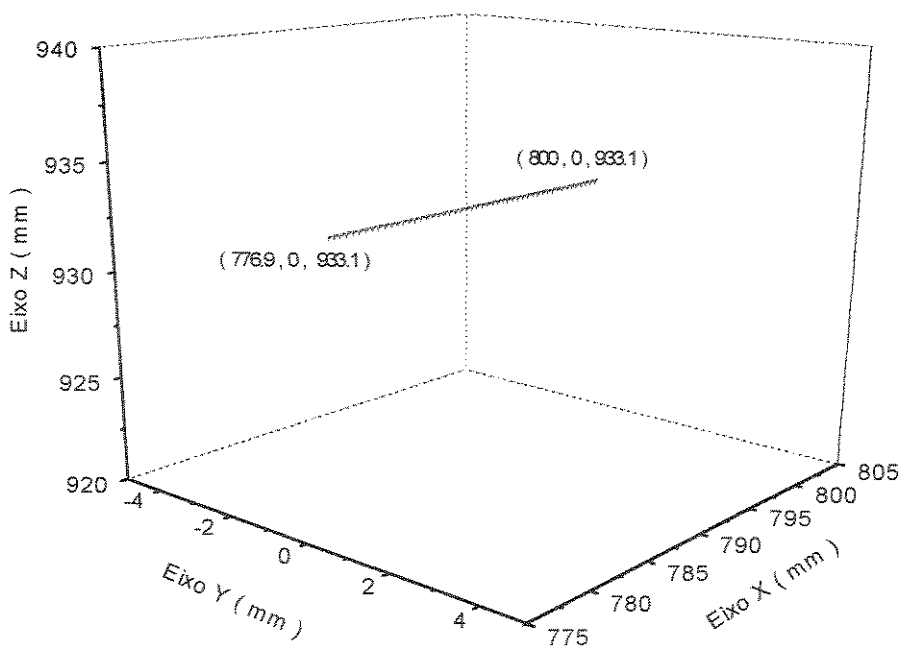


(b) Evolução espacial da garra.

Figura 5.9 : Configuração 1 (800,0,933.1,21,58,90) utilizando o primeiro método de Miss para *m* igual a 50 divisões.

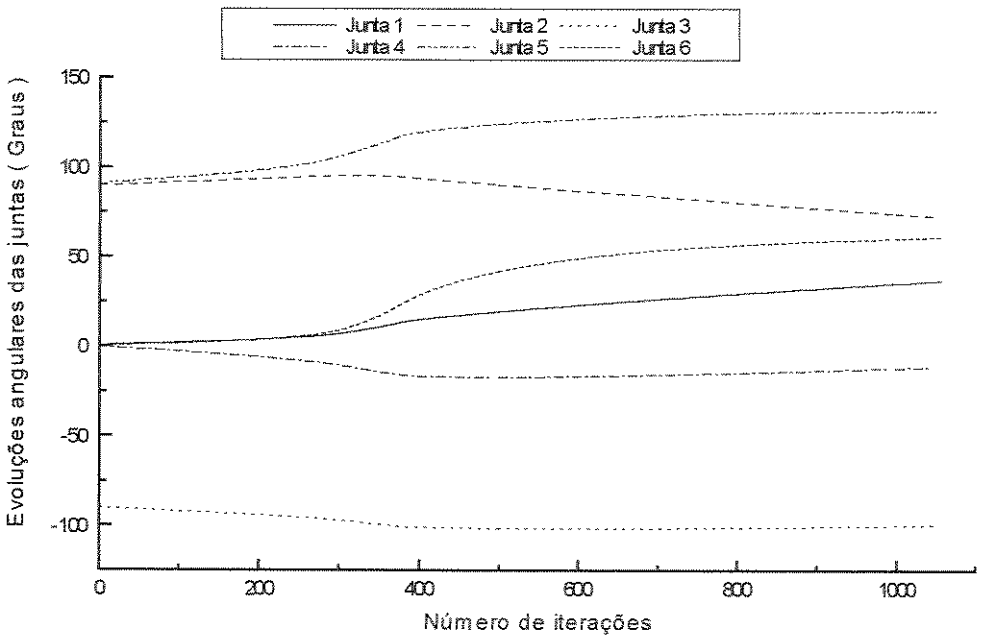


(a) Evoluções angulares das juntas.

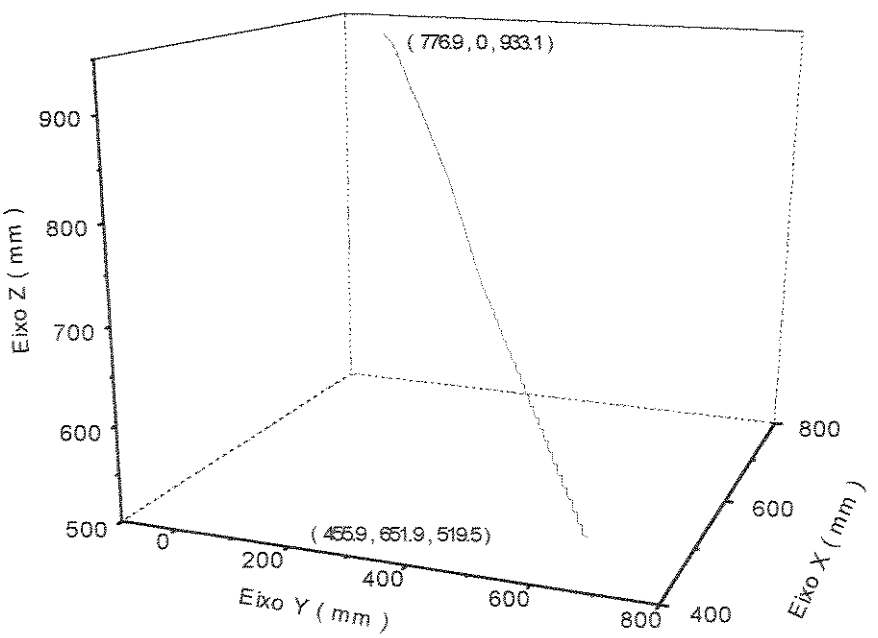


(b) Evolução espacial da garra.

Figura 5.10 : Configuração 1 (800,0,933.1,21,58,90) utilizando o segundo método de Miss para *m* igual a 50 divisões.

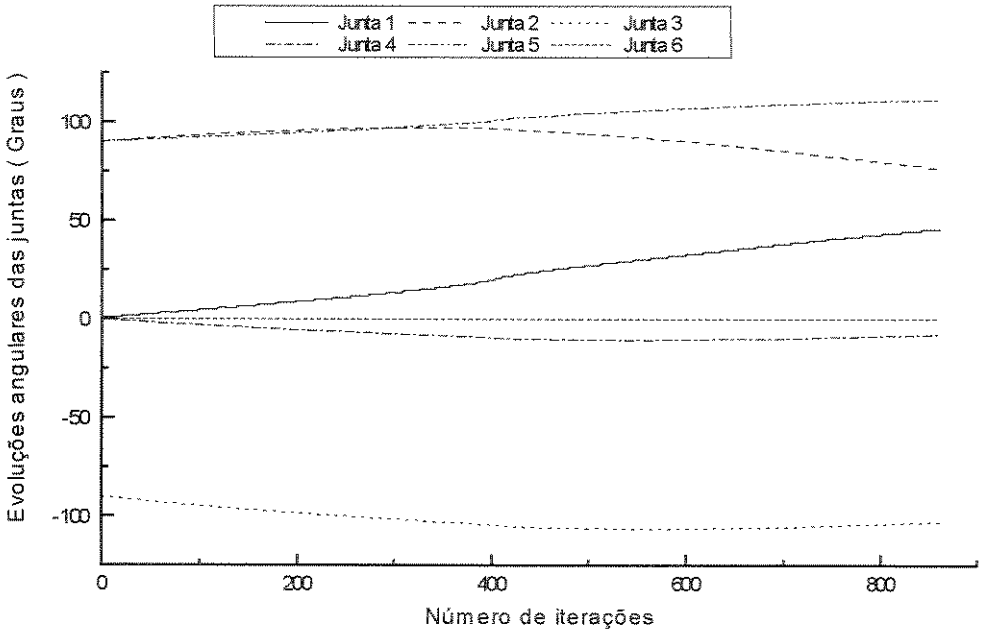


(a) Evoluções angulares das juntas.

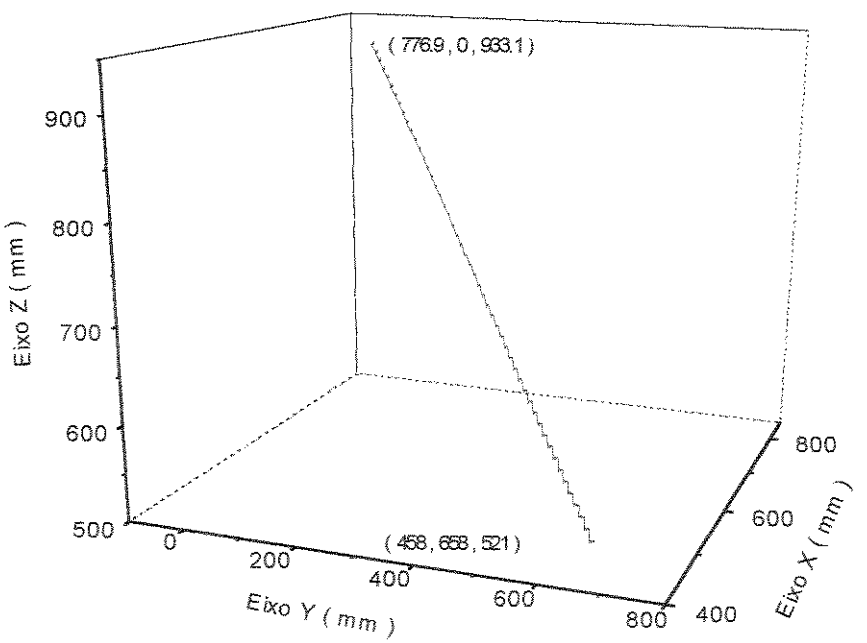


(b) Evolução espacial da garra.

Figura 5.11 : Configuração 6 (458,658,521,52,14,62) utilizando o primeiro método de Miss para *m* igaul a 50 divisões.



(a) Evoluções angulares das juntas.



(b) Evolução espacial da garra.

Figura 5.12 : Configuração 6 (458,658,521,52,14,62) utilizando o segundo método de Miss para *m* igual a 50 divisões.

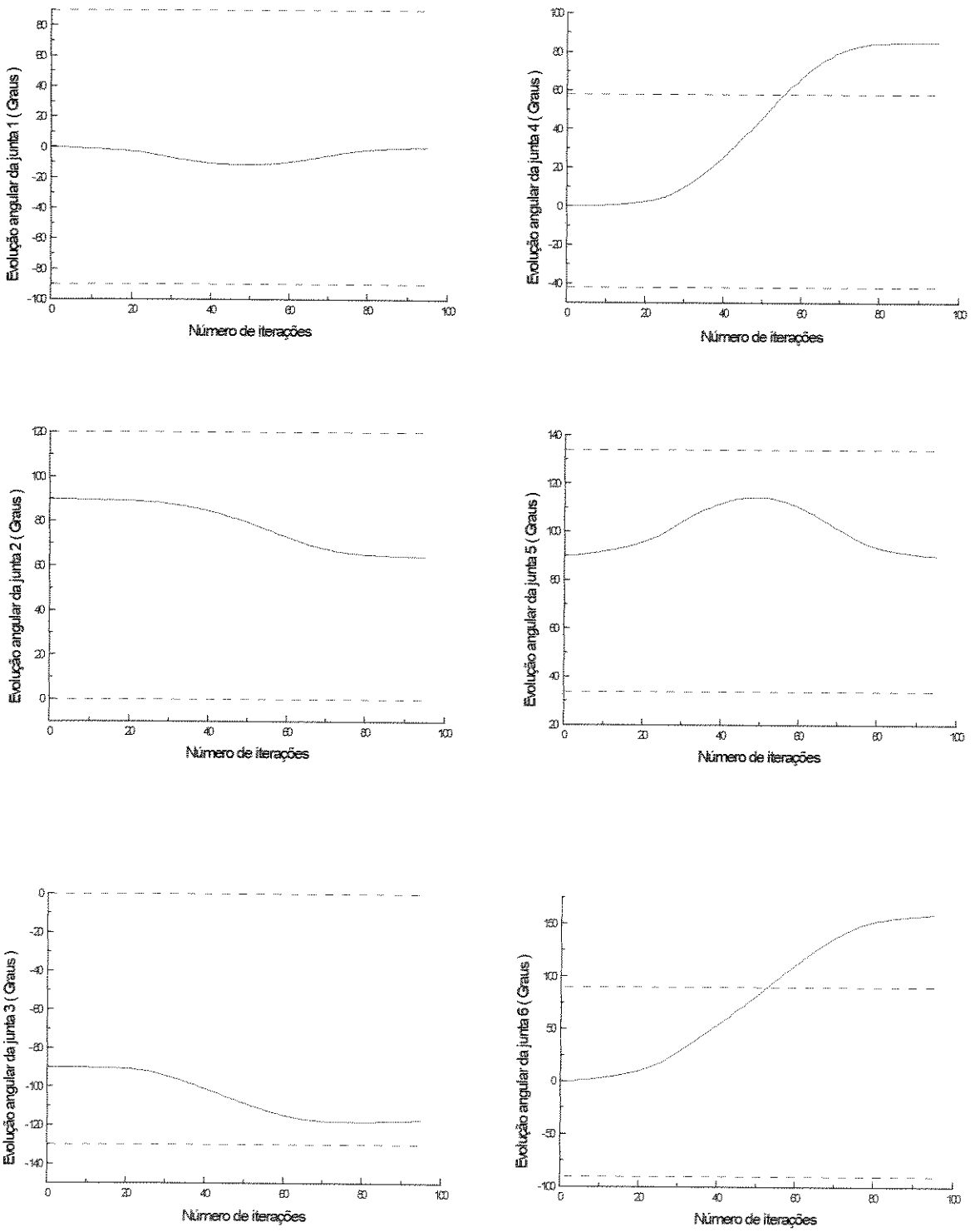


Figura 5.13 : Evoluções angulares das juntas para a configuração 1 (800,0,933.1,21,58,90) utilizando os métodos de Gauss e Greville para *m* igual a 70 divisões.

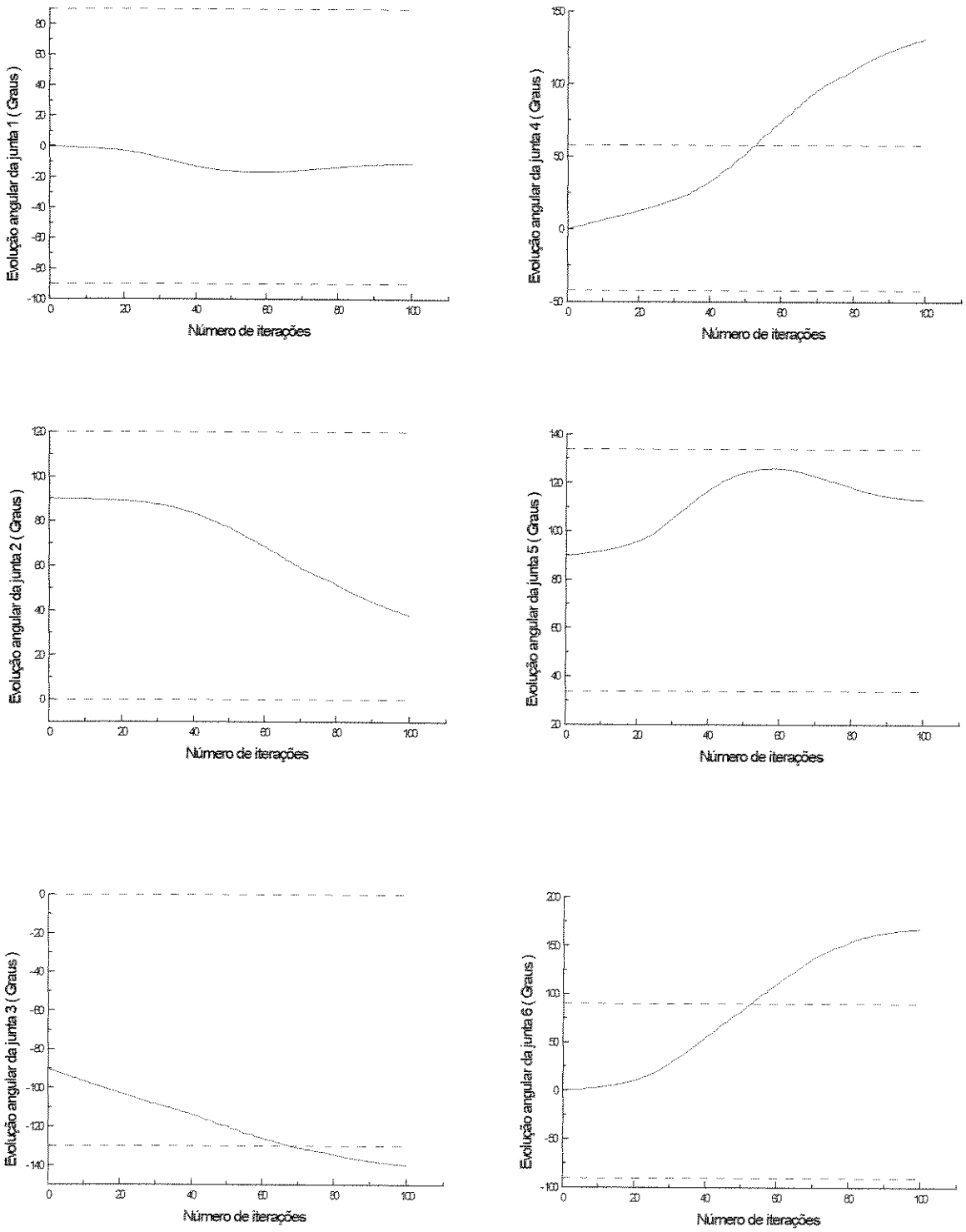


Figura 5.14 : Evoluções angulares das juntas para a configuração 2 (776.9,0,700,25,63,75) utilizando os métodos de Gauss e Greville para *m* igual a 70 divisões.

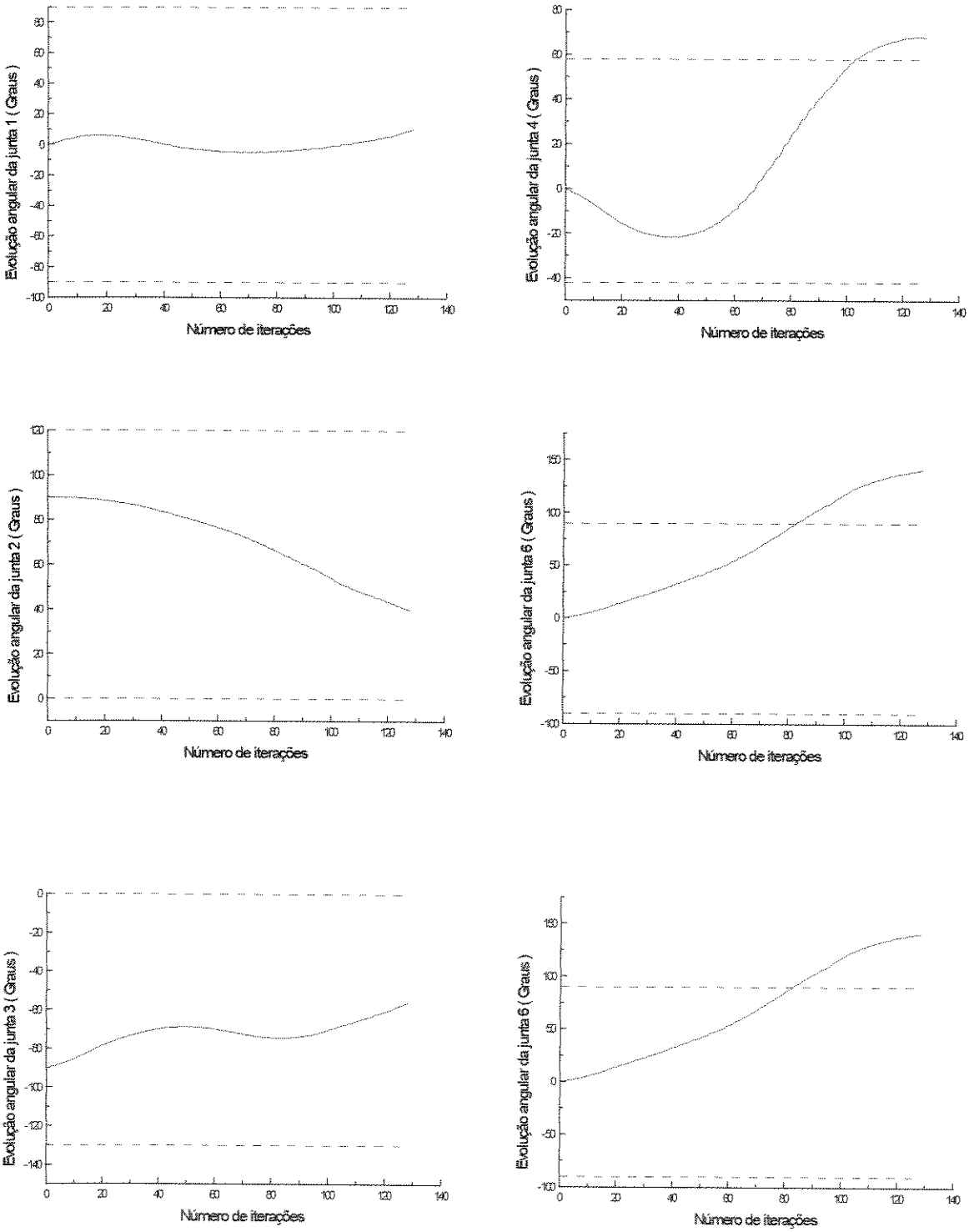


Figura 5.15 :Evoluções angulares das juntas para a configuração 3 (776.9,456,933.1,85,62,14) utilizando os métodos de Gauss e Greville para m igual a 70 divisões.

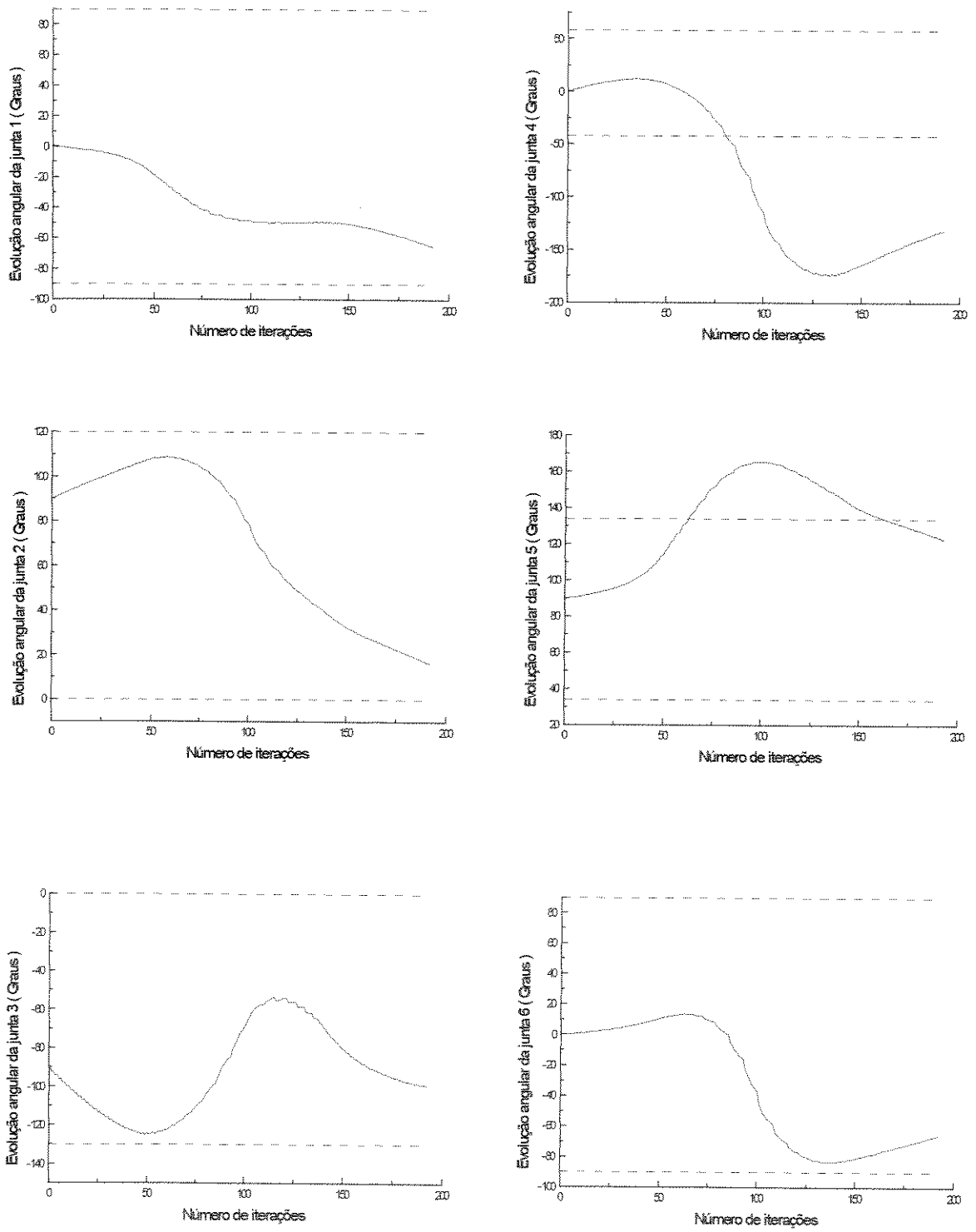


Figura 5.16 : Evoluções angulares das juntas para a configuração 4 (250,-45,450,45,62,14) utilizando osmétodos de Gauss e Greville para *m* igual a 70 divisões.

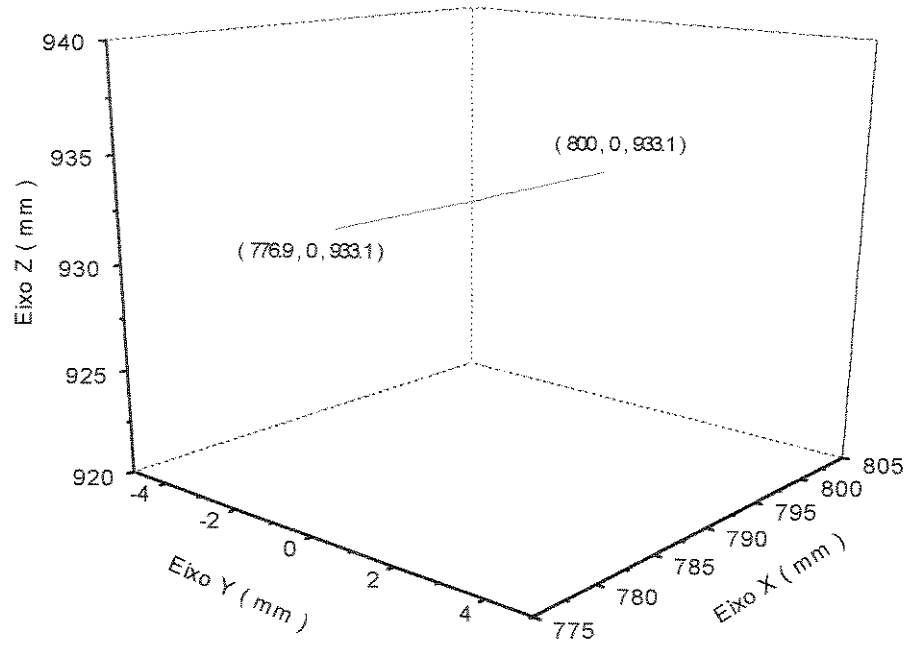


Figura 5.17 : Evolução espacial da garra para a configuração 1 (800,0,933.1,21,58,90) utilizando os métodos de Gauss e Grevillepara *m* igual a 70 divisões.

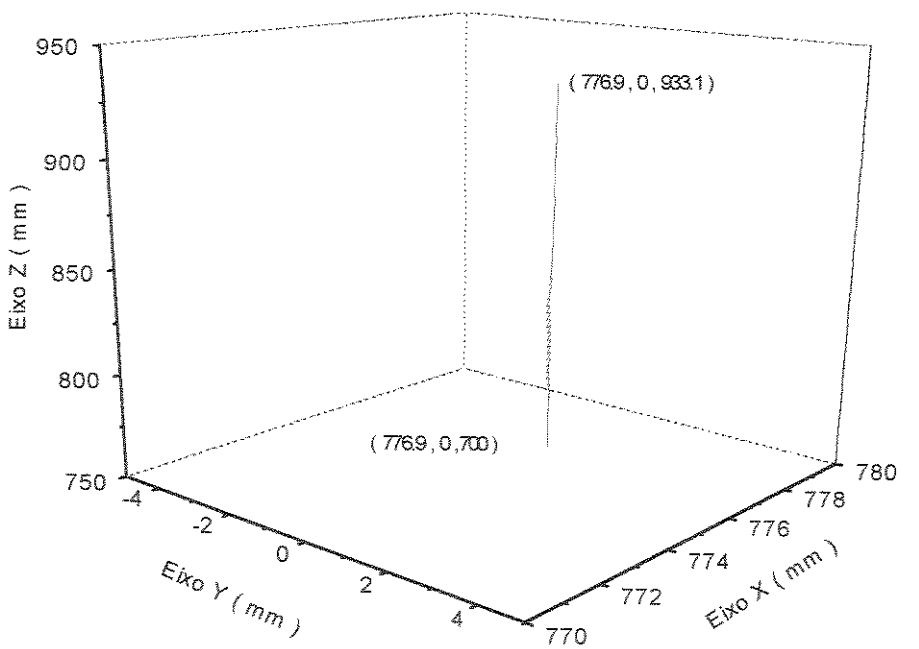


Figura 5.18 : Evolução espacial da garra para a configuração 2 (776.9,0,700,25,63,75) utilizando os métodos de Gauss e Greville para *m* igual a 70 divisões.

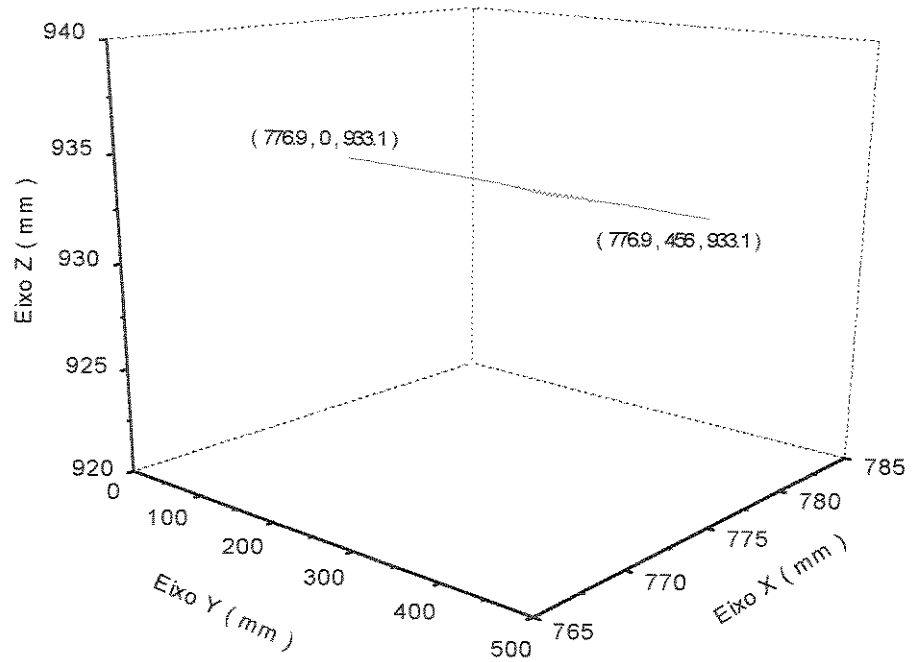


Figura 5.19 : Evolução espacial da garra para a configuração 3 (776.9,456,933.1,85,62,14) utilizando os métodos de Gauss e Greville para *m* igual a 70 divisões.

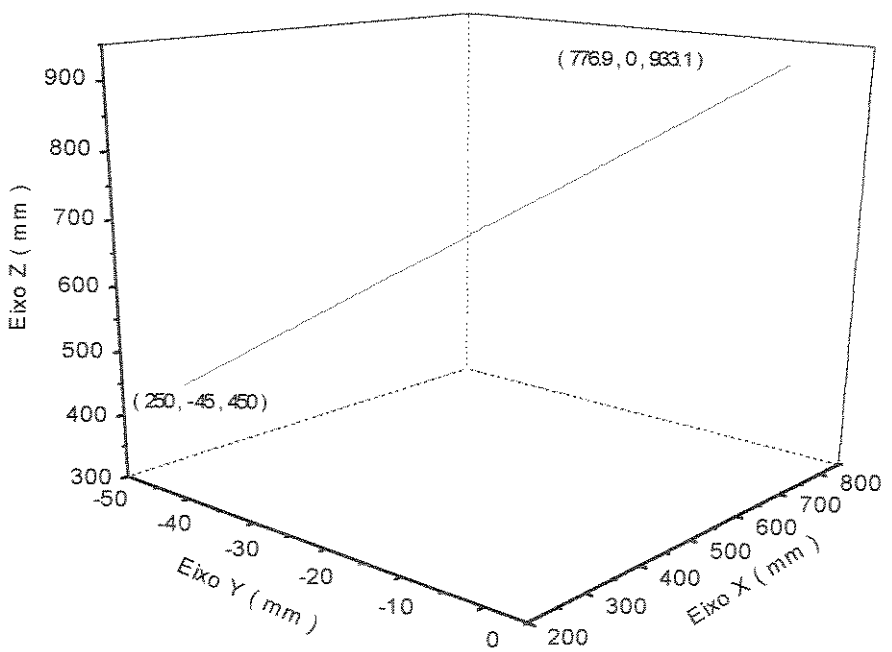
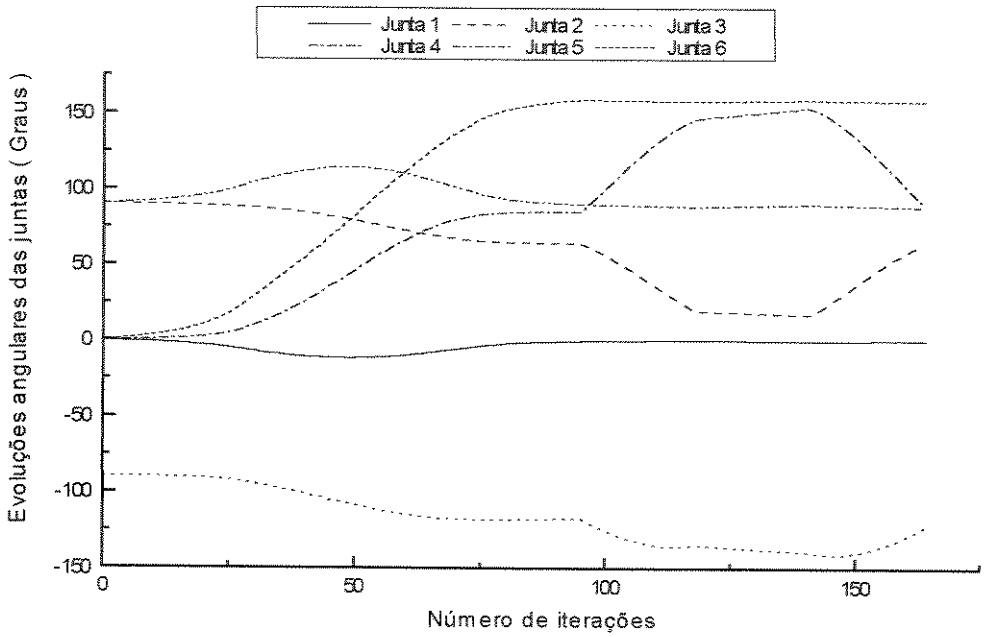
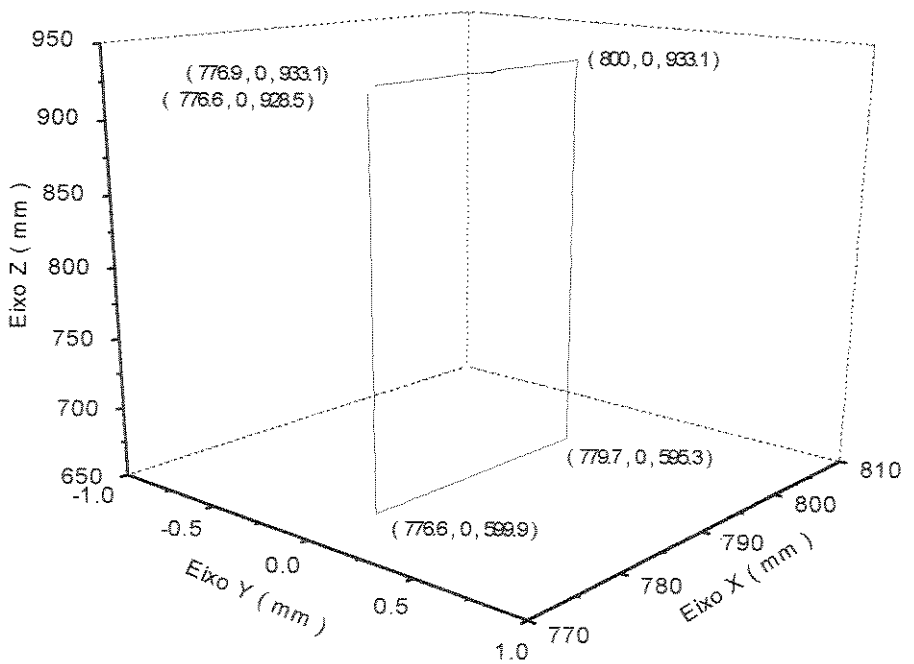


Figura 5.20 : Evolução espacial da garra para a configuração 4 (250,-45,450,45,62,14) utilizando os métodos de Gauss e Greville para *m* igual a 70 divisões.



(a) Evoluções angulares das juntas.



(b) Evolução espacial da garra.

Figura 5.21 : Configurações 5, 6, 7 e 8 utilizando os métodos de Gauss e Greville para *m* igual a 70 divisões.

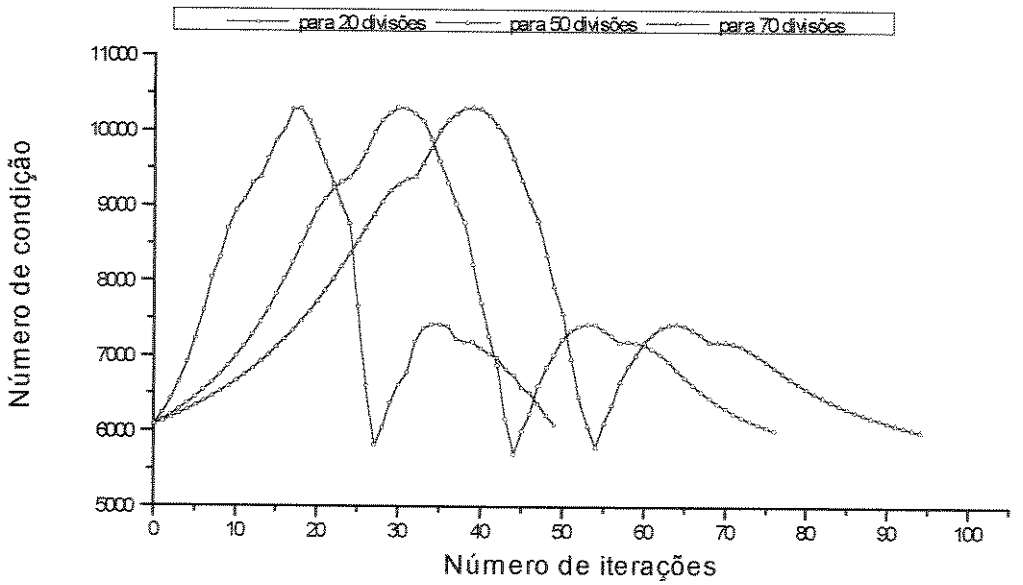


Figura 5.22 : Comportamento numérico do número de condição para a configuração 1 (800,0,933.1,21,58,90) utilizando os métodos de Gauss e Greville.

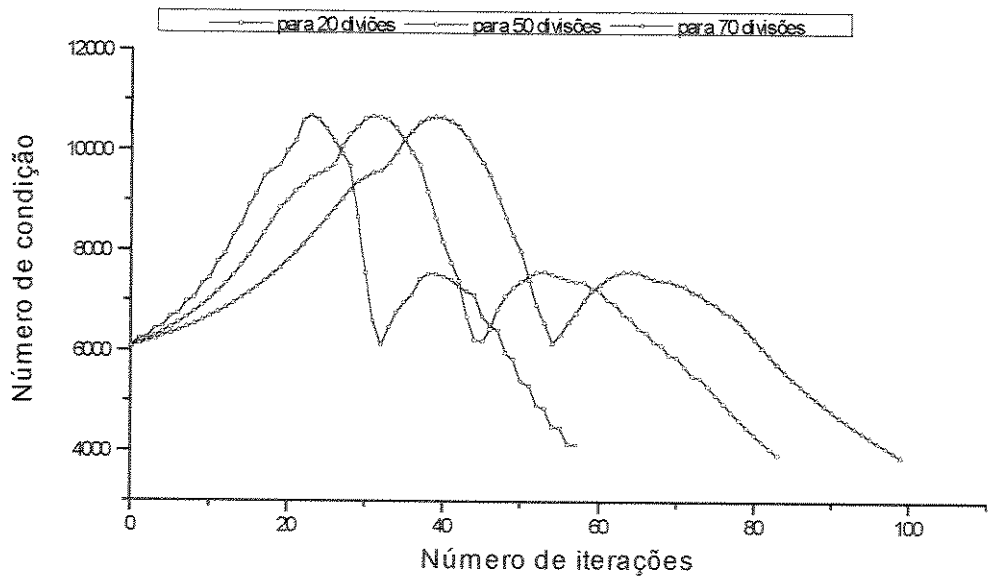


Figura 5.23 : Comportamento numérico do número de condição para a configuração 2 (776.9,0,700,25,63,75) utilizando os métodos de Gauss e Greville.

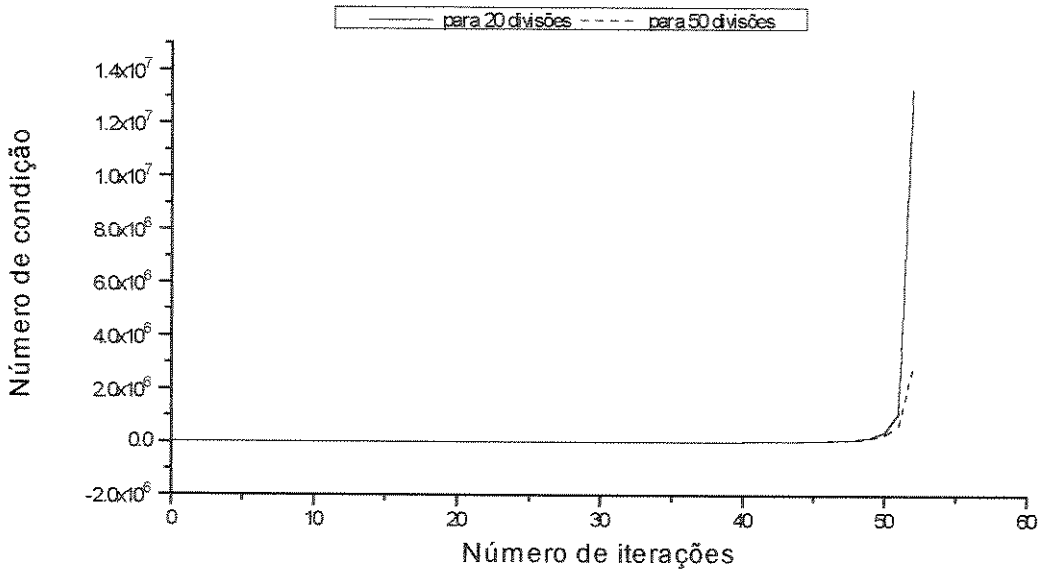


Figura 5.24 : Comportamento numérico do número de condição para a configuração 1 (800,0,933.1,21,58,90) utilizando o primeiro método de Miss.

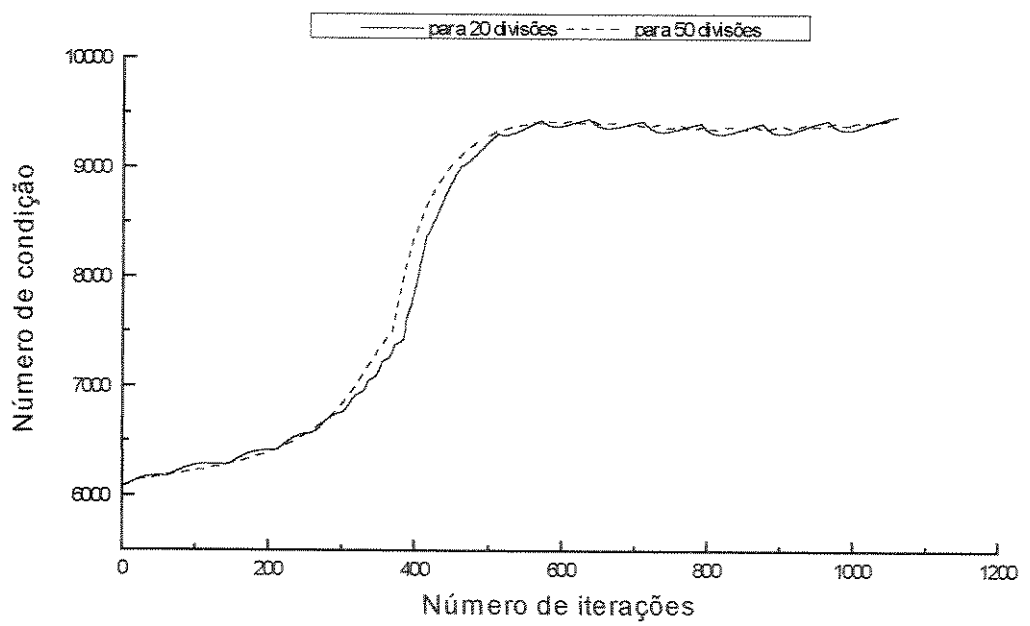


Figura 5.25 : Comportamento numérico do número de condição para a configuração 6 (458,658,521,52,14,62) utilizando o primeiro método de Miss.

5.3 - Interpretação dos resultados obtidos

5.3.1 - Resumo dos resultados

As tabelas que serão apresentadas a seguir contém o número de iterações (N) e as juntas que ultrapassaram os limites fisicos (JFL) após o termino da convergência para cada configuração e valor de *m* utilizado.

Método	Gauss				Greville			
m	***20***	***50***	***70***		***20***	***50***	***70***	
Config.	N	N	N	JFL	N	N	N	JFL
1	50	77	95	4-6	50	77	95	4-6
2	58	84	100	3-4-6	58	84	100	3-4-6
3	64	113	128	4-5-6	64	113	128	4-5-6
4	82	151	193	4	82	151	193	4
5			164	4-6			164	4-6
6				3-4-6				3-4-6
7				3-4-6				3-4-6
8				4-6				4-6

Tabela 5.2 : Número de iterações e as juntas fora dos limites (para cada valor de *m* utilizado) após o término da convergência utilizando os métodos de Gauss e Greville.

Método	Miss (primeiro método)			Miss (segundo método)		
m	***20***	***50***		***20***	***50***	
Config.	N	N	JFL	N	N	JFL
1	76	76	-	211	400	-
2	120	118	-	229	455	-
3	966	960	-	544	1119	-
4	nc	nc	nc	nc	nc	nc
9	1062	1055	-	412	862	-

Tabela 5.3 : Número de iterações e as juntas fora dos limites (para cada valor de *m* utilizado) após o término da convergência utilizando os métodos de Miss.

O tempo aproximado para cada iteração realizada, foi baseado utilizando-se um computador 486 DX2-50Mhz e são apresentados na tabela 5.4.

Método	Tempo (ms)
Gauss	9.5
Greville	8.7
Miss (primeiro método)	6.2
Miss (segundo método)	6.3

Tabela 5.4 : Tempo aproximado para cada iteração realizada.

Os tempos para os métodos de Greville e Miss foram calculados sem a realização do calculo do "Rank" da matriz necessário apenas para o método de Gauss.

5.3.2 - Análise dos resultados

Baseado nos algorítmos utilizados e resultados obtidos, será a seguir discutido a influência dos métodos nos principais parâmetros:

- Influência da Variável *m*

Pode observar que com o aumento do valor de *m* as oscilações nos gráficos das posições angulares (figuras 5.1a, 5.12a, 5.2a, 5.3a, 5.4a, 5.5a, 5.6a, 5.7a, 5.8a, 5.9a, 5.10a, 5.11a, 5.12a, 5.13, 5.14, 5,15 e 5.16) e nos gráficos da evolução espacial da garra (figuras 5.1b, 5.2b, 5.3b, 5.4b, 5.5b, 5.6b, 5.7b, 5.8b, 5.9b, 5.10b, 5.11b, 5.12b, 5.17, 5.18, 5.19 e 5.20) diminuiram. Isto se deve ao fato de que com o aumento do valor de *m* o valor do erro $d\underline{x}$ diminuiu e consequentemente os incremento das juntas, $d\underline{\theta}$, são pequenos.

Observa-se que com o aumento de *m* o número de iterações aumenta, tabelas 5.2 e 5.3 apresentadas no subtópico 5.3.1, excetuando-se para o primeiro método de Miss, para o qual, o número de iterações permanece aproximadamente o mesmo.

Pode-se observar que existe um valor ótimo de m para o qual as oscilações não apresentam mudanças acarretando apenas um aumento de iterações. Isto pode ser vizualizado através das figuras 5.1b, 5.7b e 5.17.

- Influência do número de condição

O valor do numero de condição da matriz Jacobiana para cada iteração não diminuiu com o aumento da variável m, como pode ser observado nas figuras 5.22, 5.23, 5.26 e 5.27, apenas se deslocaram.

Um detalhe importante que pode ser observado nas figuras 5.3b e 5.9b, gráficos obtidos utilizando o primeiro método de Miss, é que a trajetória da garra não segue uma linha reta desejada, devido ao alto valor do número de condição no final das iterações como pode ser observado na figura 5.24.

A utilização do algorítmo para o refinamento da solução de sistemas mal-condicionados, figura 3.3, não se mostrou necessário.

- Erros de orientação e posição

Os erros de posição e orientação obtidos, erros entre a posição e orientação final desejada e inicial após o termino das iterações, tenderam a zero para os métodos de Gauss e Greville e apresentaram um pequeno erro de orientação para os métodos de Miss. Os erros obtidos são apresentados no Anexo E, tabelas e.1 e e.2.

- Estudo das redundâncias

As posições angulares finais para os métodos de Gauss e Greville são as mesmas mas diferem das obtidas para os métodos de Miss que são praticamente iguais. As posições angulares finais são apresentadas no Anexo E, tabelas e.3 e e.4. Isto ocorre devido a existencia de posições singulares do sistema.

- Influência sobre os limites físicos das juntas

Pode-se observar que os gráficos das trajetórias angulares, obtidos utilizando-se os métodos de Gauss e Greville, excedem os limites físicos para determinadas juntas. Mas isto não impede a utilização destes métodos pois, em um robô real quando uma das juntas está para sair fora de seu limite ele automaticamente para tendo, deste modo, uma parte da trajetória gravada. Para se obter o restante da trajétoria basta rearumar as juntas do manipulador, mantendo a sua posição e orientação, e gerar o restante da trajétoria a partir do ponto anterior até a final desejada.

Isto não ocorre para os métodos de Miss. Em todas as simulações realizadas as trajetórias angulares permaneceram dentro dos limites físicos permitidos.

5.4 - Comparação entre os métodos

- Gauss e Greville

Como os resultados obtidos para os métodos de Gauss e Greville foram os mesmos a priori, não existiria diferenças, mas o número de operações matemáticas necessárias para o calculo da matriz pseudoinversa pelo método de Greville é menor, deste modo, demandando menor tempo, tabela 5.4. Além disso, para o método de Greville, não há problemas caso a matriz seja não-singular.

- Miss método 1 e método 2

Como anteriormente mencionado, para o primeiro método de Miss, o número de iterações não aumentou com o aumento de m mas, pode-se observar, através das figuras 5.5 e 5.11, que as oscilações, tanto da evolução angular quanto da evolução espacial da garra diminuiram. Para o segundo método as oscilações diminuiram mas com o aumento do número de iterações.

O tempo necessário para cada iteração é menor para o primeiro método de Miss, tabela 5.4.

5.5 - Conclusões gerais

Notou-se a necessidade da variável interna *m* pois, através do aumento da mesma as oscilações nas trajetórias angulares e, consequentemente, as oscilações nas evoluções espaciais da garra diminuiram ou até mesmo deixaram de existir.

Ao mesmo tempo o algoritmo de geração de trajetórias permite que se realize uma trajetória a partir do final de uma já realizada. Isto pode ser observado através da figura 5.21 onde é realizada uma trajetória retangular.

Não foram apresentados neste capítulo todos os gráficos, apenas alguns, pois, de maneira geral os resultados obtidos foram semelhantes.

CAPÍTULO 6

Conclusões e Perspectivas

Este trabalho teve como objetivo o desenvolvimento de um software de geração de trajetórias off-line para o manipulador submarino Kraft com vistas a sua utilização em tarefas de automação submarinas. As técnicas implementadas neste trabalho poderão ser extendidos facilmente para outros tipos de robôs industrias.

O software desenvolvido cumpre os objetivos inicialmente planejados de gerar trajetórias off-line podendo ser aplicado em tempo real através de pequenas melhorias nas funções já implementadas.

Os métodos utilizados para a inversão da matriz jacobiana apresentaram ótimos resultados o que faz destes métodos um grande atrativo para a solução do problema cinemático inverso.

Estes métodos podem ser utilizados em um procedimento para identificação e calibração de parâmetros geométricos para robôs (Campos, 1993).

A partir dos pacotes já desenvolvidos pretende-se introduzir novos módulos complementares, tais como módulos de detecção de colisões entre outros, que poderão ser implementados em tempo real, os quais terão como principal característica a fácil utilização e/ou modificação pelo usuário.

Além disso, poderá ser realizado um estudo da acurácia, repetibilidade e estabilidade do manipulador, ao descrever a trajetória automaticamente, e implementação das eventuais modificações (nos sensores do manipulador e/ou no software desenvolvido) decorrentes deste estudo. Deste modo será possível obter a efetiva utilização do manipulador para realizar tarefas automatizadas em ambientes submarinos.

Neste trabalho utilizou-se um robô com juntas rotacionais mas isto não impede que a metodologia desenvolvida seja aplicada a robôs com juntas diferentes.

Referências Bibliográficas

- **Atkinson**, K. [1985]. "Elementary Numerical Analysis".John Wiley & Sons.

- **Atkinson**, K. [1988]. "An Introduction to Numerical Analysis". John Wiley & Sons.

- **Campos**, R. , Rosário, J. M. [1993]. "Calibração e Identificação de Parâmetros" . XII Congresso Brasileiro de Engenharia Mecânica - Brasília - Brasil, Dezembro.

- **Dorn**, W. S. , McCracken, D. D. [1972]. "Numerical Methods with Fortran IV Cases Studies". John Wiley & Sons, Inc.

- **Fu**, K. S., et al [1987]. "Robotics : Control, Sensing, Vision and Intelligence". McGraw - Hill,Inc.

- **Gerald**, C. F. , Wheatley, P. O. [1992]. "Applied Numerical Analysis" . Addison-Wesley Publishing Company.

- **Gorla**, B. , Renaud, M. [1984]. "Modéles des Robots Manipulateurs Application à Leur Commande" . Cepadues-Éditions.

- **Graig**, J. J. [1986]. "Introduction to Robotics Mechanics & Control" . Addison - Wesley Publishing Company, Inc.

- **Hayati**, S. A. , Roston, G.P. [1986]. "Inverse Kinematic Solution for Near-Simple Robots and its Application to Robot Calibration". Recente Trends in Robotics Modeling, Control and Education, Elsevier Science Publishing Co., Inc, p 41-50.

- **Huang**, C. H. , Klein, C. A. [1983]. "Review of Pseudoinverse Control for Use With Kinematically Redundant Manipulators" . IEEE Transaction on Systems, Man and Cybernetics, vol. smc-13, no2, March/April,p 245-250.

- **Koivo**, A. J. [1989]. "Fundamentals for Control of Robotic Manipulators" . John Wiley & Sons, Inc.

- **Kreysizig**, E. [1983]. "Advanced Engineering Mathematics" . John Wiley & Sons, Inc.

- **Luh**, J. Y. S., Walker, M. W., Paul, R. P. C. [1980]. "Resolved-Acceleration Control of Mechanical Manipulators". IEEE Transactions on Automatic Control, vol. ac-25, no. 3, June.

- **Miss**, R. W. , Schutheis, G. F. [1991]. "The Desing of an Algorithm for the Control of Reduntant Kinematics Chains". 5th ICAR, Pisa, Italy, pp. 1783,1991.

- **Nashed**, M. Z. [1976]"Generalized Inverses and Aplications". Academic Press, Inc.

- **Paul**, R. P. , Shimano, B. , Mayer, G. E. [1981]. "Differential Kinematics Control Equations for Simple Manipulators" . IEEE Transaction on Systems, Man and Cybernetics, vol. smc-11, no 6, June,p 456-460.

- **Paul**, R. P. [1981]. "Robot Manipulators : Mathematics, Programming and Control" . The Mit Press.

- **Paul**. R. [1979]. "Manipulator Cartesian Path Control". IEEE Transactions on Systems, Man and Cybernetics, vol. smc-9, no. 11, November.

- **Campos**, R. [1993]. "Implementação de um mecanismo de calibração e identificação de parâmetros para robôs". Tese de Mestrado, Unicamp.

- **Spong**, M. W. [1989]. "Robot Dynamics and Control" . John Wiley & Sons, Inc.

- **Vandergraft**, J. S. [1983]. "Introduction to Numerical Computations". Academic Press, Inc.

- **Watt**, D. A. [1987]. "Ada : Laguage and Methodology". Prentice - Hall International (UK) Ltd.

- **Withney**, D. E. [1969]. "Resolved Motion Rate Control of Manipulators and Human Protheses". IEEE Transactions on Man-Machine Systems, vol MMS-10, No 2, pp 47 - 53, June.

- **Wu**, C.-H., Paul, R. P. [1982]. "Resolved Motion Force Control of Robot Manipulator". IEEE Transactions on Systems, Man and Cybernetics, vol. smc-12, no. 3, May/June.

ANEXO A

Método de Denavit-Hartenberg

Visto que a metodologia utilizada neste trabalho faz uso dos parâmetros de Denavit-Hartenberg para determinar as matrizes de passagem homogêneas, apresentamos neste anexo um breve comentário de como obter esses parâmetros e as equações cinemáticas de um robô.

Para obter a equação que fornece o posicionamento do elemento terminal de um robô em relação ao sistema de coordenadas da base, Denavit-Hartenberg propuseram a utilização de quatro parâmetros, θ, α, a e d para descrever uma matriz de transformação homogênea A_i^{i-1} entre dois sistemas de coordenadas vinculados a duas juntas sucessivas.

A figura a.1 apresenta os quatros parâmetros de Denavit-Hartenberg. Onde:

a_i é a menor distância entre z_i e z_{i-1}

α_i é o ângulo entre z_i e z_{i-1}

d_i é a menor distância entre x_i e x_{i-1}

θ_i é o ângulo entre x_i e x_{i-1}

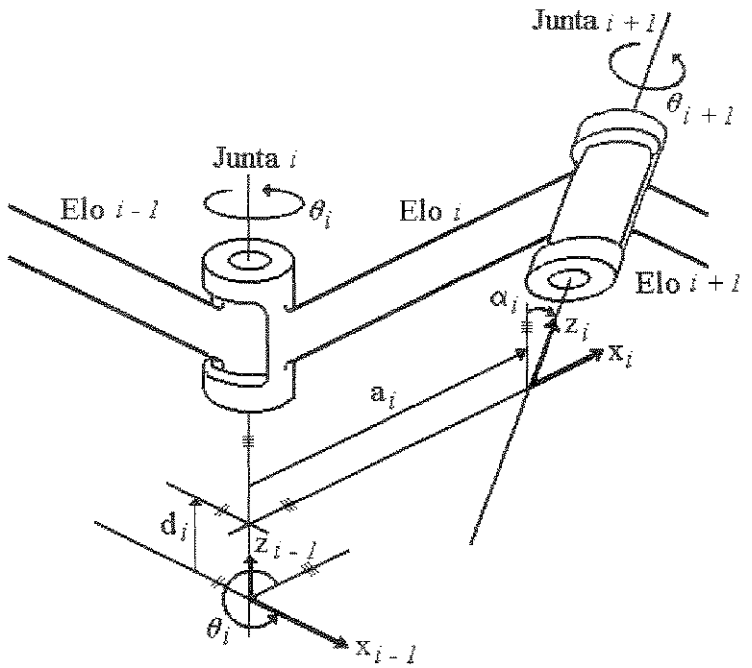


Figura a.1: Parâmetros de Denavit-Hartenberg, θ, α, a e d.

A matriz de passagem A_i^{i-1} é expressa como o produto de quatro transfomações homogêneas, isto é:

$$A_i^{i-1} = \text{rot}(z, \theta)\ \text{trans}(0, 0, d)\ \text{trans}(a, 0, 0)\ \text{rot}(x, \alpha)$$

então

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ -\sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 1 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (A.1)$$

Equações Cinemáticas

A matriz homogenea T_i^0 que especifica a localização do i-ésimo sistema de coordenadas em relação ao sistema de coordenadas da base é o produto das sucessivas matrizes A_i^{i-1} e é expessa como:

$$\mathbf{T}_i^0 = A_1^0 \, A_2^1 \, \ldots \, A_i^{i-1} \tag{A.2}$$

$$= \begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

onde $[\underline{n} \quad \underline{s} \quad \underline{a}]$ é a matriz orientação do i-ésimo sistema de coordenadas em relação ao sistema de coordenadas da base e $\underline{p}$ é o vetor posição do i-ésimo sistema de coordenadas em relação ao sistema de coordenadas da base. Os vetores $\underline{n}$, $\underline{s}$, $\underline{a}$ e $\underline{p}$ são mostrados na figura a.2.

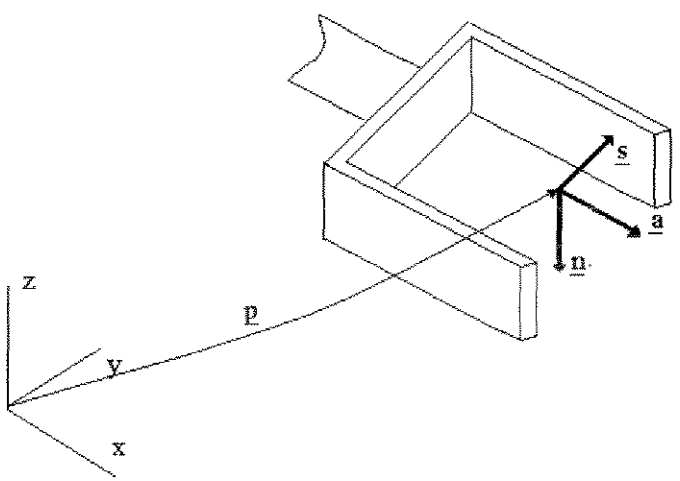


Figura a.2: Vetores $\underline{n}$, $\underline{s}$, $\underline{a}$ e $\underline{p}$.

Obtenção do Modelo Geométrico

Como anteriormente citado, existem diversas maneiras de obter a matriz de passagem homogênea do robô. Cada uma delas gera expressões diferentes que, embora equivalentes quantitativamente, podem diferir grandemente quanto ao número de operações aritméticas necessárias para fazer a avaliação numérica das mesmas. Pode-se enumerar cinco maneiras diferentes de fazê-lo, eles são:

1. Através da multiplicação das matrizes elementares completas à partir do sistema de coordenadas da base até o sistema de coordenadas do orgão terminal.

2. Igual ao método 1, mas utilizando-se matrizes de apenas três colunas, visto que as três primeiras colunas de uma matriz elementar representam vetores ortonormais e assim sendo uma delas sempre pode ser obtida como produto vetorial das demais.

3. A partir da multiplicação das matrizes elementares completas à partir do sistema anterior ao do órgão terminal até o sistema da base.

4. Igual ao método 3, mas usando-se a propriedade de ortogonalidade das três primeiras colunas da matriz de transformação homogênea.

5. Igual ao método 1, mas realizando-se também simplificações trigonométricas.

ANEXO B

Determinação da matriz Jacobiana

A matriz Jacobiana intervêm diretamente na solução númerica do modelo cinemático direto e em outras aplicações em robótica sendo deste modo de grande importância. Paul [1981] utiliza matrizes de transformação 4 x 4 para obter translações e rotações diferenciais sobre um sistema de coordenadas do qual a matriz Jacobiana pode ser derivada.

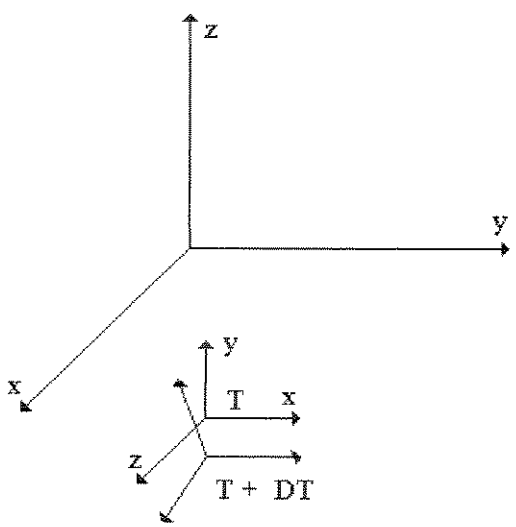


Figura b.1: Deslocamento infenitesimal T.

Dado um sistema de coordenadas T, uma mudança diferencial em T, figura b.1, corresponde a uma translação e uma rotação diferencial ao longo e sobre o sistema de coordenadas da base, isto é:

$$T + dT = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . T \qquad (B.1)$$

ou

$$dT = \left[\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right] T = \Delta T \qquad (B.2)$$

onde

$$\Delta \triangleq \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (B.3)$$

$\underline{\delta} = (\delta_x , \delta_y , \delta_z)^t$ é a rotação diferencial sobre os eixos principais do sistema de coordenadas da base e $\underline{d} = (d_x , d_y , d_z)^t$ é a translação ao longo dos eixos principais do sistema de coordenadas da base. Similarmente, a mudança diferencial em T pode ser expressa como uma translação e uma rotação diferenciais em torno do sistemas de coordenadas T:

$$T + dT = T \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (B.4)$$

ou

$$dT = T \left[\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right]$$

$$\triangleq (T)({}^{T}\Delta) \qquad \text{(B.5)}$$

onde ${}^{T}\Delta$ possui a mesma estrutura da equação (B.3), exceto que as definições de $\underline{\delta}$ e $\underline{d}$ são diferentes. $\underline{\delta} = (\delta_x , \delta_y , \delta_z)^t$ é a rotação diferencial sobre os eixos principais do sistema de coordenadas T e $\underline{d} = (d_x , d_y , d_z)^t$ é a translação ao longo dos eixos principais do sistema de coordenadas T.

A partir das equações (B.2) e (B.5), obtêm-se a relação entre Δ e ${}^{T}\Delta$:

$$\Delta T = (T) ({}^{T}\Delta)$$

ou

$${}^{T}\Delta = T^{-1} \Delta T \qquad \text{(B.6)}$$

Fu [1987] define a inversa de uma matriz de transformação homogênea como sendo:

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\underline{n}^T \underline{p} \\ s_x & s_y & s_z & -\underline{s}^T \underline{p} \\ a_x & a_y & a_z & -\underline{a}^T \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(B.7)}$$

Usando-se a equação (B.7), a equação (B.6) torna-se

$$ {}^{T}\Delta = T^{-1}\,\Delta T = \begin{bmatrix} \underline{n}(\underline{\delta}\times\underline{n}) & \underline{n}(\underline{\delta}\times\underline{s}) & \underline{n}(\underline{\delta}\times\underline{a}) & \underline{n}(\underline{\delta}\times\underline{p})+\underline{d}\,\underline{n} \\ \underline{s}(\underline{\delta}\times\underline{n}) & \underline{s}(\underline{\delta}\times\underline{s}) & \underline{s}(\underline{\delta}\times\underline{a}) & \underline{s}(\underline{\delta}\times\underline{p})+\underline{d}\,\underline{n} \\ \underline{a}(\underline{\delta}\times\underline{n}) & \underline{a}(\underline{\delta}\times\underline{s}) & \underline{a}(\underline{\delta}\times\underline{a}) & \underline{a}(\underline{\delta}\times\underline{p})+\underline{d}\,\underline{n} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(B.8)} $$

onde $\underline{\delta} = (\delta_x\ ,\ \delta_y\ ,\ \delta_z)^t$ é a rotação diferencial sobre os eixos principais do sistema de coordenadas da base e $\underline{d} = (d_x\ ,\ d_y\ ,\ d_z)^t$ é a translação ao longo dos eixos principais do sistema de coordenadas da base. Utilizando as identidades vetoriais

$$ \underline{x}\,(\underline{y}\times\underline{z}) = -\,\underline{y}\,(\underline{x}\times\underline{z}) = \underline{y}\,(\underline{z}\times\underline{x}) $$

$$ \underline{x}\,(\underline{x}\times\underline{y}) = 0 $$

a equação (B.6) torna-se

$$ {}^{T}\Delta = \begin{bmatrix} 0 & -\underline{\delta}\,(\underline{n}\times\underline{s}) & \underline{\delta}\,(\underline{a}\times\underline{n}) & \underline{\delta}\,(\underline{p}\times\underline{n})+\underline{d}\,\underline{n} \\ \underline{\delta}\,(\underline{n}\times\underline{s}) & 0 & \underline{\delta}\,(\underline{s}\times\underline{a}) & \underline{\delta}\,(\underline{p}\times\underline{s})+\underline{d}\,\underline{s} \\ -\underline{\delta}\,(\underline{a}\times\underline{n}) & \underline{\delta}\,(\underline{s}\times\underline{a}) & 0 & \underline{\delta}\,(\underline{p}\times\underline{a})+\underline{d}\,\underline{a} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(B.9)} $$

Desde que os vetores $\underline{n}$, $\underline{s}$ e $\underline{a}$ sejam ortonormais, isto é:

$$\underline{n} \times \underline{s} = \underline{a} \qquad \underline{s} \times \underline{a} = \underline{n} \qquad \underline{a} \times \underline{n} = \underline{s}$$

então a equação (B.9) torna-se:

$$ {}^{T}\Delta = \begin{bmatrix} 0 & -\underline{\delta}\ \underline{a} & \underline{\delta}\ \underline{s} & \underline{\delta}\ (\underline{p} \times \underline{n}) + \underline{d}\ \underline{n} \\ \underline{\delta}\ \underline{a} & 0 & \underline{\delta}\ \underline{n} & \underline{\delta}\ (\underline{p} \times \underline{s}) + \underline{d}\ \underline{s} \\ -\underline{\delta}\ \underline{s} & \underline{\delta}\ \underline{n} & 0 & \underline{\delta}\ (\underline{p} \times \underline{a}) + \underline{d}\ \underline{a} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(B.10)} $$

Se fizermos os elementos da matriz de ${}^{T}\Delta$ serem:

$$ {}^{T}\Delta = \begin{bmatrix} 0 & -{}^{T}\delta_z & {}^{T}\delta_y & {}^{T}d_x \\ {}^{T}\delta_z & 0 & -{}^{T}\delta_x & {}^{T}d_y \\ -{}^{T}\delta_y & {}^{T}\delta_x & 0 & {}^{T}d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(B.11)} $$

e igualando os elementos das equações (B.10) e (B.11) têm-se:

$$ {}^{T}d_x = \underline{\delta}\ (\underline{p} \times \underline{n}) + \underline{d}\ \underline{n} = \underline{n}\ [\ (\underline{\delta} \times \underline{p}\ ) + \underline{d}\ ] $$

$$ {}^{T}d_y = \underline{\delta}\ (\underline{p} \times \underline{s}) + \underline{d}\ \underline{s} = \underline{s}\ [\ (\underline{\delta} \times \underline{p}\ ) + \underline{d}\ ] $$

$$ {}^{T}d_z = \underline{\delta}\ (\underline{p} \times \underline{a}) + \underline{d}\ \underline{a} = \underline{a}\ [\ (\underline{\delta} \times \underline{p}\ ) + \underline{d}\ ] \qquad \text{(B.12)} $$

$${}^{T}\delta_x = \underline{\delta}\,\underline{n}$$

$${}^{T}\delta_y = \underline{\delta}\,\underline{s}$$

$${}^{T}\delta_z = \underline{\delta}\,\underline{a}$$

Expressando o conjunto de equações B.12 na forma matricial, têm-se:

$$\begin{bmatrix} {}^{T}d_x \\ {}^{T}d_y \\ {}^{T}d_z \\ {}^{T}\delta_x \\ {}^{T}\delta_y \\ {}^{T}\delta_z \end{bmatrix} = \begin{bmatrix} [\underline{n}, \underline{s}, \underline{a}]^T & [(\underline{p}\times\underline{n}),(\underline{p}\times\underline{s}),(\underline{p}\times\underline{a})]^T \\ 0 & [\underline{n}, \underline{s}, \underline{a}]^T \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \qquad \text{(B.13)}$$

onde 0 é uma submatriz 3×3,com todos os elementos iguais a zero. A equação (B.13) mostra a relação da translação e rotação diferenciais do sistema de coordenadas da base em relação a translação e rotação diferenciais do sistema de coordenadas de T.

Aplicando a equação (B.5) à equação cinemática de um manipulador de seis graus de liberdade, obtêm-se o diferencial de ${}^{0}T_6$:

$$d\,{}^{0}T_6 = {}^{0}T_6\,{}^{T_6}\Delta \qquad \text{(B.14)}$$

No caso de um manipulador de seis graus de liberdade, um movimento diferencial na junta *i* pode induzir uma mudança equivalente em ${}^{0}T_6\,{}^{T_6}\Delta$:

$$d\,{}^{0}T_{6} = {}^{0}T_{6}\,{}^{T_6}\Delta = {}^{0}A_{1}\,{}^{1}A_{2}\ldots{}^{i-2}A_{i-1}\,{}^{i-1}\Delta_{i}\,{}^{i-1}A_{i}\ldots{}^{5}A_{6} \qquad \text{(B.15)}$$

onde ${}^{i-1}\Delta_{i}$ é definido como a transformação diferencial ao longo e sobre o eixo da junta *i* do movimento e é definido como:

$${}^{i-1}\Delta_{i} = \begin{bmatrix} 0 & -d\theta_{i} & 0 & 0 \\ d\theta_{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(B.16)}$$

A partir da equação (B.15), obtêm-se ${}^{T_6}\Delta$, devido a mudança diferencial no movimento da junta *i*:

$${}^{T_6}\Delta = ({}^{i-1}A_{i}\,{}^{i}\Delta_{i+1}\ldots{}^{5}A_{6})^{-1}\,{}^{i-1}\Delta_{i}\,({}^{i-1}A_{i}\,{}^{i}\Delta_{i+1}\ldots{}^{5}A_{6})$$

ou

$${}^{T_6}\Delta = U_{i}^{-1}\,{}^{i-1}\Delta_{i}\,U_{i} \qquad \text{(B.17)}$$

onde

$$U_{i} = {}^{i-1}A_{i}\,{}^{i}\Delta_{i+1}\ldots{}^{5}A_{6}$$

Expressando U_{i} na forma geral da matriz de transformação homogênea, isto é:

$$U_i = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.18}$$

e utilizando as equações (B.16) e (B.18) a equação (B.17) torna-se:

$${}^{T_6}\Delta = \begin{bmatrix} 0 & -a_z & s_z & p_x n_y - p_y n_x \\ a_z & 0 & -n_z & p_x s_y - p_y s_x \\ -s_z & n_z & 0 & p_x a_y - p_y a_x \\ 0 & 0 & 0 & 0 \end{bmatrix} d\theta_i \tag{B.19}$$

A partir dos elementos de ${}^{T_6}\Delta$ definidos na equação (B.11), e igualando os elementos das matrizes nas equações (B.11) e (B.19), têm-se:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = \begin{bmatrix} p_x n_y - p_y n_x \\ p_x s_y - p_y s_x \\ p_x a_y - p_y a_x \\ n_z \\ s_z \\ a_z \end{bmatrix} d\theta_i \tag{B.20}$$

onde i = 1, 2, . . ., 6.

Então, o Jacobiano de um manipulador pode ser obtido a patir da equação (B.20). Para um manipulador de seis graus de liberdade:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = \mathrm{J}(\underline{q}) \begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix} \tag{B.21}$$

onde, as colunas da matriz Jacobiana são obtidas a partir da equação (B.20).

ANEXO C

Inversão matricial pelo método de Gauss

Um algorítmo de execução bastante simples para inverter uma matriz A (n x n) pode ser obtido através de uma adaptação do método de eliminação de A seguir será apresentada esta adaptação para uma matriz 3X3.

Seja a matriz 3X3

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad (C.1)$$

onde det A $\neq$ 0 e sua inversa dada por:

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \qquad (C.2)$$

Os três vetores coluna da matriz inversa de serão chamados de $\underline{b}_1$, $\underline{b}_2$ e $\underline{b}_3$. Pela definição de matriz inversa, tem-se:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(C.3)}$$

Quando se considera a formação da primeira coluna da matriz identidade pela multiplicão A por B, tem-se:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{(C.4)}$$

Deste modo para determinar a 1ª coluna b_1 da matriz inversa, basta resolver o sistema

$$A\,\underline{b}_1 = \underline{e}_1 \text{ onde } \underline{e}_1 = (1\ 0\ 0)^T \qquad \text{(C.5)}$$

Do mesmo modo temos para a 2ª e 3ª colunas. Assim. para obter b_2 resolvemos o sistema

$$A\,\underline{b}_2 = \underline{e}_2 \text{ onde } \underline{e}_2 = (0\ 1\ 0)^T \qquad \text{(C.6)}$$

e para obter b_3

$$A\,\underline{b}_3 = \underline{e}_3 \text{ onde } \underline{e}_3 = (0\ 0\ 1)^T \qquad \text{(C.7)}$$

Os três sistemas, C.5, C.6 e C.7, são resolvidos pelo método de Gauss

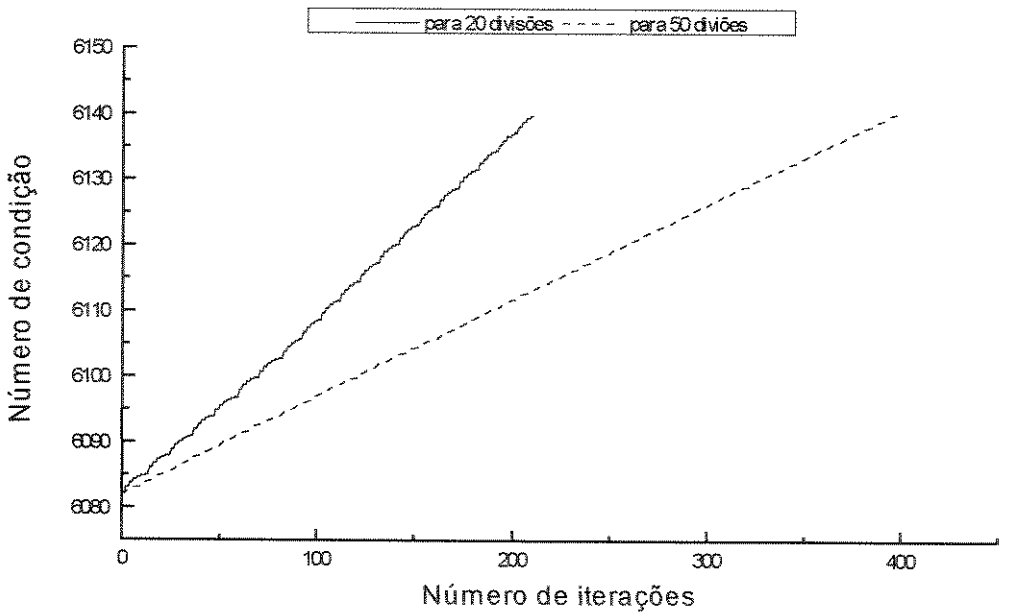


Figura 5.26 : Comportamento numérico do número de condição para a configuração 1 (800,0,933.1,21,58,90) utilizando o segundo método de Miss.

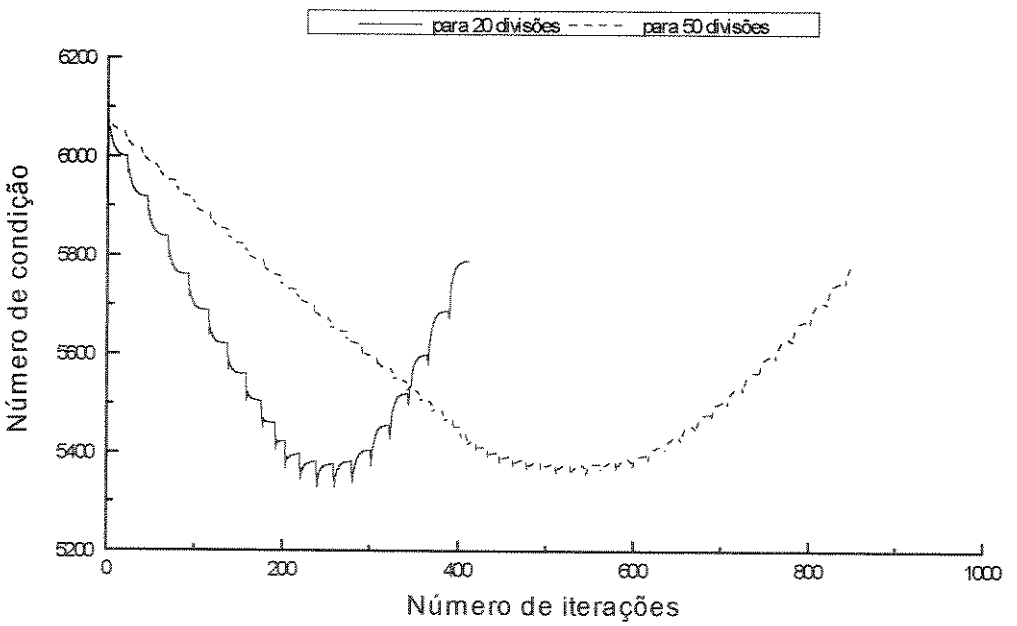


Figura 5.27 : Comportamento numérico do número de condição para a configuração 6 (458,658,521,52,14,62) utilizando o segundo método de Miss.

ANEXO D

Manipulador Kraft

O manipulador Kraft, figura d.1, possui seis juntas rotacionais e foi desenvolvido para executar tarefas gerais em ambientes hostis e submarinos. Os seus movimentos são comandados a distância através de um controle chamado "master", que é um modelo em escala reduzida do manipulador.

Suas trajetórias podem ser definidas pelo operador ou por programações pré definidas. O sistema completo robô tele-operado é denominado sistema robótico.

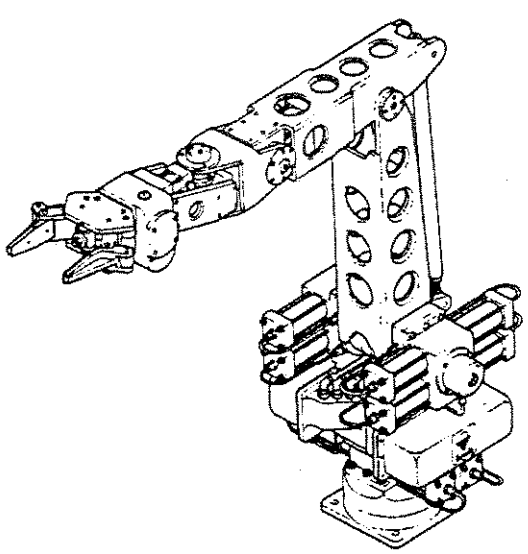


Figura d.1: Manipulador Kraf.

A representação e os parâmetros de Denavit-Hartenberg, para o manipulador Kraft, são apresentados na figura d.2 e na tabela d.1, respectivamente

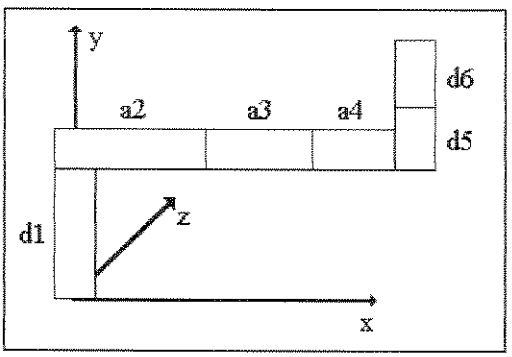


Figura d.2 : Representação do manipulador Kraft.

JUNTA	θ (graus)	d (mm)	α (graus)	a (mm)	RANGE (graus)
1	θ_1	d_1	90.0	0.0	- 90.0 /+ 90.0
2	θ_2	0.0	0.0	a_2	0 / + 120.0
3	θ_3	0.0	0.0	a_3	0.0/-130.0
4	θ_4	0.0	- 90.0	a_4	-42.0 /+58.0
5	θ_5	d_5	90.0	0.0	+ 34.0 / + 134.0
6	θ_6	d_6	0.0	0.0	- 90.0 /+ 90.0

Tabela d.1 : Parâmetros de Denavit - Hartenberg para o Manipulador Kraft.

As matrizes de passagem, tabela d.2, são obtidas utilizando-se os parâmetros de Denavit-Hartenberg e a equação A.1.

$$T_{0,1} = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_{1,2} = \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_{2,3} = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_{3,4} = \begin{bmatrix} c_4 & 0 & -s_4 & a_4 c_4 \\ s_4 & 0 & c_4 & a_4 s_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_{4,5} = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_{5,6} = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Tabela d.2 : Matrizes de passagem para o Manipulador Kraft.

Onde s_i significa sen(θ_i) e c_i cos(θ_i).

O modelo geométrico obtido através da equação A.2, é dado por:

Orientação

$n_x = c_1 (c_5 c_6 c_{234} - s_6 s_{234}) - s_1 s_5 c_6,$

$n_y = s_1 (c_5 c_6 c_{234} - s_6 s_{234}) + c_1 s_5 c_6,$

$n_z = c_5 c_6 s_{234} - s_6 c_{234},$

$s_x = - c_1 (c_5 s_6 c_{234} + c_6 s_{234}) + s_1 s_5 s_6,$

$s_y = - s_1 (c_5 s_6 c_{234} + c_6 s_{234}) - c_1 s_5 s_6,$

$s_z = - c_5 s_6 s_{234} + c_6 c_{234},$

$a_x = c_1 s_5 c_{234} + s_1 c_5,$

$a_y = s_1 s_5 c_{234} - c_1 c_5,$

$a_z = s_5 s_{234}.$

Posição

$$p_x = d_6 (c_1 s_5 c_{234} + s_1 c_5) + c_1 (- d_5 s_{234} + a_4 c_{234} + a_3 c_{23} + a_2 c_2),$$

$$p_y = d_6 (s_1 s_5 c_{234} - c_1 c_5) + s_1 (- d_5 s_{234} + a_4 c_{234} + a_3 c_{23} + a_2 c_2),$$

$$p_z = d_6 s_5 s_{234} + d_5 c_{234} + a_4 s_{234} + a_3 s_{23} + a_2 s_2 + d_1 .$$

Onde

$$c_{23} = c_2 c_3 - s_2 s_3 ,$$

$$s_{23} = s_2 c_3 + s_3 c_2 ,$$

$$c_{234} = c_{23} c_4 - s_{23} s_4 ,$$

$$s_{234} = c_{23} s_4 - s_{23} c_4 .$$

Os valores númericos dos parâmetros de Denavit-Hartenberg para o manipulador submarino Kraft, são apresendados na tabela d.3.

Parâmetros	(mm)
a2	532.65
a3	264.32
a4	132.16
d1	352.43
d5	48.06
d6	380.46

Tabela d.3: Valores dos parâmetros de Denavit-Hartenberg.

ANEXO E

Estudo dos erros

As tabelas e.1 e e.2 apresentadas a seguir contém os erros obtidos em relação a posição e orientação final. As tabelas e.3 e e.4 o valor angular final das juntas após o termino da convergência para cada configuração desejada e valor de *m* (número de divisões do caminho desejado) utilizado.

Os erros e_x, e_y e e_z (erros de posição) são apresentados em mm e os erros $e_{\underline{n}}$, $e_{\underline{s}}$ e $e_{\underline{a}}$ (erros de orientação) são apresentados em graus. Os valores angulares finais são apresentados em graus.

A tabela a seguir, já apresentada no capítulo 5, apresenta as posições e orientações desejadas utilizadas para as simulações.

config.	X (mm)	Y (mm)	Z (mm)	Psi (graus)	Teta(graus)	Phi(graus)
1	800.0	0.0	933.1	21.0	58.0	90.0
2	776.9	0.0	700.0	25.0	63.0	75.0
3	776.9	456.0	933.1	85.0	62.0	14.0
4	250.0	-45.0	450.0	45.0	62.0	14.0
5	800.0	0.0	933.1	21.0	58.0	90.0
6	800.0	0.0	600.0	21.0	58.0	91.0
7	776.9	0.0	600.0	21.0	58.0	90.0
8	776.9	0.0	933.1	21.0	58.0	91.0
9	458.0	658.0	521.0	52.0	14.0	62.0

Tabela 4.1 : Posição e orientação desejadas, $\underline{X}_d$.

m *Config.*	***20***	***50***	***70***
1	$e_x = 0.0$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$
2	$e_x = 0.2$ $e_y = 0.0$ $e_z = 0.1$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$
3	$e_x = 0.6$ $e_y = 0.1$ $e_z = 0.2$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.1$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.0$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$
4	$e_x = 0.0$ $e_y = 0.1$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.3$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.0$ $e_y = 0.1$ $e_z = 0.0$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$
5 a 8	$e_x = 0.2$ $e_y = 0.0$ $e_z = 4.8$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.1$ $e_{\underline{a}} = 0.0$	$e_x = 0.3$ $e_y = 0.0$ $e_z = 4.6$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$	$e_x = 0.3$ $e_y = 0.0$ $e_z = 4.6$ $e_{\underline{n}} = 0.0$ $e_{\underline{s}} = 0.0$ $e_{\underline{a}} = 0.0$

Tabela e.1: Erros de orientação e posição da garra utilizando os métodos de Gauss e Greville.

	Primeiro método		Segundo método	
m *Config.*	***20***	***50***	***20***	***50***
1	e_x = 7.9 e_y = 0.0 e_z = 2.0 $e_{\underline{n}}$ = 72.3 $e_{\underline{s}}$ = 69.0 $e_{\underline{a}}$ = 31.8	e_x = 7.9 e_y = 0.0 e_z = 2.0 $e_{\underline{n}}$ = 72.3 $e_{\underline{s}}$ = 69.0 $e_{\underline{a}}$ = 31.8	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 159.0 $e_{\underline{s}}$ = 142.7 $e_{\underline{a}}$ = 31.6	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 159.0 $e_{\underline{s}}$ = 142.7 $e_{\underline{a}}$ = 31.6
2	e_x = 3.0 e_y = 0.0 e_z = 4.8 $e_{\underline{n}}$ = 68.3 $e_{\underline{s}}$ = 72.4 $e_{\underline{a}}$ = 49.8	e_x = 2.3 e_y = 0.0 e_z = 6.0 $e_{\underline{n}}$ = 68.3 $e_{\underline{s}}$ = 72.4 $e_{\underline{a}}$ = 49.7	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 157.7 $e_{\underline{s}}$ = 126.2 $e_{\underline{a}}$ = 48.8	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 157.7 $e_{\underline{s}}$ = 126.2 $e_{\underline{a}}$ = 48.8
3	e_x = 4.6 e_y = 3.6 e_z = 1.6 $e_{\underline{n}}$ = 26.7 $e_{\underline{s}}$ = 35.5 $e_{\underline{a}}$ = 38.8	e_x = 4.7 e_y = 3.7 e_z = 1.6 $e_{\underline{n}}$ = 26.7 $e_{\underline{s}}$ = 35.4 $e_{\underline{a}}$ = 38.7	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 112.0 $e_{\underline{s}}$ = 97.1 $e_{\underline{a}}$ = 45.0	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 111.9 $e_{\underline{s}}$ = 97.0 $e_{\underline{a}}$ = 45.0
4	nc	nc	nc	nc
6	e_x = 2.1 e_y = 6.2 e_z = 1.6 $e_{\underline{n}}$ = 104.5 $e_{\underline{s}}$ = 22.9 $e_{\underline{a}}$ = 109.2	e_x = 2.1 e_y = 6.1 e_z = 1.5 $e_{\underline{n}}$ = 104.6 $e_{\underline{s}}$ = 22.7 $e_{\underline{a}}$ = 109.2	e_x = 0.0 e_y = 0.1 e_z = 0.0 $e_{\underline{n}}$ = 91.4 $e_{\underline{s}}$ = 86.1 $e_{\underline{a}}$ = 110.7	e_x = 0.0 e_y = 0.0 e_z = 0.0 $e_{\underline{n}}$ = 91.5 $e_{\underline{s}}$ = 86.3 $e_{\underline{a}}$ = 110.8

Tabela e.2 : Erros de orientação e posição da garra utilizando os métodos de Miss.

m *Config.*	***20***	***50***	***70***
1	$\theta_1 = 0.0$ $\theta_2 = 64.19$ $\theta_3 = -117.25$ $\theta_4 = 85.07$ $\theta_5 = 90.0$ $\theta_6 = 159.0$	$\theta_1 = 0.0$ $\theta_2 = 64.19$ $\theta_3 = -117.25$ $\theta_4 = 85.06$ $\theta_5 = 90.0$ $\theta_6 = 159.0$	$\theta_1 = 0.0$ $\theta_2 = 64.19$ $\theta_3 = -117.25$ $\theta_4 = 85.06$ $\theta_5 = 90.0$ $\theta_6 = 159.0$
2	$\theta_1 = -11.05$ $\theta_2 = 37.81$ $\theta_3 = -139.67$ $\theta_4 = 131.42$ $\theta_5 = 113.03$ $\theta_6 = 167.51$	$\theta_1 = -11.05$ $\theta_2 = 37.81$ $\theta_3 = -139.63$ $\theta_4 = 131.38$ $\theta_5 = 113.03$ $\theta_6 = 167.51$	$\theta_1 = -11.05$ $\theta_2 = 37.81$ $\theta_3 = -139.63$ $\theta_4 = 131.38$ $\theta_5 = 113.03$ $\theta_6 = 167.51$
3	$\theta_1 = 10.60$ $\theta_2 = 39.78$ $\theta_3 = -55.71$ $\theta_4 = 67.87$ $\theta_5 = 143.41$ $\theta_6 = 140.74$	$\theta_1 = 10.59$ $\theta_2 = 39.72$ $\theta_3 = -55.55$ $\theta_4 = 67.77$ $\theta_5 = 143.41$ $\theta_6 = 140.73$	$\theta_1 = 10.59$ $\theta_2 = 39.72$ $\theta_3 = -55.53$ $\theta_4 = 67.77$ $\theta_5 = 143.41$ $\theta_6 = 140.75$
4	$\theta_1 = -65.54$ $\theta_2 = 16.01$ $\theta_3 = -99.98$ $\theta_4 = -131.10$ $\theta_5 = 123.31$ $\theta_6 = -65.45$	$\theta_1 = -65.53$ $\theta_2 = 160.2$ $\theta_3 = -99.13$ $\theta_4 = -131.06$ $\theta_5 = 123.32$ $\theta_6 = -65.45$	$\theta_1 = -65.53$ $\theta_2 = 16.01$ $\theta_3 = -99.11$ $\theta_4 = -131.08$ $\theta_5 = 123.31$ $\theta_6 = -65.45$
5 a 8	$\theta_1 = 0.71$ $\theta_2 = 65.39$ $\theta_3 = -121.96$ $\theta_4 = 88.38$ $\theta_5 = 88.57$ $\theta_6 = 158.1$	$\theta_1 = 0.70$ $\theta_2 = 65.39$ $\theta_3 = -121.96$ $\theta_4 = 88.58$ $\theta_5 = 88.57$ $\theta_6 = 158.11$	$\theta_1 = 0.70$ $\theta_2 = 65.39$ $\theta_3 = -121.96$ $\theta_4 = 88.58$ $\theta_5 = 88.57$ $\theta_6 = 158.11$

Tabela e.3 : Posição angular final das juntas após o termino da convergência utilizando os métodos de Gauss e Greville.

	Primeiro método		Segundo método	
m *Config.*	***20***	***50***	***20***	***50***
1	$\theta_1 = 0.0$ $\theta_2 = 88.35$ $\theta_3 = -89.35$ $\theta_4 = 1.01$ $\theta_5 = 90.0$ $\theta_6 = 90.0$	$\theta_1 = 0.0$ $\theta_2 = 88.35$ $\theta_3 = -89.15$ $\theta_4 = 1.01$ $\theta_5 = 90.0$ $\theta_6 = 90.0$	$\theta_1 = 0.0$ $\theta_2 = 87.48$ $\theta_3 = -88.15$ $\theta_4 = 1.07$ $\theta_5 = 90.0$ $\theta_6 = 0.0$	$\theta_1 = 0.0$ $\theta_2 = 87.48$ $\theta_3 = -88.15$ $\theta_4 = 1.07$ $\theta_5 = 90.0$ $\theta_6 = 0.01$
2	$\theta_1 = 0.0$ $\theta_2 = 87.57$ $\theta_3 = -97.24$ $\theta_4 = -1.92$ $\theta_5 = 90.0$ $\theta_6 = 90.0$	$\theta_1 = 0.0$ $\theta_2 = 87.67$ $\theta_3 = -97.26$ $\theta_4 = -10.93$ $\theta_5 = 90.0$ $\theta_6 = 90.0$	$\theta_1 = 0.0$ $\theta_2 = 87.85$ $\theta_3 = -100.57$ $\theta_4 = -6.86$ $\theta_5 = 90.0$ $\theta_6 = 90.0$	$\theta_1 = 0.0$ $\theta_2 = 87.85$ $\theta_3 = -100.57$ $\theta_4 = 6.86$ $\theta_5 = 90.0$ $\theta_6 = 0.01$
3	$\theta_1 = 21.05$ $\theta_2 = 74.56$ $\theta_3 = -79.64$ $\theta_4 = 9.95$ $\theta_5 = 112.37$ $\theta_6 = 90.0$	$\theta_1 = 20.98$ $\theta_2 = 74.54$ $\theta_3 = -79.62$ $\theta_4 = 9.96$ $\theta_5 = 112.54$ $\theta_6 = 90.0$	$\theta_1 = 25.98$ $\theta_2 = 75.69$ $\theta_3 = -78.54$ $\theta_4 = 6.20$ $\theta_5 = 100.55$ $\theta_6 = 0.02$	$\theta_1 = 25.98$ $\theta_2 = 75.69$ $\theta_3 = -78.53$ $\theta_4 = 6.20$ $\theta_5 = 100.54$ $\theta_6 = 0.03$
4	nc	nc	nc	nc
6	$\theta_1 = 36.60$ $\theta_2 = 72.56$ $\theta_3 = -99.45$ $\theta_4 = -10.99$ $\theta_5 = -131.38$ $\theta_6 = 60.81$	$\theta_1 = 36.61$ $\theta_2 = 72.55$ $\theta_3 = -99.43$ $\theta_4 = -10.99$ $\theta_5 = 131.36$ $\theta_6 = 61.01$	$\theta_1 = 45.36$ $\theta_2 = 76.74$ $\theta_3 = -102.93$ $\theta_4 = -7.88$ $\theta_5 = 111.01$ $\theta_6 = 0.01$	$\theta_1 = 45.38$ $\theta_2 = 76.74$ $\theta_3 = -102.94$ $\theta_4 = -7.88$ $\theta_5 = 110.98$ $\theta_6 = 0.01$

Tabela e.4 : Posição angular final das juntas após o termino da convergência utilizando os métodos de Miss.

ANEXO F

Listagem do programa funcões.ada

Este programa (sintaxe ADA) contém os algorítmos apresentados neste trabalho, necessários para a geração de uma trajetória utilizando o modelo cinematico inverso.

```
-- Autores :      CLAUDIO EDUARDO ARAVÉCHIA DE SÁ
--                JOÃO MAURÍCIO ROSÁRIO

with
math_cad,
math_init,
defin;

use
math_cad,
math_init,
defin;

package body funcoes is

type vector is array ( 1 .. 3 ) of float;

-- CALCULO DA MATRIZ ORIENTAÇÃO FINAL DESEJADA
function euler(psi_e,tet_e,phi_e : in float) return matrix is

   cos_psi,sin_psi,sin_teta,cos_teta,sin_phi,cos_phi : float;
   ma_euler : matrix;

begin

cos_psi:=cos(psi_e);
sin_psi:=sin(psi_e);
cos_teta:=cos(tet_e);
sin_teta:=sin(tet_e);
cos_phi:=cos(phi_e);
sin_phi:=sin(phi_e);
ma_euler(1,1):= cos_psi*cos_phi - sin_psi*cos_teta*sin_phi;
ma_euler(2,1):= - cos_psi*sin_phi - sin_psi*cos_teta*cos_phi;
```

```
ma_euler(3,1):= sin_teta*sin_psi;
ma_euler(1,2):= sin_psi*cos_phi + cos_psi*cos_teta*sin_phi;
ma_euler(2,2):= - sin_psi*sin_phi + cos_psi*cos_teta*cos_phi;
ma_euler(3,2):= - sin_teta*cos_psi;
ma_euler(1,3):= sin_teta*sin_phi;
ma_euler(2,3):= sin_teta*cos_phi;
ma_euler(3,3):= cos_teta;

return ma_euler;

end euler;

-- MODELO CINEMATICO DIRETO KRAFT
function md_kraft( t_a : in vetor6 ) return matrix4 is

   atual : matrix4;

begin

-- calculo da matriz de euler referente a posicao atual
s1 := sin (t_a(1));
s2 := sin (t_a(2));
s3 := sin (t_a(3));
s4 := sin (t_a(4));
s5 := sin (t_a(5));
s6 := sin (t_a(6));
c1 := cos (t_a(1));
c2 := cos (t_a(2));
c3 := cos (t_a(3));
c4 := cos (t_a(4));
c5 := cos (t_a(5));
c6 := cos (t_a(6));
s23 := s2*c3 + s3*c2;
c23 := c2*c3 - s2*s3;
s234 := c23*s4 + s23*c4;
c234 := c23*c4 - s23*s4;

atual(1,1) := c1*(c5*c6*c234 - s6*s234) - s1*s5*c6;
atual(2,1) := s1*(c5*c6*c234 - s6*s234) + c1*s5*c6;
atual(3,1) := c5*c6*s234 + s6*c234;

atual(1,2) := - c1*(c5*s6*c234 + c6*s234) + s1*s5*s6;
atual(2,2) := - s1*(c5*s6*c234 + c6*s234) - c1*s5*s6;
atual(3,2) := - c5*s6*s234 + c6*c234;

atual(1,3) := c1*s5*c234 + s1*c5;
atual(2,3) := s1*s5*c234 - c1*c5;
atual(3,3) := s5*s234;

atual(1,4) := d6*( c1*s5*c234 + s1*c5 ) + c1*(-d5*s234 + a4*c234 + a3*c23 + a2*c2);
atual(2,4) := d6*( s1*s5*c234 - c1*c5 ) + s1*(-d5*s234 + a4*c234 + a3*c23 + a2*c2);
atual(3,4) := d6*s5*s234 + d5*c234 + a4*s234 + a3*s23 + a2*s2 + d1;

return atual;

end md_kraft;

-- CALCULO DA MATRIZ JACOBIANA
```

```
function jacobiano(t_a : in vetor6) return matrix6 is

   J : matrix6;
   A01,A12,A23,A34,A45,A56 : matrix4;
   T1,t06,t16,t26,t36,t46,t56 : matrix4;
   dx_ja,dy_ja,dz_ja,deltx_ja,delty_ja,deltz_ja : float;
   n_o_ja,s_o_ja,a_o_ja,p_p_ja : vector;

begin

-- definicao das matrizes de passagem KRAFT
s1 := sin (t_a(1));
s2 := sin (t_a(2));
s3 := sin (t_a(3));
s4 := sin (t_a(4));
s5 := sin (t_a(5));
s6 := sin (t_a(6));
c1 := cos (t_a(1));
c2 := cos (t_a(2));
c3 := cos (t_a(3));
c4 := cos (t_a(4));
c5 := cos (t_a(5));
c6 := cos (t_a(6));
   -- matrizes de passagem ( KRAFT )
   A01 := (1=> (c1,0.0,s1,0.0),   2=> (s1,0.0,-c1,0.0),
         3=> (0.0,1.0,0.0,d1),  4=> (0.0,0.0,0.0,1.0));
   A12 := (1=> (c2,-s2,0.0,a2*c2),2=> (s2,c2,0.0,a2*s2),
         3=> (0.0,0.0,1.0,0.0), 4=> (0.0,0.0,0.0,1.0));
   A23 := (1=> (c3,-s3,0.0,a3*c3),2=> (s3,c3,0.0,a3*s3),
         3=> (0.0,0.0,1.0,0.0), 4=> (0.0,0.0,0.0,1.0));
   A34 := (1=> (c4,0.0,-s4,a4*c4),2=> (s4,0.0,c4,a4*s4),
         3=> (0.0,-1.0,0.0,0.0),4=> (0.0,0.0,0.0,1.0));
   A45 := (1=> (c5,0.0,s5,0.0),   2=> (s5,0.0,-c5,0.0),
         3=> (0.0,1.0,0.0,d5),  4=> (0.0,0.0,0.0,1.0));
   A56 := (1=> (c6,-s6,0.0,0.0),  2=> (s6,c6,0.0,0.0),
         3=> (0.0,0.0,1.0,d6),  4=> (0.0,0.0,0.0,1.0));

t06:= (A01*A12)*(A23*A34)*(A45*A56);
t16:= (A12*A23)*(A34*A45)*A56;
t26:= (A23*A34)*(A45*A56);
t36:= (A34*A45)*A56;
t46:= (A45*A56);
t56:= A56;

for i in 1 .. 6 loop
      for y in 1 .. nl-2  loop
            for k in 1 .. nc-2 loop
                  T1(y,k):=0.0;
            end loop;
      end loop;
      dx_ja:=0.0;
      dy_ja:=0.0;
      dz_ja:=0.0;
      deltx_ja:=0.0;
      delty_ja:=0.0;
      deltz_ja:=0.0;
      if i = 1 then
            T1 := t06;
```

```
        elsif i = 2 then
            T1 := t16;
        elsif i = 3 then
            T1 := t26;
        elsif i = 4 then
            T1 := t36;
        elsif i = 5 then
            T1 := t46;
        else
            T1 := t56;
       end if;
       for y in 1 .. 3 loop
            n_o_ja(y):=T1(y,1);
            s_o_ja(y):=T1(y,2);
            a_o_ja(y):=T1(y,3);
            p_p_ja(y):=T1(y,4);
       end loop;
       dx_ja:= (- n_o_ja(1)*p_p_ja(2)) + (n_o_ja(2)*p_p_ja(1));
       dy_ja:= (- s_o_ja(1)*p_p_ja(2)) + (s_o_ja(2)*p_p_ja(1));
       dz_ja:= (- a_o_ja(1)*p_p_ja(2)) + (a_o_ja(2)*p_p_ja(1));
       deltx_ja:= n_o_ja(3);
       delty_ja:= s_o_ja(3);
       deltz_ja:= a_o_ja(3);
       -- ordenacao da matriz jacobiana
       J(1,i):=dx_ja;
       J(2,i):=dy_ja;
       J(3,i):=dz_ja;
       J(4,i):=deltx_ja;
       J(5,i):=delty_ja;
       J(6,i):=deltz_ja;

end loop;

return J;

end jacobiano;

-- CALCULO DOS INCREMENTOS PELO METODO DE MISS ( MODELO 1 )
function miss1( D : in vetor6 ; J : in matrix6 ) return vetor6 is

   const1,const2,vmax,cmax,norme : float;
   som1,som2,som3,som4 : float;
   delta_in,v_max,v_g : vetor6;

begin

const1 := 1.0;
const2 := 1.0;
vmax := 0.9;
cmax := 0.2;
norme := 1.0;
v_max := (vmax,vmax,vmax,cmax,cmax,cmax);
for i in 1 .. 6  loop
      if abs(D(i)) > v_max(i) then
            const2 := abs(D(i))/v_max(i);
            if const2 > const1 then
                  const1 := const2;
            end if;
```

```
        end if;
end loop;
for i in 1 .. 6 loop
        v_g(i) := D(i)/const1;
end loop;

-- calculo dos incrementos pelo metodo de miss (metodo 1)
for k in 1 .. nl loop
        som1 := 0.0;
        som2 := 0.0;
        som3 := 0.0;
        som4 := 0.0;
        delta_in(k) := 0.0;
        for i in 1 .. 3 loop
                som1 := som1 + J(i,k) * v_g(i);
                som2 := som2 + J(i,k) * J(i,k);
                som3 := som3 + J(i+3,k) * v_g(i+3);
                som4 := som4 + J(i+3,k) * J(i+3,k);
        end loop;
        delta_in(k):= (som1*norme + som3)/(som2*norme + som4);
end loop;

return delta_in;

end miss1;

-- CALCULO DOS INCREMENTOS PELO METODO DE MISS ( MODELO 2 )
function miss2( D : in vetor6 ; J : in matrix6 ) return vetor6 is

   const1,const2,vmax,cmax,mud1,mud2,varia2,tal : float;
   delta_in,v_max,v_g,der_s : vetor6;

begin

for i in 1 .. 6 loop
        v_g(i) := D(i);
end loop;

-- calculo dos incrementos pelo metodo de miss (metodo 2)
der_s := (0.0,0.0,0.0,0.0,0.0,0.0,0.0);
for i in 1 .. nl loop
        for k in 1 .. nl loop
                der_s(i) := der_s(i) +(-2*v_g(k)*J(k,i));
        end loop;
end loop;
mud1 := 0.0;
mud2 := 0.0;
for i in 1 .. nl loop
        varia2 := 0.0;
        for k in 1 .. nc loop
                varia2 := varia2 +(J(i,k)*der_s(k));
        end loop;
        mud1 := mud1 + varia2*varia2;
        mud2 := mud2 + v_g(i)*varia2;
end loop;
tal := ( mud2 / mud1 )*0.95;
for i in 1 .. nl loop
        delta_in(i) := tal * der_s(i);
```

```
end loop;

return delta_in;

end miss2;

-- INVERSÃO MATRICIAL PELO METODO DE GAUSS
function gauss( J : in matrix6 ) return matrix6 is

   k,ip,jp: integer ;
   som,tempo,const: float ;
   vet_so : matcar;
   b,temp : lign ;
   r,matriz_in : matrix6;
   s_col,delta_in,pivot : vetor6 ;
   mt : matcar;
   test_rank : boolean := true;

begin

for h in 1 .. nl loop
      for j in 1 .. nc loop
            vet_so(h)(j):=0.0;
      end loop;
end loop;

for i in 1 .. nl loop
      for h in 1 .. nc loop
            mt(i)(h) := J(i,h);
      end loop;
end loop;
-- matriz identidade
for i in 1 .. nl loop
      for j in 1 .. nc loop
            if i = j then
                  r (i,i) := 1.0 ;
            else
                  r (i,j) :=  0.0 ;
            end if;
      end loop;
end loop;
for k in 1..nl loop
      -- procura do pivo
      pivot(k) := mt(k)(k);
      ip := k ;
      jp := k ;
      for i in k .. nl loop
            for j in k .. nc loop
                  if (abs(mt(i)(j)) >= abs(pivot(k))) or
                    (abs(mt(i)(j)) = abs(pivot(k))) then
                        pivot(k) := mt(i)(k);
                        ip := i ;
                        jp := k ;
                  end if ;
            end loop ;
      end loop;
      -- permutacao de linha (mt)
    if abs(pivot(k)) > 0.00001 then
```

```
      if ip /= k then
      temp := mt(k) ;
      mt(k) := mt(ip) ;
      mt(ip) := temp ;
      for l in 1 .. nl  loop
            tempo := r(k,l) ;
            r(k,l) := r(ip,l) ;
            r(ip,l) := tempo ;
      end loop ;
      end if ;
      -- permutacao de colunas
      if jp /= k then
      for l in 1 .. nl loop
            tempo := mt(l)(k) ;
            mt(l)(k) := mt(l)(jp) ;
            mt(l)(jp) := tempo ;
      end loop ;
      end if ;
      -- triangularisacao
      for i in K+1 .. nl loop
            if mt(i)(k) /= 0.0 then
                  const := - mt(i)(k)/pivot(k) ;
                  -- modificacao de mt
                  for j in K+1  .. nc loop
                        mt(i)(j) := mt(i)(j)+mt(k)(j)*const ;
                  end loop ;
                  mt(i)(k) := 0.0 ;
                  -- modificacao de r
                  for j in 1 .. nl loop
                        r (i,j) := r(i,j) + r(k,j)*const ;
                  end loop ;
            end if;
      end loop ;
    end if;
end loop ;
-- determinacao dos componentes da matriz inversa
for y in 1 .. nl loop
      for u in 1 .. nl loop
            b(u) := r(u,y);
      end loop;
      vet_so(y)(nc) := b(nl) / (mt(nl)(nc));
      for i in reverse 1 .. nl-1 loop
            som := 0.0;
                  for j in i+1 .. nc loop
                        som := som + (mt(i)(j))*(vet_so(y)(j));
                  end loop;
            vet_so(y)(i) := (b(i) - som) / (mt(i)(i));
      end loop;
end loop;
-- ordenacao da matriz inversa
for y in 1 .. nl loop
      for j in 1 .. nc loop
            matriz_in(y,j):= vet_so(j)(y);
      end loop;
end loop;

return matriz_in;
```

```
end gauss;

-- TESTE DO RANK DA MATRIZ
function t_rank( matriz_gg : in matrix6 ) return integer is

   rank : integer;
   s_col : vetor6;

begin

rank := 0;
s_col := (0.0,0.0,0.0,0.0,0.0,0.0);
for i in 1 .. nl loop
      for j in 1 .. nc loop
            s_col(i) := s_col(i) + matriz_gg(i,j);
      end loop;
      if s_col(i) > 0.0 or s_col(i) < 0.0 then
            rank := rank + 1;
      end if;
end loop;

return rank;

end t_rank;

-- DETERMINAÇ¦O DA MATRIZ PSEUDOINVERSA PELO METODO DE GREVILLE
function greville( J : in matrix6 ) return matrix6 is

   a1, j1, dk, ve_temp, ck, ve_tempo, bk : ve;
   j_p, j2, m_temp : matrix6;
   sum, soma_a1, soma_d, temp5, soma_c : float;
   delta_in : vetor6;
   n : integer := 6;

begin

-- inicialazacao de a1 e jp
for u in 1 .. n loop
      a1(u) := J(u,1);
end loop;
soma_a1:=0.0;
for u in 1 .. n  loop
      soma_a1 := soma_a1 + a1(u)*a1(u);
end loop;

if soma_a1 = 0.0 then
      for u in 1 .. n loop
            j_p(1,u):= 0.0;
      end loop;
else
      for u in 1 ..n loop
            j_p(1,u) := a1(u) / soma_a1;
      end loop;
end if;
-- inicio do calculo da matriz pseudo_inversa
for k in 2 .. n loop
      for u in 1 .. n loop
            j1(u) := J(u,k);
```

```
end loop;
for u in 1 .. n loop
      dk(u):=0.0;
end loop;
for y in 1 .. n loop
      for u in 1 .. k-1 loop
            dk(u):=dk(u)+j_p(u,y)*j1(y);
      end loop;
end loop;
for u in 1 .. n loop
      for y in 1 .. k-1 loop
            j2(u,y) := 0.0;
            j2(u,y) := J(u,y);
      end loop;
end loop;
for u in 1 ..n loop
      ve_temp(u):=0.0;
end loop;
for u in 1 .. n loop
      for y in 1 .. k-1 loop
            ve_temp(u) := ve_temp(u) + j2(u,y)*dk(y);
      end loop;
end loop;
for u in 1 .. n loop
      ck(u) := j1(u) - ve_temp(u);
end loop;
soma_c:=0.0;
soma_d:=0.0;
for u in 1 .. n  loop
      soma_c := soma_c + ck(u)*ck(u);
end loop;
for u in 1 .. n loop
      soma_d := soma_d + dk(u)*dk(u);
end loop;
if soma_c < 0.00000001 then
      soma_c :=0.0;
end if;
if soma_c = 0.0 then
      temp5:= 1/(1+soma_d);
      for u in 1 .. n loop
            ve_tempo(u):= 0.0;
      end loop;
      for u in 1 .. n loop
            for y in 1 .. k-1 loop
                  ve_tempo(u) := ve_tempo(u)+dk(y)*j_p(y,u);
            end loop;
      end loop;
      for u in 1 .. n loop
            bk(u):=temp5*ve_tempo(u);
      end loop;
else
      for u in 1 .. n loop
            bk(u):=(1/soma_c)*ck(u);
      end loop;
end if;

for u in 1 .. n loop
      for y in 1 .. k-1 loop
```

```
                m_temp(y,u) := 0.0;
                m_temp(y,u) := dk(y)*bk(u);
            end loop;
        end loop;
        -- ordenacao da matriz pseudo_inversa ( j_p )
        for u in 1 .. n loop
            for y in 1 .. k-1 loop
                j_p(y,u) := j_p(y,u)-m_temp(y,u);
            end loop;
        end loop;
        for u in 1 .. n loop
            j_p(k,u):=bk(u);
        end loop;

end loop;

return j_p;

end greville;

--DETERMINAÇ¦O DOS INCREMENTOS PARA OS METODOS DE GAUSS E GREVILLE
function increm( matriz_gg : in matrix6 ; D : in vetor6) return vetor6 is

   delta_in : vetor6;

begin

-- calculo do incremento
for uu in 1 .. nl loop
        delta_in(uu):=0.0;
        for u in 1 .. nc loop
            delta_in(uu) :=delta_in(uu)+matriz_gg(uu,u)*D(u);
        end loop;
end loop;

return delta_in;

end increm;

-- DETERMINAÇ¦O DO QUADRANTE
function quad_co( t_a : in vetor6 ) return vetor6 is

   x,y,t_a_cand : float;
   t_a_r : vetor6;

begin

for i in 1 .. 6 loop

return t_a_r;

end quad_co;

-- TESTE DOS LIMITES DAS JUNTAS
function test_lim( t_a , delta_in : in vetor6  ) return vetor6 is

   ang_atual,ang_max,d_in : vetor6;
```

```
begin

for i in 1 .. nl loop
      d_in(i) := delta_in(i) * rad_deg;
      ang_atual(i) := t_a(i)*rad_deg;
      if ang_atual(i) > f_c_pos(i) then
            ang_max(i) :=  f_c_pos(i);
      elsif ang_atual(i) < f_c_neg(i) then
            ang_max(i) :=  f_c_neg(i);
      else
            ang_max(i) := ang_atual(i);
      end if;
end loop;

return ang_max;

end test_lim;

-- CALCULO DO NUMERO DE CONDIÇ¦O DA MATRIZ JACOBIANA
function num_cond( J , j_inv : in matrix6 ) return float is

   norm_j, norm_ji : vetor6;
   norj, norji, n_cond : float;

begin

norm_ji := (0.0,0.0,0.0,0.0,0.0,0.0);
norm_j := (0.0,0.0,0.0,0.0,0.0,0.0);
for i in 1 .. nl loop
      for k in 1 .. nc loop
            norm_j(i) := norm_j(i) + abs(J(i,k));
            norm_ji(i) := norm_ji(i) + abs(j_inv(i,k));
      end loop;
end loop;
norj := norm_j(1);
norji := norm_ji(1);
for k in 2 .. nl loop
      if norm_j(k) > norj then
            norj := norm_j(k);
      end if;
      if norm_ji(k) > norji then
            norji := norm_ji(k);
      end if;
end loop;
n_cond := norj * norji;

return n_cond;

end num_cond;

-- CALCULO DO VETOR ERRO PARA SISTEMAS MAL CONDICIONADOS
function v_erro( J,matriz_gg : in matrix6;delta_in,D : in vetor6 ) return vetor6 is

   armaz6,residuo,erro : vetor6;

begin

armaz6 := (0.0,0.0,0.0,0.0,0.0,0.0);
```

```
for i in 1 .. nl loop
      for k in 1 .. nc loop
            armaz6(i) := armaz6(i) + J(i,k)*delta_in(k);
      end loop;
      residuo(i) := D(i) - armaz6(i);
end loop;
erro := (0.0,0.0,0.0,0.0,0.0,0.0);
for i in 1 .. nl loop
      for k in 1 .. nc loop
            erro(i) := erro(i) + matriz_gg(i,k)*residuo(k);
      end loop;
end loop;

return erro;

end v_erro;

end funcoes;
```

ANEXO G

Evoluções angulares das juntas

As tabelas apresentadas a seguir ,mostram as evoluções angulares das juntas para os métodos de Gauss e Greville. Os valores angulares são apresentados em graus.

Tabela g.1: Evoluções angulares das juntas para a configuração 1 (800,0,933.1,21,58,90) para *m* igual a 20 divisões.

Iteração número	Junta1	Junta 2	Junta 3	Junta 4	Junta 5	Junta 6
1	0	90	-90	0	90	0
2	-0.25	89.87	-89.97	0.15	90.5	0.95
3	-0.57	89.74	-89.97	0.35	91.16	2.08
4	-0.94	89.6	-90	0.61	91.92	3.45
5	-1.38	89.45	-90.09	0.97	92.84	5.14
6	-1.93	89.28	-90.25	1.46	93.97	7.23
7	-2.61	89.09	-90.52	2.17	95.38	9.89
8	-3.46	88.85	-90.96	3.22	97.16	13.33
9	-4	88.74	-91.41	4.06	98.28	15.61
10	-4.92	88.43	-92.06	5.55	100.21	19.58
11	-5.57	88.24	-92.72	6.86	101.56	22.56
12	-6.03	88.08	-93.25	7.92	102.53	24.78
13	-7.01	87.6	-94.3	10.29	104.63	29.73
14	-7.74	87.22	-95.34	12.51	106.18	33.82
15	-8.28	86.87	-96.24	14.47	107.34	37.17
16	-8.69	86.57	-97	16.13	108.21	39.89
17	-8.99	86.31	-97.62	17.51	108.87	42.1
18	-9.85	85.41	-99.28	21.56	110.76	48.57
19	-10.44	84.59	-100.85	25.38	112.04	54.2
20	-10.82	83.82	-102.22	28.8	112.9	59.03
21	-11.07	83.13	-103.37	31.76	113.45	63.1
22	-11.24	82.53	-104.33	34.28	113.81	66.52
23	-11.34	82	-105.11	36.39	114.05	69.37
24	-11.4	81.55	-105.75	38.15	114.19	71.74
25	-11.44	81.18	-106.28	39.6	114.28	73.7
26	-11.56	79.56	-108.16	45.28	114.58	81.4
27	-11.45	78.18	-109.68	50	114.34	87.78
28	-11.23	77.02	-110.87	53.78	113.84	92.96
29	-10.98	76.09	-111.79	56.75	113.27	97.12
30	-10.72	75.34	-112.49	59.07	112.72	100.44
31	-10.49	74.75	-113.03	60.89	112.21	103.09
32	-10.3	74.27	-113.45	62.31	111.77	105.2
33	-9.6	72.58	-114.64	66.96	110.28	112.23
34	-8.92	71.41	-115.46	70.12	108.79	117.4
35	-8.34	70.6	-116.02	72.28	107.53	121.17
36	-7.87	70.04	-116.41	73.78	106.54	123.93
37	-7.52	69.63	-116.68	74.83	105.78	125.94
38	-6.56	68.46	-117.19	77.54	103.74	131.32
39	-5.85	67.82	-117.53	79.07	102.24	134.8
40	-5.36	67.44	-117.73	79.98	101.2	137.06
41	-4.39	66.62	-117.92	81.64	99.18	141.46
42	-3.8	66.27	-118.06	82.43	97.95	143.94
43	-3.46	66.09	-118.13	82.83	97.23	145.33
44	-2.75	65.61	-118.1	83.61	95.75	148.28
45	-2.4	65.46	-118.13	83.94	95.01	149.68
46	-1.79	65.1	-118.02	84.41	93.75	152.08
47	-1.56	65.02	-118.03	84.57	93.26	152.99
48	-1.1	64.77	-117.86	84.81	92.31	154.76

49	-0.65	64.53	-117.67	84.98	91.36	156.51
50	-0.28	64.34	-117.47	85.06	90.59	157.93
51	0	64.19	-117.25	85.07	90	159

Tabela g.2: Evoluções angulares das juntas para a configuração 1 (800,0,933.1,21,58,90) para m igual a 50 divisões.

Iteração número	Junta1	Junta 2	Junta 3	Junta 4	Junta 5	Junta 6
1	0	90	-90	0	90	0
2	-0.08	89.95	-89.99	0.06	90.16	0.38
3	-0.22	89.9	-89.98	0.13	90.44	0.79
4	-0.34	89.84	-89.98	0.2	90.69	1.23
5	-0.47	89.79	-89.98	0.29	90.96	1.71
6	-0.61	89.74	-89.99	0.38	91.24	2.23
7	-0.76	89.68	-90	0.49	91.56	2.8
8	-0.93	89.62	-90.03	0.61	91.9	3.42
9	-1.11	89.56	-90.06	0.76	92.27	4.1
10	-1.3	89.5	-90.11	0.92	92.68	4.85
11	-1.52	89.44	-90.17	1.11	93.13	5.67
12	-1.76	89.37	-90.24	1.33	93.62	6.59
13	-2.03	89.29	-90.34	1.59	94.17	7.62
14	-2.32	89.21	-90.46	1.9	94.78	8.76
15	-2.65	89.13	-90.62	2.26	95.45	10.05
16	-3.01	89.03	-90.81	2.71	96.21	11.51
17	-3.42	88.92	-91.05	3.24	97.05	13.17
18	-3.87	88.79	-91.34	3.89	98	15.06
19	-4.38	88.64	-91.72	4.7	99.07	17.24
20	-4.95	88.46	-92.19	5.7	100.27	19.76
21	-5.59	88.23	-92.77	6.95	101.61	22.7
22	-6.04	88.07	-93.28	7.98	102.57	24.89
23	-6.63	87.81	-93.94	9.41	103.81	27.83
24	-7.06	87.61	-94.51	10.6	104.72	30.12
25	-7.65	87.27	-95.29	12.34	105.98	33.39
26	-8.08	87.01	-95.98	13.84	106.91	36.01
27	-8.41	86.78	-96.55	15.08	107.61	38.11
28	-8.92	86.36	-97.43	17.14	108.71	41.53
29	-9.3	86.01	-98.21	18.92	109.53	44.35
30	-9.58	85.71	-98.87	20.43	110.15	46.66
31	-10.03	85.15	-99.91	22.98	111.13	50.54
32	-10.35	84.67	-100.82	25.21	111.84	53.8
33	-10.58	84.25	-101.59	27.1	112.35	56.5
34	-10.75	83.89	-102.23	28.7	112.72	58.75
35	-11.03	83.18	-103.33	31.63	113.37	62.86
36	-11.22	82.56	-104.28	34.14	113.77	66.32
37	-11.33	82.03	-105.07	36.27	114.02	69.2
38	-11.4	81.58	-105.72	38.04	114.18	71.6
39	-11.44	81.2	-106.25	39.52	114.27	73.58
40	-11.5	80.36	-107.27	42.53	114.43	77.65
41	-11.51	79.66	-108.11	45.03	114.43	81.03
42	-11.47	79.08	-108.8	47.09	114.35	83.81
43	-11.42	78.59	-109.36	48.77	114.24	86.09
44	-11.29	77.62	-110.33	51.91	113.97	90.39
45	-11.13	76.85	-111.11	54.43	113.62	93.87
46	-10.97	76.22	-111.73	56.43	113.25	96.68
47	-10.81	75.71	-112.21	58.02	112.9	98.94
48	-10.5	74.74	-113	60.86	112.23	103.03

49	-10.2	74.01	-113.62	63.02	111.58	106.25
50	-9.94	73.44	-114.09	64.68	110.99	108.77
51	-9.48	72.49	-114.76	67.27	110.01	112.81
52	-9.08	71.81	-115.27	69.15	109.14	115.87
53	-8.75	71.31	-115.63	70.53	108.42	118.2
54	-8.22	70.5	-116.11	72.58	107.28	121.76
55	-7.79	69.95	-116.47	74	106.36	124.37
56	-7.18	69.2	-116.87	75.82	105.06	127.88
57	-6.72	68.73	-117.15	77.02	104.08	130.33
58	-6.11	68.1	-117.43	78.46	102.78	133.46
59	-5.67	67.73	-117.63	79.35	101.87	135.54
60	-5.11	67.24	-117.79	80.4	100.69	138.16
61	-4.74	66.98	-117.92	81.02	99.91	139.83
62	-4.26	66.61	-118.01	81.76	98.9	141.95
63	-3.73	66.24	-118.08	82.49	97.79	144.22
64	-3.21	65.91	-118.12	83.12	96.7	146.4
65	-2.72	65.63	-118.13	83.63	95.69	148.37
66	-2.29	65.39	-118.11	84.02	94.79	150.09
67	-1.92	65.2	-118.07	84.31	94.02	151.56
68	-1.61	65.04	-118.02	84.53	93.36	152.79
69	-1.34	64.91	-117.96	84.69	92.8	153.84
70	-1.11	64.79	-117.89	84.81	92.32	154.74
71	-0.91	64.69	-117.82	84.9	91.9	155.51
72	-0.73	64.6	-117.74	84.96	91.53	156.19
73	-0.58	64.52	-117.67	85	91.21	156.79
74	-0.44	64.44	-117.58	85.03	90.92	157.32
75	-0.31	64.38	-117.5	85.05	90.65	157.8
76	-0.2	64.31	-117.42	85.06	90.42	158.24
77	-0.09	64.25	-117.34	85.06	90.2	158.64
78	0	64.19	-117.25	85.06	90	159

Tabela g.3: Evoluções angulares das juntas para a configuração 1 (800,0,933.1,21,58,90) para *m* igual a 70 divisões.

Iteração número	Junta1	Junta 2	Junta 3	Junta 4	Junta 5	Junta 6
1	0	90	-90	0	90	0
2	-0.06	89.96	-89.99	0.04	90.12	0.27
3	-0.15	89.93	-89.98	0.09	90.31	0.56
4	-0.24	89.89	-89.98	0.14	90.48	0.86
5	-0.32	89.85	-89.98	0.19	90.66	1.18
6	-0.42	89.81	-89.98	0.25	90.85	1.52
7	-0.51	89.77	-89.98	0.32	91.05	1.88
8	-0.62	89.73	-89.99	0.39	91.26	2.26
9	-0.73	89.69	-90	0.47	91.49	2.67
10	-0.84	89.65	-90.02	0.55	91.73	3.11
11	-0.97	89.61	-90.04	0.65	91.98	3.57
12	-1.1	89.57	-90.06	0.75	92.26	4.07
13	-1.24	89.53	-90.1	0.87	92.55	4.61
14	-1.39	89.48	-90.14	1	92.86	5.18
15	-1.56	89.43	-90.18	1.14	93.2	5.8
16	-1.73	89.38	-90.24	1.3	93.56	6.47
17	-1.92	89.33	-90.31	1.49	93.95	7.2
18	-2.12	89.28	-90.39	1.69	94.37	7.99
19	-2.34	89.22	-90.49	1.93	94.82	8.85
20	-2.58	89.16	-90.6	2.19	95.31	9.79
21	-2.84	89.09	-90.73	2.5	95.85	10.82
22	-3.12	89.01	-90.89	2.85	96.43	11.95
23	-3.42	88.93	-91.07	3.26	97.06	13.19
24	-3.75	88.84	-91.29	3.73	97.75	14.57
25	-4.11	88.73	-91.54	4.29	98.51	16.1
26	-4.51	88.61	-91.85	4.93	99.33	17.81
27	-4.94	88.48	-92.21	5.7	100.23	19.71
28	-5.4	88.31	-92.63	6.61	101.22	21.85
29	-5.91	88.12	-93.14	7.69	102.29	24.25
30	-6.46	87.89	-93.74	8.98	103.45	26.97
31	-6.86	87.71	-94.26	10.05	104.3	29.05
32	-7.35	87.45	-94.89	11.43	105.33	31.69
33	-7.71	87.26	-95.43	12.58	106.1	33.76
34	-8.17	86.95	-96.12	14.16	107.09	36.54
35	-8.51	86.71	-96.73	15.48	107.83	38.77
36	-8.96	86.33	-97.53	17.34	108.8	41.83
37	-9.29	86.02	-98.22	18.92	109.52	44.33
38	-9.54	85.76	-98.79	20.24	110.06	46.36
39	-9.9	85.32	-99.62	22.26	110.85	49.44
40	-10.16	84.95	-100.34	23.98	111.43	51.98
41	-10.36	84.64	-100.94	25.43	111.86	54.08
42	-10.65	84.11	-101.83	27.71	112.5	57.36
43	-10.85	83.66	-102.59	29.65	112.94	60.1
44	-10.99	83.28	-103.23	31.29	113.26	62.36
45	-11.19	82.64	-104.17	33.85	113.71	65.9
46	-11.31	82.1	-104.98	36.02	113.99	68.86
47	-11.39	81.64	-105.64	37.83	114.16	71.31
48	-11.44	81.24	-106.18	39.34	114.26	73.34
49	-11.49	80.55	-107.05	41.87	114.4	76.76

50	-11.5	79.97	-107.77	43.97	114.42	79.59
51	-11.49	79.49	-108.36	45.7	114.39	81.93
52	-11.44	78.69	-109.22	48.4	114.28	85.59
53	-11.35	78.03	-109.93	50.61	114.1	88.59
54	-11.26	77.5	-110.5	52.39	113.88	91.04
55	-11.08	76.65	-111.3	55.05	113.51	94.74
56	-10.9	75.98	-111.94	57.16	113.1	97.72
57	-10.73	75.44	-112.45	58.83	112.72	100.11
58	-10.44	74.61	-113.13	61.26	112.1	103.63
59	-10.18	73.98	-113.66	63.12	111.53	106.41
60	-9.95	73.49	-114.07	64.56	111.02	108.59
61	-9.59	72.74	-114.61	66.61	110.25	111.79
62	-9.29	72.19	-115.02	68.15	109.58	114.25
63	-8.85	71.44	-115.52	70.13	108.64	117.52
64	-8.49	70.91	-115.89	71.55	107.86	119.96
65	-8.01	70.23	-116.29	73.29	106.82	123.06
66	-7.62	69.76	-116.6	74.49	106.01	125.31
67	-7.13	69.18	-116.91	75.92	104.96	128.09
68	-6.77	68.8	-117.14	76.88	104.18	130.05
69	-6.3	68.31	-117.36	78	103.19	132.47
70	-5.77	67.81	-117.57	79.15	102.07	135.07
71	-5.4	67.51	-117.73	79.86	101.3	136.8
72	-4.97	67.15	-117.85	80.63	100.39	138.79
73	-4.51	66.8	-117.96	81.39	99.42	140.86
74	-4.05	66.47	-118.05	82.07	98.45	142.86
75	-3.61	66.17	-118.1	82.65	97.53	144.73
76	-3.2	65.92	-118.14	83.13	96.69	146.42
77	-2.83	65.71	-118.15	83.52	95.92	147.92
78	-2.51	65.53	-118.15	83.83	95.24	149.23
79	-2.22	65.37	-118.13	84.08	94.64	150.38
80	-1.96	65.24	-118.1	84.29	94.11	151.39
81	-1.74	65.12	-118.07	84.45	93.63	152.28
82	-1.54	65.02	-118.03	84.58	93.21	153.07
83	-1.36	64.92	-117.98	84.69	92.84	153.78
84	-1.19	64.84	-117.94	84.77	92.49	154.41
85	-1.04	64.77	-117.88	84.84	92.18	154.98
86	-0.91	64.7	-117.83	84.9	91.9	155.51
87	-0.78	64.63	-117.78	84.94	91.64	155.99
88	-0.67	64.57	-117.72	84.98	91.4	156.43
89	-0.56	64.52	-117.67	85.01	91.18	156.83
90	-0.47	64.46	-117.61	85.03	90.98	157.21
91	-0.37	64.41	-117.55	85.05	90.79	157.56
92	-0.29	64.37	-117.49	85.06	90.61	157.89
93	-0.21	64.32	-117.43	85.06	90.44	158.19
94	-0.14	64.28	-117.37	85.06	90.28	158.48
95	-0.07	64.23	-117.31	85.06	90.14	158.75
96	0	64.19	-117.25	85.06	90	159