UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

Este exemplar corresponde à redação final da tese defendida por Ana Clelia Ferreira e aprovada pela Comissão Julgadora em 31 / 08 / 90.

Orientador

OTIMIZAÇÃO NA GERAÇÃO AUTOMÁTICA DOS MODELOS GEOMÉTRICOS E NA GERAÇÃO ASSISTIDA DE MODELOS PARA A COORDENAÇÃO DE MOVIMENTOS DE ROBÔS

por: Ana Clelia Ferreira
Orientador: Edson de Paula Ferreira

/90

Tese de Mestrado apresentada à
Faculdade de Engenharia Elétrica da
Universidade Estadual de Campinas

31 de agosto de 1990

Este trabalho contou com o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e do Programa Argentino-Brasileiro de Pesquisas e Estudos Avançados em Informática (PABI); contou com apoio técnico e logístico do Centro Tecnológico para Informática (CTI), PETROBRAS/Refinaria de Paulínia (REPLAN), e Universidade Federal do Espírito Santo (UFES).

Dedico este trabalho aos meus pais
Anníbal e Maria Brum

AGRADECIMENTOS

No curso deste trabalho, contamos com a participação e apoio de muitas pessoas que contribuiram para que o concluíssemos com sucesso. Dentre estas queremos aqui registrar o agradecimento:

Ao prof. EDSON DE PAULA FERREIRA, meu orientador de tese, pelas inúmeras oportunidades de trabalho propiciadas, pela crítica constante, objetividade, otimismo, incentivo e disponibilidade durante todo este tempo;

Aos profs. CLÉSIO LUIS TOZZI e ÁLVARO GERALDO BADAN PALHARES, pela participação nesta banca e pelas críticas à dissertação;

Agradecimento especial ao amigo e colega de equipe JOSÉ PAULO DE ANDRADE FILHO, pela qualidade e quantidade de idéias, críticas e sugestões ao longo de todo o trabalho;

Aos companheiros de equipe JORGE, JOSUÉ, RALPH, OTHON e MARCUS, pelas sugestões, críticas, e cobranças;

Ao Chefe do Setor de Automação Industrial da Refinaria de Paulínia, ITAMAR JOSÉ MACHADO, pela lucidez, apoio e infraestrutura proporcionada;

Ao prof. LÉO PINI MAGALHÃES e a MARIA PAULINA JULIANE, pelo apoio à realização dos Laboratórios de Automação e Robótica das III^{a} e IV^{a} Escola Brasileira-Argentina de Informática (EBAI);

Aos alunos e colegas dos LABORATÓRIOS DE AUTOMAÇÃO INDUSTRIAL E ROBÓTICA das III^{a} e IV^{a} EBAI que, com suas críticas e sugestões, auxiliaram no aprimoramento das ferramentas e métodos apresentados neste trabalho;

Ao professor JOSÉ ARMÍNIO FERREIRA, pela orientação na aplicação de ferramentas matemáticas;

RESUMO

Este trabalho constitui uma contribuição para a redução do tempo necessário ao desenvolvimento de modelos de robôs e para a otimização da expressão analítica destes, minimizando o seu tempo de cálculo. Os modelos que se tem em vista são o modelo geométrico, o geométrico inverso e alguns desdobramentos da geometria na modelagem dinâmica.

A finalidade principal da utilização do modelo geométrico é o controle de trajetórias e obtenção do modelo dinâmico. A finalidade principal da utilização do modelo inverso é a coordenação de movimentos.

Estes modelos são de grande complexidade, e sua obtenção manual, além de demorada, é extremamente árida e bastante sujeita a erros. Por este motivo, implementamos um sistema para geração simbólica automática de modelos geométricos e das equações de auxílio à inversão do modelo.

Neste trabalho é proposto um método de obtenção para o modelo geométrico ótimo, onde são tratadas de forma separada as informações de rotação e translação; é feito um estudo exaustivo das situações onde aparecem simplificações trigonométricas nas expressões do modelo; e é proposta também uma extensão do método analítico descrito em [PAUL 81] para de inversão do modelo geométrico.

PALAVRAS-CHAVE: Robótica, Modelagem Geométrica, Coordenação de Movimentos, Otimização de Modelos.

ABSTRACT

This work is a contribution to the efforts for reducing the robot models development time, as well as optimizing its analytical expressions, so minimizing its calculation time. The models we're looking to are the geometric model, the inverse geometric model and some geometric results for dynamic modelling.

Our main purpose is to enable the generation of models suited for the use in trajectory control and dynamic modelling, in the case of the geometric models; and for the use in movement coordination, in the case of inverted geometric models.

These are quite complex models and their manual derivation is tedious, costly (time-consumming) and often error-prone. So it was implemented a system for automatically generating symbolic geometric models, and automatically generating symbolic auxiliary expressions for geometric models inversion.

In this work it is proposed a method for obtaining optimal geometric models, which treats in a separate way the rotation and translation informations; it's also proposed an extension for the analytical inversion method described by [PAUL 81], and the situations where the model expressions have trigonometric simplifications are exaustively studied.

KEYWORDS: Robotics, Geometric Modelling, Movement Coordination, Model Optimization.

CONTEÚDO

CAPÍTULO 1

INTRODUÇÃO GERAL

1.1. HISTÓRICO

Este trabalho faz parte de um conjunto de esforços para construção de ferramentas para modelagem e simulação de robôs cuja motivação básica é a "queima" de etapas no desenvolvimento de trabalhos na área. Com relação à obtenção de modelos analíticos, o desenvolvimento manual destes consistia numa etapa árdua, com resultados de baixa confiabilidade, daí a necessidade de ferramentas para geração rápida de modelos confiáveis.

Esta motivação ganhou espaço a partir do II^{o} Encontro Argentino - Brasileiro de Pesquisa e Estudos Avançados em Informática, realizado durante a II^{a} EBAI (Escola Brasileiro-Argentina de Informática) em fevereiro de 1987, dentro da elaboração do projeto SIGMA-I, onde foram propostos o desenvolvimento de uma ferramenta para geração de modelos e simulação de robôs e a realização de um Laboratório de Automação Industrial e Robótica, com utilização desta ferramenta, no espaço da III^{a} EBAI a ser realizada um ano depois.

Uma primeira versão desta ferramenta - GMSIR - foi desenvolvida no ano de 1987 por pesquisadores do Centro Tecnológico para Informática e alunos de mestrado da UNICAMP ([FERREIRA 88] e [SILVA 88]), sendo o Laboratório de Automação Industrial e Robótica coordenado pelo professor Edson de Paula Ferreira, da Universidade Federal do Espírito Santo.

Foram premissas de desenvolvimento a possibilidade de utilização do GMSIR de forma didática e versátil na formação de pessoal, além da utilização por pesquisadores da área, bem como a sua implementação em máquinas de pequeno porte.

De resultado prático dos Laboratórios de Automação Industrial e Robótica das III^{a} e IV^{a} EBAI, temos:

- Difusão da ferramenta nas universidades brasileiras e argentinas

- Evolução da ferramenta para o GMSIR versão 2,

- Desenvolvimento de sistemas auxiliares em validação de modelos, simulação gráfica e outros, no âmbito dos Laboratórios.

Nosso trabalho se insere no âmbito do desenvolvimento de métodos e ferramentas para a modelagem geométrica e geométrica inversa, sendo que o passo inicial foi a implementação destes aspectos no GMROB (Gerador de Modelos de Robôs, um dos módulos do GMSIR), prosseguindo com a ênfase no desenvolvimento de métodos de obtenção de modelos ótimos, e finalizando com proposições para extensão do próprio GMROB para contemplar estes últimos aspectos.

1.2. CONTEXTO E OBJETIVOS

Nossa proposta inicial, dentro do universo da geração de modelos era de obtenção de uma ferramenta para geração completa do modelo inverso analítico, automatizando procedimentos até onde fosse possível, assistindo o usuário no restante.

Para construção de uma ferramenta com este propósito, seria necessário o levantamento do conhecimento (em forma de heurística, estudos de casos, estudos das simplificações trigonométricas, desenvolvimento de métodos, etc), etapa completamente inexplorada e de grau de complexidade com dimensão incompatível com um único trabalho.

Nossa contribuição nesta área se deu nos seguintes aspectos:

- Desenvolvimento de uma ferramenta automatizando a etapa árdua de preparação das expressões para a inversão, utilizando uma extensão do método descrito em [PAUL 81];

- Estudo analítico exaustivo das simplificações trigonométricas constantes nestas expressões, estabelecendo com precisão a ocorrência destas;

- Estudo sobre validação do modelo inverso de uma robô, constituindo um embrião de ferramenta genérica de validação.

Ao longo do desenvolvimento do nosso trabalho, uma outra questão ganhou relevância: a otimização dos modelos geométricos para cálculo em tempo real e para obtenção do modelo dinâmico, além da otimização em relação ao aspecto de melhor visualização das expressões para auxílio à inversão do modelo.

Nesta área se situa a melhor parte da nossa contribuição original:

- Estabelecimento de critérios para a otimização do modelo geométrico;

- Desenvolvimento de algoritmo para geração das seqüências de multiplicação das matrizes elementares para escolha do melhor modelo geométrico;

- Desenvolvimento de um método de obtenção das matrizes intermediárias, sendo os termos de rotação e translação gerados independentemente;

- Determinação das situações onde aparecem simplificações trigonométricas e sua resolução.

1.3. ESTRUTURA DO SISTEMA PARA GERAÇÃO AUTOMÁTICA DE MODELOS

Como o desenvolvimento do GMROB fez parte da primeira etapa do nosso trabalho, já tendo sido descrito em publicações anteriores, consideramos adequado apresentar sua estrutura no capítulo introdutório.

No eixo de desenvolvimento do GMROB, este trabalho contempla a geração automática de modelos geométricos e auxílio à modelagem inversa, sendo que a geração automática de modelos dinâmicos é um dos objetivos da dissertação [SILVA 90].

O sistema para geração de modelos é constituído funcionalmente dos seguintes módulos:

- Gerador Automático de Modelo Geométrico Direto;

- Ferramenta de Auxílio à Obtenção do Modelo Geométrico Inverso;

- Gerador Automático de Modelo Dinâmico;

A estrutura do sistema está implementada da seguinte forma:

- Supervisão: coordena o uso da Inicialização e Modelagem.

- Inicialização: permite que sejam introduzidos parâmetros geométricos e dinâmicos de robôs, ou que sejam manuseados os parâmetros que já tenham sido introduzidos anteriormente. As opções existentes na inicialização são: criar ou apagar um arquivo, modificar o seu conteúdo, visualizar o conteúdo de um arquivo ou o conteúdo do diretório de arquivos. No caso de se tentar criar um arquivo cujo nome já exista, serão oferecidas as seguintes opções: modificar o nome ou conteúdo do arquivo ou simplesmente utilizá-lo como dado. Quando se deseja apagar um arquivo de inicialização, o sistema auxilia o usuário no sentido de apagar todos os arquivos que são resultantes da geração de modelos e que usam este arquivo de inicialização. A destruição dos arquivos é feita um por um, mediante autorização do usuário, de forma a apagar apenas os arquivos desejados.

- Modelagem: permite a geração de modelos geométrico direto e dinâmico em sua forma simbólica a partir dos parâmetros geométricos e dinâmicos do robô. Gera expressões de auxílio à obtenção do modelo geométrico inverso.

A figura (1.1.) apresenta um esquema dos módulos descritos acima.

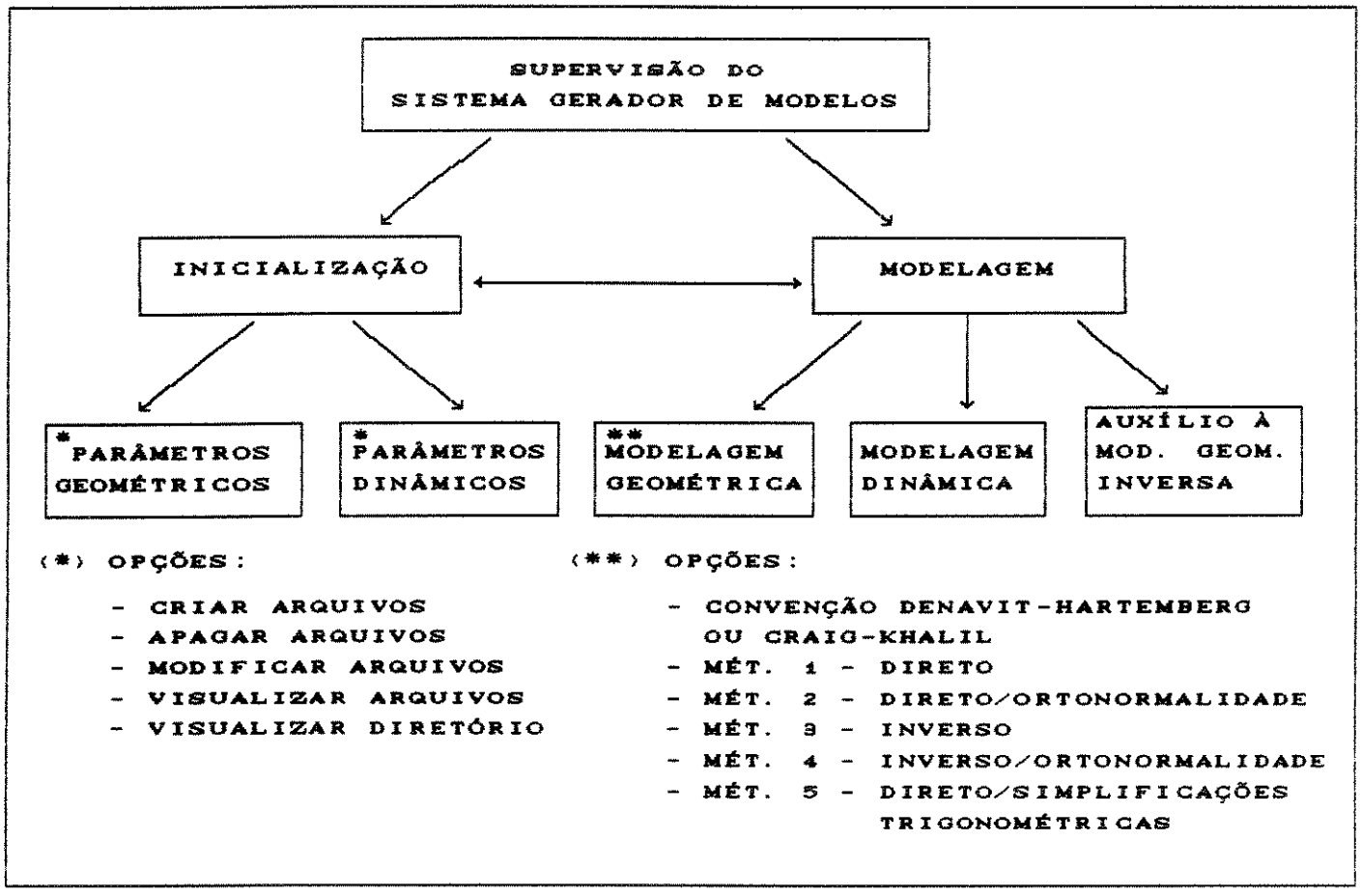


Figura (1.1.): Estrutura básica do Sistema Gerador de Modelos

No item 2.3., é feita uma descrição complementar do módulo de geração do modelo geométrico do GMROB (SIS_MGD), sendo apresentados os cinco métodos de montagem do modelo que aparecem na figura (1.1.).

Os métodos de montagem do modelo não se restrigem aos utilizados no SIS_MGD, sendo este assunto trabalhado pormenorizadamente no capítulo dois, que trata exatamente dos objetivo e técnicas para otimização do modelo geométrico.

Como motivação para leitura do capítulo dois, vamos apresentar na tabela (1.1.), a grande variação do número de operações em função do uso de diferentes métodos de montagem do modelo (para ilustração são mostrados apenas os números de operações correspondentes aos cinco métodos implementados no SIS_MGD, com ou sem utilização de variáveis auxiliares, tendo sidos usados os parâmetros do robô Puma 560 para

geração dos modelos).

MÉTODO	VAR. AUX.	NÚMERO DE OPERAÇÕES				
D-H	e/n	*	+	-	TOTAL	SEN/COS
1	e	70	25	11	106	12
1	n	205	40	37	282	12
2	e	71	23	13	107	12
2	n	1299	306	175	1780	12
3	e	70	20	14	104	12
3	n	131	41	36	208	12
4	e	62	19	12	93	12
4	n	253	66	79	253	12
5	e	53	21	07	81	14

Tabela 1.1 - Número de operações versus Método Implementado

1.4. ESTRUTURA DO TRABALHO

Além da introdução, nossa dissertação tem mais três capítulos, constando, no segundo capítulo, os resultados pertinentes à otimização do modelo direto; no terceiro capítulo, os resultados pertinentes à inversão do modelo; no quarto, a síntese dos principais resultados e apresentação de algumas perspectivas de evolução.

O segundo capítulo está dividido em seis itens:

- O item 2.1. introduz o leitor em alguns conceitos e ferramentas básicas para o entendimento do restante do capítulo, que são as transformações homogêneas, as matrizes de transformação elementares e o modelo geométrico direto.

- No item 2.2. são apresentados os diversos critérios de

otimização para o modelo geométrico com vistas a sua utilização em controle de trajetórias, obtenção do modelo dinâmico e auxílio à inversão do modelo geométrico. São levantadas as possíveis formas de obtenção do modelo com vistas a otimização, em relação a: seqüência de multiplicação das matrizes elementares, ao uso de equações auxiliares, simplificações trigonométricas e o uso de propriedades de bases ortonormais das expressões de rotação das matrizes de transformação.

- No item 2.3. é apresentada uma ferramenta (SIS_MGD DO GMROB) que gera automaticamente o modelo geométrico de vinte maneiras diferentes, executando os primeiros passos em direção à otimização do modelo. Os dois itens subsequentes fornecem subsídios para a proposição de uma metodologia mais genérica que a contida no GMROB para geração de modelos ótimos.

- No item 2.4. é apresentado um método de obtenção para as matrizes intermediárias, desenvolvido com dois objetivos básicos: facilitar a identificação analítica das simplificações trigonométricas e permitir o estudo da contraposição entre estas e as equações auxiliares.

- No item 2.5. é identificado um elenco de situações das expressões do modelo onde aparecem simplificações trigonométricas, e são deduzidas as expressões simplificadas.

- No item 2.6. são sintetizados os passos para obtenção dos modelos ótimos, dentro dos critérios estabelecidos no item 2.2., e discutida a aplicação destes modelos na otimização do modelo dinâmico.

O terceiro capítulo também está dividido em seis itens:

- O item 3.1. é utilizado para introduzir o leitor no conceito de coordenação de movimentos e na complexidade da inversão do modelo geométrico.

- No item 3.2. são detalhadas as dificuldades da inversão do modelo em relação a: existência e número de soluções, generalidade dos métodos, singularidades, redundâncias, escolha da representação do modelo direto mais apropriada à inversão (quando a inversão é analítica), e cálculo do modelo em tempo real (quando a inversão é numérica).

- No item 3.3. é apresentada uma breve comparação entre métodos numéricos e analíticos para obtenção do modelo inverso, e apresentado o método descrito em [PAUL 81], bem como a justificativa de sua adoção neste trabalho.

- No item 3.4. é apresentada uma ferramenta (SIS_INV do GMROB) que gera automaticamente as expressões de auxílio à inversão segundo o método de [PAUL 81], inserindo uma extensão ao método, para proporcionar em alguns casos melhor visualização das variáveis nas expressões.

- No item 3.5. é apresentada uma extensão do método de obtenção das matrizes intermediárias descrito no item 2.4., que constitui também uma extensão do método apresentado em [PAUL 81] para inversão do modelo, que tem por finalidade ampliar o elenco de maneiras de apresentação do modelo geométrico para estudo sistemático da correspondência entre a estrutura do robô e os elementos das matrizes mais adequados à explicitação das coordenadas generalizadas.

- O item 3.6. tem por objetivo mostrar a justificativa da necessidade de validação dos modelos inversos (por simulação) e a apresentação dos esforços feitos nesse sentido pela equipe do Laboratório de Automação e Robótica da III^{a} EBAI.

CAPÍTULO 2

OTIMIZAÇÃO DO MODELO GEOMÉTRICO DE ROBÔS MANIPULADORES

2.1. INTRODUÇÃO

Neste capítulo trataremos da otimização do modelo geométrico para uso em controle de trajetória e da otimização dos modelos geométricos intermediários para uso na obtenção do modelo dinâmico.

Com este objetivo, introduziremos neste item os conceitos básicos necessários ao entendimento do restante do capítulo; no item 2.2. apresentaremos uma discussão sobre os critérios de otimização aplicáveis; no item 2.3. faremos uma breve descrição da ferramenta implementada para geração automática de modelos geométricos simbólicos e utilizada para testes nos desenvolvimentos teóricos apresentados nos itens seguintes; no item 2.4. descreveremos um novo método para geração das matrizes intermediárias, visando sua simplificação; no item 2.5. trataremos da redução do modelo por simplificações trigonométricas e finalmente no item 2.6. trataremos do cálculo dos modelos geométricos ótimos aplicado à obtenção do modelo dinâmico e proporemos a construção de uma ferramenta de geração de modelos mais adequada a otimização.

2.1.1. CONCEITO DE MODELO GEOMÉTRICO DE ROBÔ MANIPULADOR

Estamos neste trabalho tratando de robôs manipuladores (RM) cuja estrutura mecânica seja constituída por um conjunto de elementos (considerados) rígidos, denominados corpos, reunidos em cadeia através de elementos que permitam um movimento relativo entre os corpos, denominados ligações ou juntas.

Existem várias ligações possíveis entre dois corpos, porém é suficiente considerar somente os tipos de ligações que permitem rotação em torno de um eixo (R) ou a que permite translação ao longo de um eixo (P) [FERREIRA 87].

No caso de um RM, chama-se modelo geométrico as expressões matemáticas da posição e orientação do órgão terminal ou mão-ferramenta do RM descritos em relação a um sistema de coordenadas fixo em sua base, em função do movimento de cada elemento de sua

cadeia cinemática, rotação nos casos de ligações rotacionais, translação no caso de ligação prismática.

A partir daqui, às coordenadas que descrevem a posição e orientação do órgão terminal do RM denominaremos coordenadas operacionais e aos movimentos de cada elemento de sua cadeia cinemática denominaremos coordenadas generalizadas, conforme uso corrente na literatura.

As expressões matemáticas do modelo geométrico podem se apresentar de várias formas, de acordo com a ferramenta matemática adotada na sua obtenção, sistemas de coordenadas utilizados, posições de eixos arbitradas, parâmetros escolhidos e outras opções que serão ainda objeto de análise. Porém, neste ítem nos deteremos nos conceitos básicos para representação do modelo por produtos de matrizes de transformação (ou passagem) homogêneas, posicionamento dos eixos e escolha dos parâmetros segundo a convenção de Denavit-Hartemberg [DENAVIT 55], que foi utilizada na maior parte dos trabalhos que deram origem a esta dissertação, escolhido de acordo com critérios de simplicidade de representação, facilidade de manipulação simbólica, facilidade de sistematização etc.

2.1.2. A TRANSFORMAÇÃO HOMOGÊNEA E O MODELO GEOMÉTRICO

Além do Modelo Geométrico, que relaciona somente a posição e orientação do órgão terminal com as rotações e deslocamento das ligações, é necessário também conhecer-se as relações entre os diversos elementos de sua cadeia cinemática e entre a cadeia e os objetos do ambiente.

A transformação homogênea é uma ferramenta matemática que permite a descrição destas relações pela forma matricial, facilitando a obtenção das equações cinemáticas do manipulador, com a propriedade que qualquer composição de transformações é obtida pela multiplicação das matrizes de transformação homogêneas correspondentes.

Em sendo utilizadas transformações homogêneas podemos então redefinir o Modelo Geométrico de um robô como sendo a matriz que

relaciona o sistema de coordenadas solidário ao órgão terminal com o sistema de coordenadas solidário à base do robô, obtida pelo produto consecutivo das chamadas matrizes de transformação homogêneas elementares, matrizes que relacionam o sistema de coordenadas de um elemento com o sistema do elemento imediatamente anterior. Observamos que estas últimas matrizes cada uma é função de uma única coordenada de ligação.

Outra definição importante que será muito utilizada ao longo desta dissertação é a de Modelo Geométrico Intermediário, representado pelas matrizes de transformação homogêneas intermediárias, obtidas de multiplicação das elementares consecutivas entre os sistemas de coordenadas de dois elementos quaisquer da cadeia cinemática. Para ressaltar a diferença da matriz de transformação intermediária chamamos de matriz de transformação final a representação do Modelo Geométrico.

Para compreensão das transformações homogêneas vamos conceituar coordenadas homogêneas e derivar as transformações de rotação e translação.

2.1.3. COORDENADAS HOMOGÊNEAS, TRANSFORMADAS DE ROTAÇÃO E TRANSLAÇÃO.

A representação de pontos em um espaço n-dimensional através de coordenadas homogêneas é uma entidade espacial de n+1 dimensões, sendo que esta dimensão adicional pode ser vista como um fator de escala.

Seja um vetor $\vec{p} = p_x\vec{i} + p_y\vec{j} + p_z\vec{k}$, onde $\vec{i}$, $\vec{j}$ e $\vec{k}$ são vetores unitários nas direções "x", "y" e "z" respectivamente.

Ou em notação matricial $p = [p_x\ p_y\ p_z]^t$, onde "t" denota transposição

A sua representação em coordenadas homogêneas é uma matriz coluna do tipo:

$$P_h = [\ x\ y\ z\ e]^t \qquad (2.1.)$$

onde $p_x = x/e$, $p_y = y/e$, $p_z = z/e$, e = fator de escala.

É importante ressaltar a equivalência entre os vetores $\vec{p}$ e "p_h", que são apenas representações diferentes para o mesmo vetor. Também observamos que a representação de um vetor em coordenadas homogêneas não é única, pois o fator de escala pode ser qualquer (diferente de zero).

Há três observações importantes a fazer neste ponto:

1) Na representação de um vetor nulo o fator de escala pode ser qualquer fator não nulo:

$$p_h = [\ 0\ \ 0\ \ 0\ \ e]^t \qquad e \neq 0 \tag{2.2.}$$

2) Uma direção pode ser representada por um vetor do tipo:

$$p_h = [\ x\ 0\ 0\ 0\ ]^t \text{ (representa a direção 'x')}. \tag{2.3.}$$

3) Não se define o vetor:

$$p_h = [\ 0\ 0\ 0\ 0\ ]^t \tag{2.4.}$$

Embora a representação através de coordenadas homogêneas seja interessante por permitir representação matricial, não interessa na representação geométrica de robôs introduzir nenhum fator de escala. Assim, se for escolhido um fator de escala unitário, as próprias coordenadas físicas do vetor poderão ser consideradas como coordenadas homogêneas.

Este tipo de representação de pontos em um espaço Euclidiano tridimensional é útil na obtenção de matrizes de transformação que incluam transformações de rotação, translação, escalamento e perspectiva.

Para o nosso caso, simplificaremos a notação de "p_h" para "p", usando as coordenadas homogêneas com $e = 1$, $p = [\ p_x\ p_y\ p_z\ 1\ ]^t$.

2.1.3.1. TRANSFORMAÇÕES DE ROTAÇÃO

Considere um vetor "p" definido em um espaço Euclidiano tridimensional através de um sistema de coordenadas dito de referência.

Suponha agora que este vetor efetue uma rotação de um ângulo "e" ao redor do eixo 'z' do sistema (Figura 2.1.).

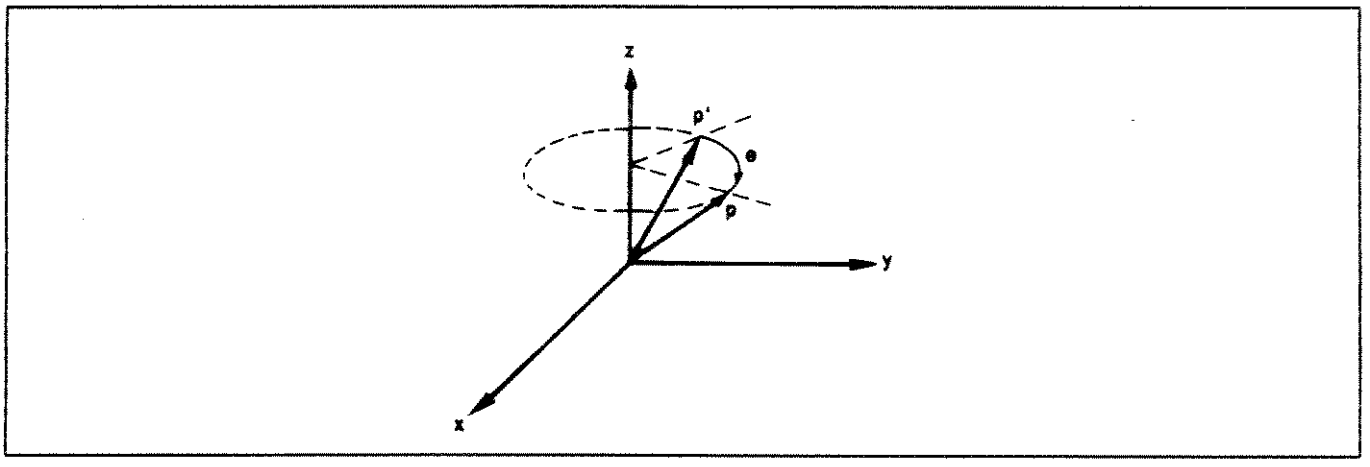


Figura (2.1.): Rotação de um Vetor "p"

A matriz 3x3 que permite determinar as novas coordenadas do vetor $p' = p_r$ em função das anteriores é uma matriz que é chamada de matriz de rotação. Ela opera (produto matricial) sobre as coordenadas antigas gerando as novas. Assim, chamando de "R" a matriz de rotação:

$$p_r = R\ p \qquad (2.5.)$$

Supondo agora que o vetor "p" represente um ponto fixo de um corpo tridimensional, podemos definir dois sistemas de coordenadas com origens coincidentes:

1) Um sistema fixo em relação ao espaço tridimensional, chamado de sistema de referência (Po), com eixos "x", "y", "z";

2) Um sistema fixo em relação ao vetor "p", e portanto em relação ao corpo, chamado de sistema do corpo (Pi), com eixos "u", "v", "w".

Nesse caso a rotação do vetor "p" corresponde à rotação do sistema "Pi" em relação ao sistema "Po", e a matriz de rotação "R" pode ser interpretada como a que descreve a rotação de um sistema em relação ao outro. Esta segunda interpretação é mais conveniente no caso de cadeia cinemáticas, pois elas são constituídas de diversos corpos aos quais são atribuidos sistemas de coordenadas fixos em relação a cada corpo. Assim sendo, a movimentação de um corpo em relação ao outro pode ser vista simplesmente como o movimento de um sistema de coordenadas em relação a outro.

As matrizes de rotação que expressam as rotações básicas são:

1) Rotação de um ângulo "θ" em torno do eixo "x":

$$Rot(x, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\text{sen}(\theta) \\ 0 & \text{sen}(\theta) & \cos(\theta) \end{bmatrix} \qquad (2.6.)$$

2) Rotação de um ângulo "θ" em torno do eixo "y":

$$Rot(y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \text{sen}(\theta) \\ 0 & 1 & 0 \\ -\text{sen}(\theta) & 0 & \cos(\theta) \end{bmatrix} \qquad (2.7.)$$

3) Rotação de um ângulo "θ" em torno do eixo "z":

$$Rot(z, \theta) = \begin{bmatrix} \cos(\theta) & -\text{sen}(\theta) & 0 \\ \text{sen}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.8.)$$

No caso de aplicação de sucessivas transformações de rotação à um sistema, o efeito resultante pode ser obtido através do produto sucessivo das matrizes de rotação elementares, na ordem efetiva em que se sucederam as rotações. Assim sendo, a matriz de rotação que expressa rotações sucessivas ao redor dos eixos "y", "z" e "x" será:

$$R = Rot(y, \theta_1)\, Rot(z, \theta_2)\, Rot(x, \theta_3) \qquad (2.9.)$$

onde $Rot(y, \theta_1)$ significa rotação de "θ_1" em torno de "y".

2.1.3.2. TRANSFORMAÇÃO DE TRANSLAÇÃO

Dado um vetor:

$$p = [\ p_x\ p_y\ p_z\ ]^t, \qquad (2.10.)$$

uma transformação de translação corresponde a efetuar deslocamentos em uma ou mais das direções de seus eixos. Assim uma translação de "a" na direção "x", de "b" na direção "y" e de "c" na direção "z" gera o vetor resultante "p_t" (translação de "p"; não confundir com a notação de transposição):

$$p_t = [\ p_x+a \quad p_y+b \quad p_z+c]^t \qquad (2.11.)$$

Observando-se este vetor fica evidente que uma transformação de translação corresponde a somar um vetor de deslocamento do tipo:

$$d = [\ a\ b\ c\ ]^t \qquad (2.12.)$$

E portanto:

$$p_t = p + d \qquad (2.13.)$$

2.1.4. MATRIZES DE TRANSFORMAÇÃO HOMOGÊNEAS

Para representar de forma conjunta as transformações de translação e de rotação deve-se observar que:

$$p_r = R.p \qquad e \qquad p_t = p_r + d \qquad (2.14.)$$

portanto:

$$p_{tr} = R.p + d \qquad (2.15.)$$

A representação desta transformação conjunta como uma matriz é possível caso os vetores sejam representados através de coordenadas homogêneas e "R" e "d" sejam elementos de partição de uma matriz, da seguinte forma:

$$\left|\begin{array}{c} P_{tr} \\ \hline 1 \end{array}\right| = T \cdot \left|\begin{array}{c} p \\ \hline 1 \end{array}\right| = \left|\begin{array}{c|c} R & d \\ \hline o & 1 \end{array}\right| \left|\begin{array}{c} p \\ \hline 1 \end{array}\right| = \left|\begin{array}{c} R.p + d \\ \hline 1 \end{array}\right| \quad (2.16.)$$

Onde $\left|\begin{array}{c} P_{tr} \\ \hline 1 \end{array}\right|$ e $\left|\begin{array}{c} p \\ \hline 1 \end{array}\right|$ são os vetores "p_{tr}" e "p" representados em coordenadas homogêneas, e $T = \left|\begin{array}{c|c} R\ (3x3) & d(3x1) \\ \hline 0\ (1x3) & 1(1x1) \end{array}\right|$ é a matriz de transformação homogênea, $0 = [0\ 0\ 0]$ é um vetor-linha nulo e 1 é o fator de escala unitário.

Na verdade a forma mais geral da matriz de transformação homogênea é a seguinte:

$$T = \left|\begin{array}{c|c} \begin{array}{c}\text{Matriz de rotação}\\ (3x3)\end{array} & \begin{array}{c}\text{Vetor de transl.}\\ \text{fator de escala}\\ (1x1)\end{array} \\ \hline \begin{array}{c}\text{Transf. Perspectiva}\\ (1x3)\end{array} & \end{array}\right| \quad (2.17.)$$

Embora úteis para outras aplicações, as transformações de perspectiva e de escalamento não nos são convenientes. Para cancelar o seu efeito e preservar a representação homogênea é que fazemos nulo o vetor das transformações de perspectivas e, como já foi dito, unitário o fator de escala.

A matriz genérica de transformação homogênea fica então da forma:

$$T = \begin{bmatrix} nx & ox & ax & px \\ ny & oy & ay & py \\ nz & oz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.18.)$$

De acordo com esta representação as matrizes que representam a translação e as rotações básicas são:

Rotação de um ângulo "θ" em torno do eixo "x":

$$\text{Rot}(x, \theta) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\text{sen}(\theta) & 0 \\ 0 & \text{sen}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.19.)$$

Rotação de um ângulo "θ" em torno do eixo "y":

$$\text{Rot}(y, \theta) = \begin{vmatrix} \cos(\theta) & 0 & \text{sen}(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.20.)$$

Rotação de um ângulo "θ" em torno do eixo "z":

$$\text{Rot}(z, \theta) = \begin{vmatrix} \cos(\theta) & -\text{sen}(\theta) & 0 & 0 \\ \text{sen}(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.21.)$$

Translação das distâncias "a", "b", "c" nas direções "x", "y", "z":

$$\text{Trans}(a,b,c) = \begin{vmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.22.)$$

A interpretação geométrica da matriz de transformação homogênea é importante para dar um sentimento físico à ferramenta matemática. Assim tem-se que:

1) As três primeiras colunas da matriz "T" ("n", "o", "a") representam as direções dos novos eixos "u", "v" e "w" em relação ao sistema de coordenadas de referência.

2) A quarta coluna da matriz "T" representa a posição da origem do sistema novo em relação ao sistema de referência.

Em resumo pode-se dizer que a matriz de transformação homogênea representa a posição e orientação do novo sistema de coordenadas em relação ao sistema de referência.

2.1.5. REPRESENTAÇÃO DE DENAVIT-HARTEMBERG

As matrizes de transformação homogênea representam uma sequência de transformações a que deve ser submetido um sistema de coordenadas de referência de modo a coincidir com um outro sistema de coordenadas. Embora isto possa ser feito de diversas maneiras, adotamos a convenção definida em [DENAVIT 55], visto que esta gera matrizes com grande número de termos nulos ou unitários. Ela é conhecida como convenção de Denavit-Hartemberg e consiste na seguinte sequência de operações:

1) Rotação de um ângulo "θ" em torno do eixo "z";
2) Translação de uma distância "d" ao longo do eixo "z";
3) Translação de uma distância "a" ao longo do eixo "x";
4) Rotação de um ângulo "α" em torno do eixo "x".

Os parâmetros de rotação, "α" e "θ", associados aos de translação, "a" e "d", são conhecidos como os parâmetros de Denavit-Hartemberg e estão ilustrados na figura (2.2.).

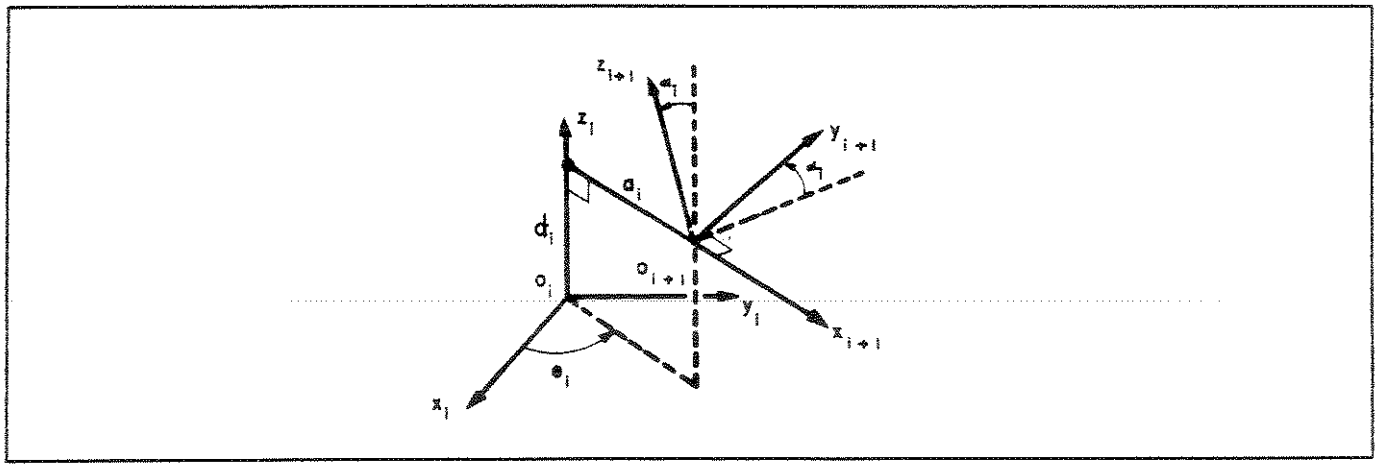


Figura (2.2.): Parâmetros de Denavit-Hartemberg

No caso de ligações rotacionais, os parâmetros "α", "a" e "d" são constantes enquanto "θ" é a variável de ligação, pois muda quando a ligação gira.

No caso de ligações prismáticas os parâmetros "θ", "α" e "a" são constantes enquanto "d" é variável de ligação, pois muda quando a ligação se desloca.

A sequência de operações descrita acima, está indicada na expressão seguinte:

$$T = Rot(z, \theta)\ trans(0,0,d)\ trans(a,0,0)\ Rot(x,\alpha) \qquad (2.23.)$$

Substituindo as equações (2.19.), (2.21.) e (2.22.) obtem-se:

$$T = \begin{vmatrix} \cos\theta & -\text{sen}\theta & 0 & 0 \\ \text{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\text{sen}\theta & 0 \\ 0 & \text{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.24.)$$

Portanto, a matriz "T" tem a forma geral:

$$T = \begin{vmatrix} \cos\theta & -\text{sen}\theta.\cos\alpha & \text{sen}\theta.\text{sen}\alpha & a.\cos\theta \\ \text{sen}\theta & \cos\theta.\cos\alpha & -\cos\theta.\text{sen}\alpha & a.\text{sen}\theta \\ 0 & \text{sen}\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.25.)$$

Considerando o caso de uma cadeia cinemática, as matrizes de transformação homogêneas que representam uma sequência específica de transformações a que deve ser submetido o sistema de coordenadas de um elemento para coincidir com o sistema de coordenadas do elemento subsequente são chamadas de matrizes de transformação homogêneas elementares.

2.1.5.1. ESTABELECIMENTO DE REFERENCIAIS

A figura (2.3.) representa um robô de cadeia cinemática simples

com "n" ligações ($l_1, l_2, \ldots, l_n$).

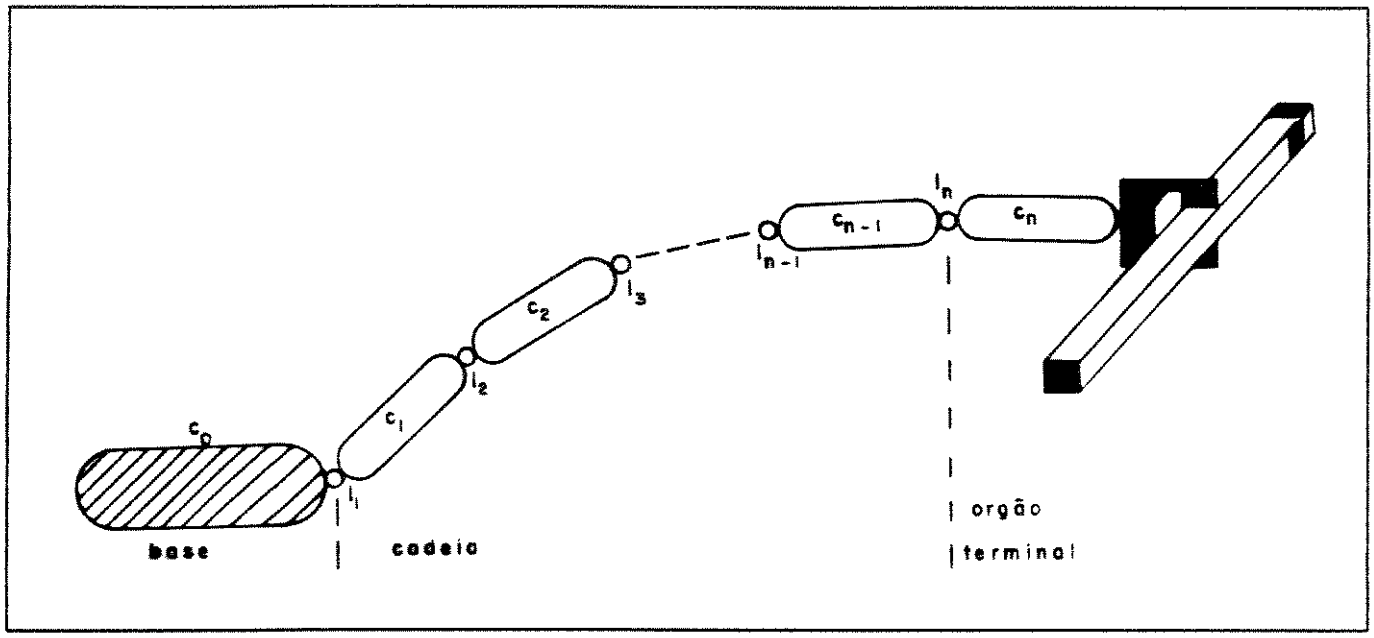


Figura (2.3.): Esquema de um robô de cadeia cinemática simples

Para descrever as relações de translação e rotação entre elementos adjacentes de uma cadeia cinemática, Denavit e Hartemberg propuseram um método para estabelecer de modo sistemático um sistema de coordenadas fixo em cada elemento da cadeia.

Deve-se estabelecer um sistema de coordenadas para a base, que servirá como sistema inercial, e após este um sistema de coordenadas de índice "i" fixo em relação ao elemento "i" e em relação ao qual se movimenta a ligação "i+1", para cada um dos elementos da cadeia.

Exemplificando:

Quando a ligação "i" se movimenta, o elemento "i" se move em relação ao elemento "i-1". Como o sistema de coordenadas "i" é fixo em relação ao elemento "i" eles movem-se juntos em relação ao sistema de índice "i-1".

As regras básicas para estabelecimento de cada sistema de coordenadas são três:

a1) O eixo "z_{i-1}" deve coincidir com o eixo de movimento da ligação de índice "i", sua orientação é arbitrária. Ao orgão terminal deve-se associar o eixo "z_n", onde "n" é o número de ligações.

a2) O eixo "x_i" deve ser normal ao eixo "z_{i-1}", e apontar para longe deste.

a3) O eixo "y_i" deve completar o sistema ortonormal segundo a regra da mão direita.

Segundo estas regras, existe a liberdade de se escolher a posição do sistema de coordenadas inercial da base. É possível estabelecê-lo em qualquer lugar da base desde que seu eixo "z" coincida com o eixo de movimento da ligação "1". Da mesma forma o último sistema de coordenadas pode ser estabelecido em qualquer parte do orgão terminal (que pode ser uma garra por exemplo) desde que o eixo "x_n" seja normal ao eixo "z_{n-1}" .

É possível acrescentar a estas três regras básicas outras regras auxiliares que irão facilitar a obtenção de sistemas de coordenadas que irão gerar modelos geométricos mais simples:

b1) Estabelecer a origem do sistema de coordenadas de índice "i" como sendo a interseção do eixo "z_i" com o eixo "z_{i-1}". Se os dois eixos forem coincidentes ou paralelos, deve-se escolher uma perpendicular comum, levando-se sempre em conta os aspectos de simetria e simplicidade.

b2) Escolher o eixo "x_i" coincidente com o produto vetorial $z_{i-1} \times z_i$ ou na direção da normal comum em caso de paralelismo dos eixos "z". A orientação deve ser no sentido de "z_{i-1}" para "z_i". Se os eixos "z_{i-1}" e "z_{i-1}" forem concorrentes ou coincidentes deve-se escolher a orientação levando-se em conta os aspectos de simetria e simplicidade.

2.1.5.2. RECONHECIMENTO DOS PARÂMETROS DE DENAVIT-HARTEMBERG

Uma vez especificados os sistemas de coordenadas segundo as

regras acima, é preciso obter os parâmetros de Denavit-Hartemberg para que possa ser utilizada a representação matemática que foi apresentada. Para obter-se os parâmetros de uma dada ligação de índice "i" pode-se proceder da seguinte maneira para o par de sistemas de coordenadas consecutivas, de índices "i-1" e "i":

1) Girar o eixo "z_{i-1}" de um ângulo "θ_i" até alinhar o eixo "x_{i-1}" com o eixo "x_i";

2) Transladar ao longo do eixo "z_{i-1}" a distância "d_i" até fazer os eixos "x_{i-1}" e "x_i" coincidentes;

3) Transladar ao longo do eixo "x_{i-1}" a distância "a_i" até fazer as origens dos dois sistemas coincidentes;

4) Girar o eixo "x_{i-1}" de um ângulo "α_i" até fazer os dois eixos "z", e portanto os dois sistemas, coincidentes.

EXEMPLO: Robô V80

A figura (2.4.) representa a cadeia cinemática deste robô com os eixos de rotação das ligações indicados, bem como o eixo de movimentação da garra.

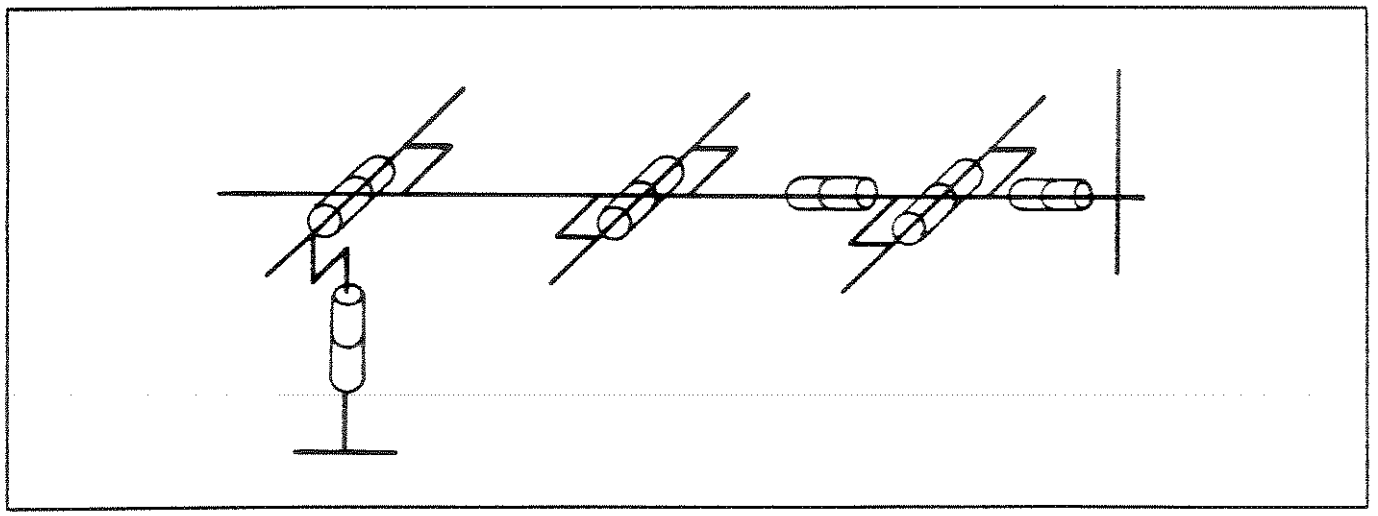


Figura (2.4.): Cadeia cinemática do robô V80

Utilizando a regra a1) estabelecemos as direções e sentidos dos eixos "z", conforme mostrado na figura (2.5.). A orientação destes eixos foi escolhida arbitrariamente.

Utilizando as regras a2) e b2) estabelecemos as direções dos eixos "x" e o sentido de cada um, conforme mostrado na figura (2.6.). Observa-se que:

- As direções e sentidos de "x_1", "x_3", "x_4", "x_5", e "x_6" ficam definidos claramente.

- As direções de "x_0" e "x_2" foram escolhidas arbitrariamente, bem como o sentido de "x_0".

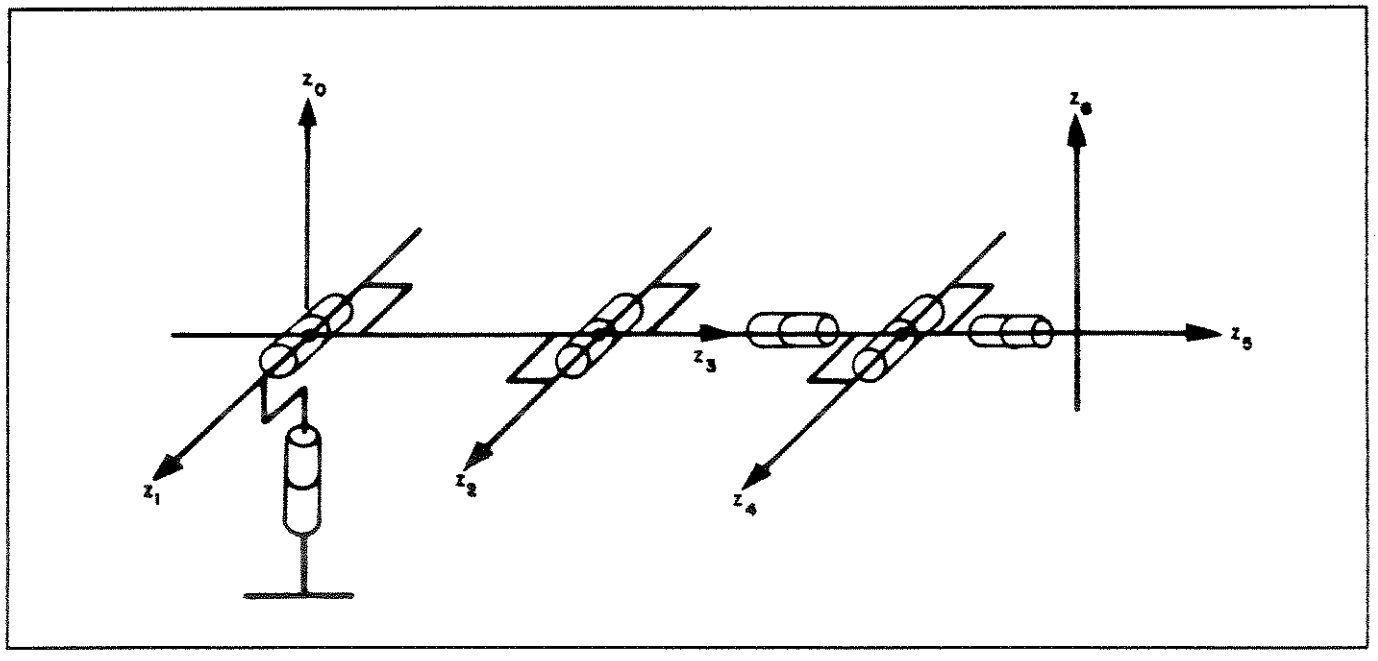


Figura (2.5.): Definição dos eixos "z"

Utilizando a regra b1) determinamos as posições das origens dos sistemas de coordenadas, conforme mostrado na figura (2.6.). Observa-se que:

- As origens "o_1", "o_3", "o_4", "o_5" e "o_6" ficam claramente definidas.

- As origens "o_0" e "o_2" foram escolhidas arbitrariamente, mas de forma indireta, visto que a escolha dos eixos "x_1" e "x_3" foi arbitrária.

Os eixos "y" estão agora definidos pela aplicação da regra a3), mas justamente por serem óbvios não serão representados aqui.

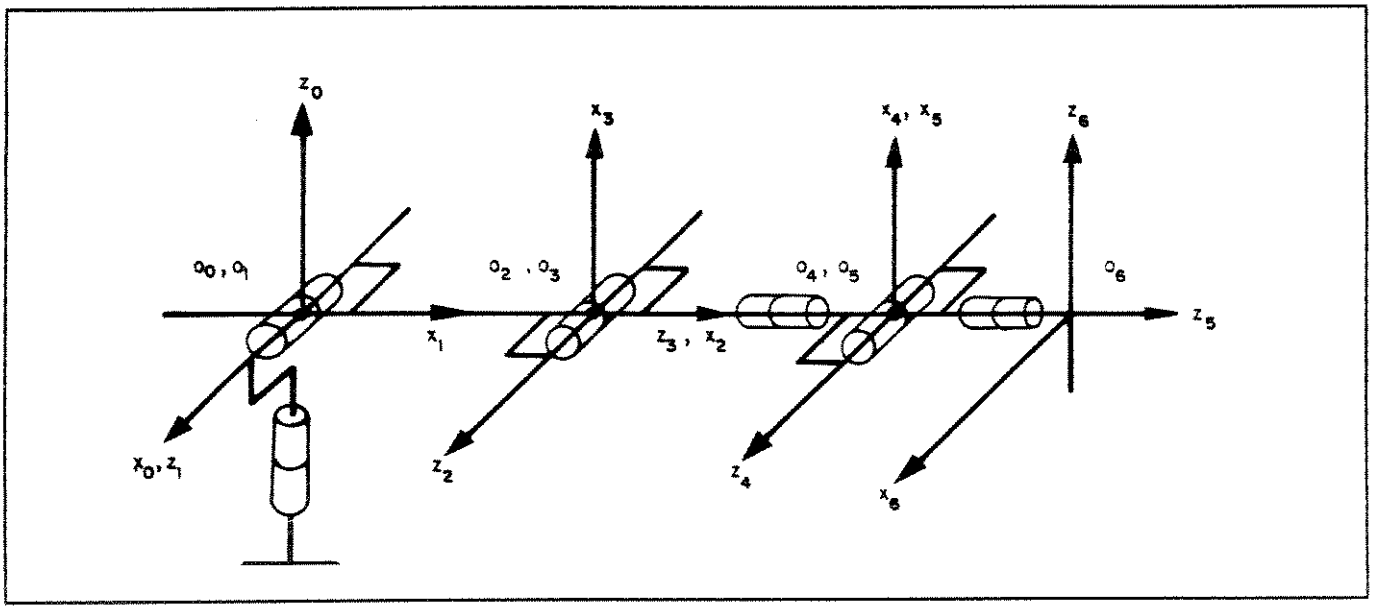


Figura (2.6.): Definição dos eixos "x"

Os parâmetros de Denavit-Hartemberg correspondentes à escolha de referenciais feita são então os seguintes:

junta	θ	α	a	d
1	θ_1	90	0	0
2	θ_2	0	a2	0
3	θ_3	90	0	0
4	θ_4	-90	0	d4
5	θ_5	90	0	0
6	θ_6	90	0	d6

Tabela (2.1.): Parâmetros de Danavit-Hatemberg do robô V80.

2.1.5.3. REPRESENTAÇÃO DE CRAIG-KHALIL

Alguns autores como Craig [CRAIG 85] e Khalil [KHALIL 86], definem a matriz "T" de uma forma um pouco diferente, adotando uma outra seqüência de definição dos parâmetros geométricos e conseqüentemente da matriz "T":

1) Girar o eixo "x_i" de um ângulo "α_i" até alinhar "z_i" com "z_{i+1}";

2) Transladar ao longo de "x_i" a distância "a_i" até fazer "z_i" novo e "z_{i+1}" coincidentes;

3) Girar "z_i" novo de um ângulo "θ_i" até fazer "x_i" e "x_{i+1}" coincidentes;

4) Transladar ao longo de "z_i" novo a distância de "α_i" até fazer as origens dos dois sistemas coincidentes.

Assim:

$$T = ROT(x_i, \alpha_i)\ TRANS(a_i, 0, 0)\ ROT(z_i, \theta_i)\ TRANS(0, 0, d_i) \quad (2.26)$$

E portanto, a forma geral de "T" escrita nesta seqüência será:

$$T = \left|\begin{array}{ccc|c} \cos\theta_i & -\operatorname{sen}\theta_i & 0 & a_i \\ \operatorname{sen}\theta_i\cos\alpha_i & \cos\theta_i.\cos\alpha_i & -\operatorname{sen}\alpha_i & -\operatorname{sen}\alpha_i d_i \\ \operatorname{sen}\theta_i\operatorname{sen}\alpha_i & \cos\theta_i\operatorname{sen}\alpha_i & \cos\alpha_i & \cos\alpha_i d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right| \quad (2.27)$$

Esta notação apresenta em alguns casos a vantagem de se ter um menor número de operações na avaliação do modelo geométrico direto.

2.2. CRITÉRIOS DE OTIMIZAÇÃO DO MODELO GEOMÉTRICO

Neste trabalho, estamos tratando da otimização do modelo geométrico e dos modelos geométricos intermediários para as seguintes finalidades:

A) Cálculo em tempo real do modelo geométrico para controle de trajetória, que fornece as coordenadas operacionais em função das coordenadas generalizadas.

B) Utilização na obtenção do modelo dinâmico (formalismo de Lagrange) através do métodos descritos em: [MEGAHED 84], implementado no GMROB, e [SILVA 90], que faz uso de todas as matrizes intermediárias (trataremos com mais detalhes no item 2.6.).

C) Auxílio à inversão do Modelo para utilização na coordenação de movimentos (trataremos com mais detalhes no Capítulo 3).

No caso A) interessa apenas que a matriz de transformação final leve menor tempo de cálculo (UT), portanto que tenha menor número de operações aritméticas (OA) e cálculos de funções trigonométricas (FT), sendo que o peso relativos destas operações é aproximadamente (por ensaios):

$$1\ UT/OA \quad \text{contra} \quad 10\ UT/FT, \tag{2.28.}$$

quando são usadas as funções residentes nas máquinas, ou

$$1\ UT/OA \quad \text{contra} \quad 1\ UT/FT, \tag{2.29.}$$

se utilizada uma implementação alternativa com valores das FT colocados numa tabela. [CRAIG 85] considera:

$$\begin{gathered} 5\ UT/FT \text{ contra } 1\ UT/\text{multiplicação}, \\ 3\ UT/\text{multiplicação contra } 1\ UT/\text{adição e} \\ 5\ UT/\text{multiplicação contra } 1\ UT/\text{atribuição de valores.} \end{gathered} \tag{2.30.}$$

No caso B) entram todas as matrizes intermediárias mais a matriz de transformação final. O objetivo neste caso, é otimizar o modelo

dinâmico, portanto não basta apenas que cada uma das matrizes individualmente leve o menor tempo de cálculo, temos ainda que considerar a interação com o modelo dinâmico (forma e frequência com que aparecem as matrizes) e a possibilidade de eliminação de expressões redundantes entre as matrizes de transformação utilizadas.

No caso C), como se trata de um método empírico, o que conta é a melhor visualização das expressões do modelo geométrico e das outras expressões que utilizam as matrizes intermediárias.

Já dissemos que os modelos podem ser obtidos de muitas maneiras. A rigor existem infinitas maneiras de se obter o modelo geométrico e para cada robô existe uma maneira que resulta no modelo ótimo. Sem o estabelecimento de critérios mais restritivos, a otimização, resultaria numa busca em árvore com infinitos ramos.

Vamos então descrever as formas de obtenção dos modelos e os critérios adotados neste trabalho.

2.2.1. REPRESENTAÇÃO E PARAMETRIZAÇÃO

Para representação das posições e orientações no espaço operacional em função das coordenadas generalizadas é necessária a adoção de um sistema de coordenadas que pode ser, entre outros vistos na literatura [GORLA 84]:

A) Para posição (veja figura 2.7.):

Coordenadas cartesianas (l, m, n)
Coordenadas cilíndricas (ρ, θ, z)
Coordenadas esféricas (r, θ, ϕ)

B) Para orientação:

Parâmetros (ξ, η, ζ): se obtem o referencial "P_n" do referencial "P_o" pela rotação de um ângulo "ε" em torno do vetor unitário $e = [e_x \; e_y \; e_z]$

$$\xi = e_x \varepsilon \qquad \eta = e_y \varepsilon \qquad \zeta = e_z \varepsilon \qquad (2.31.)$$

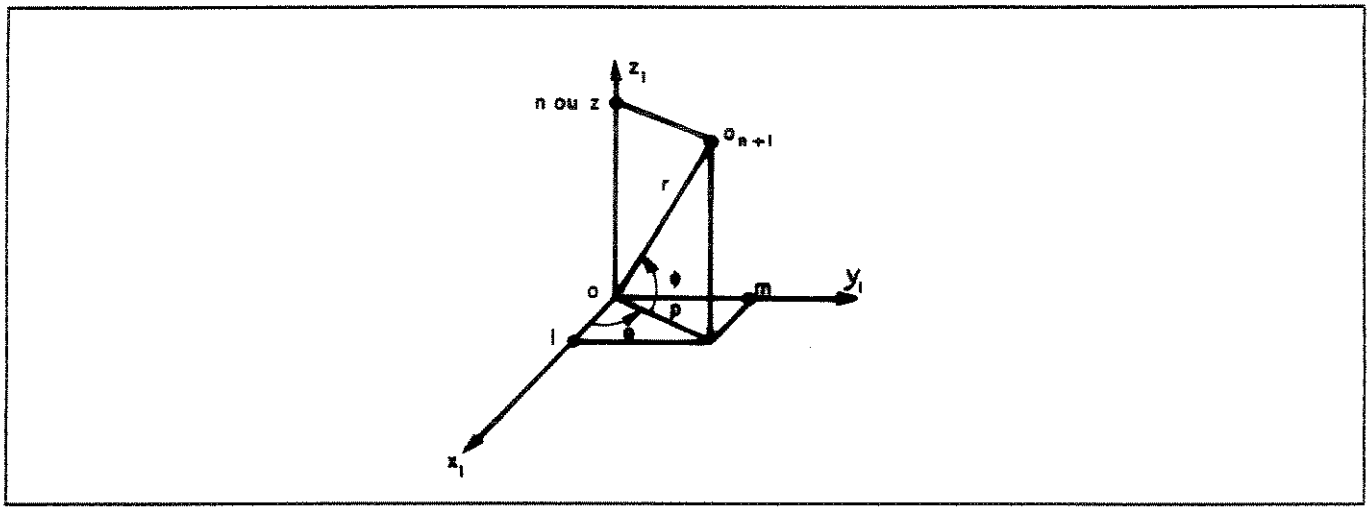


Figura (2.7.): Escolha de Coordenadas para Definir a Posição

Parâmetros de Euler (p, q, r, s): com a mesma notação que os anteriores porém escolhendo como parâmetros:

$$p = e_x \sin(\varepsilon/2) \qquad q = e_y \sin(\varepsilon/2)$$
$$r = e_z \sin(\varepsilon/2) \qquad s = \cos(\varepsilon/2) \tag{2.32.}$$

(parâmetros dependentes pois $p^2 + q^2 + r^2 + s^2 = 1$).

Parâmetros de Rotação Finita (u, v, w):

$$u = e_x \operatorname{tg}(\varepsilon/2) \qquad v = e_y \operatorname{tg}(\varepsilon/2) \qquad w = e_z \operatorname{tg}(\varepsilon/2) \tag{2.33.}$$

$$\left[u = p/s \qquad v = q/s \qquad w = r/s\right] \tag{2.34.}$$

Ângulos de Bryant (λ, μ, ν): "P_n" se obtem por três rotações sucessivas "λ" em torno de "x_o" (eixo "x" de "P_o"), "μ" em torno do novo eixo "y_λ", et "ν" em torno do novo eixo "z_μ".

Cossenos Diretores: considerando-se dois referenciais ortonormais $P_o = [x_o, y_o, z_o]$ e $P_n = [x, y, z]$, onde os vetores [x, y, z] são expressos no referencial "P_o", a matriz "R" que transforma "P_o" em "Pn" é a matriz de transformação clássica, formada por cossenos diretores.

$$Pn = R\ Po \text{ onde } R = \begin{vmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{vmatrix} \tag{2.35.}$$

(parâmetros dependentes, pois [x, y, z] são vetores unitários ortonormais).

Além destas, já vimos no item anterior a representação de Denavit-Hartemberg, obtida com parâmetros "θ", "r", "a", "α" que fornecem o novo referencial afim (com informações de posição e orientação).

Poderia aqui ser questionado o número de parâmetros desta representação ser quatro e não seis (3 de posição e 3 de orientação). Isto pode ser explicado pela natureza restritiva desta representação, adequada a referenciais de elementos consecutivos da cadeia cinemática com movimentos de rotação em torno de um eixo ou translação, e que só pode ser obtida para toda a cadeia cinemática se a colocação de eixos for apropriada (se a escolha do eixo for qualquer, necessariamente temos a escolher uma parametização a 6 parâmetros).

Aliás esta é a grande vantagem (e o motivo da nossa escolha) desta representação, pois fornece matrizes de transformação elementares esparsas, o que significa simplificações no produto das matrizes para obtenção do modelo geométrico.

Não consideraremos neste trabalho as decomposições aproximadas da situação do órgão terminal em uma translação no espaço (movimento amplo) obtido através de uma estrutura de base do robô, e uma rotação no espaço (movimentos finos) obtida pela movimentação do "punho" do robô. Com esta simplificação, os problemas de representação e de controle também são decompostos em correspondentes translação do "punho" pela estrutura de base e orientação do órgão terminal pelo "punho".

Então, quanto ao sistema de coordenadas de posição, a estrutura da base do manipulador (em geral as três primeiras ligações) permitiriam definir o tipo de coordenada que melhor se adapta [GORLA 84].

Neste contexto, no escopo do nosso trabalho, para tratarmos o problema de forma geral, descrevendo sistematicamente toda a cadeia cinemática, para permitir inclusive a automatização da obtenção das matrizes de transformação, optamos pela utilização da representação de

Denavit-Hartemberg (e representações assemelhadas, tipo quatro parâmetros, como a que chamamos Craig-Khalil), por ser a representação mais simplificada.

2.2.2. ALOCAÇÃO DE REFERENCIAIS

A alocação de referenciais num elemento da cadeia cinemática pode ser feita de modo aleatório. A consequência pode ser uma "má" representação, com o aumento do número de operações e aumento de complexidade na obtenção do modelo geométrico (ou geométrico intermediário).

Porém, mesmo adotando um tipo de representação restritivo, como a de Denavit-Hartemberg ou assemelhada, em geral podemos ter algumas orientações diferentes para cada referencial da cadeia, podendo também ser colocados num número infinito de posições. Certamente uma delas (ou várias) vai nos fornecer o melhor modelo.

Por isto, a alocação de referenciais multiplica os ramos da nossa árvore de busca.

Neste trabalho não exploramos métodos melhores que os já existentes na literatura nas questões de Representação e colocação de referenciais, deixando esta abordagem para futuros trabalhos.

Restringimos o nosso problema então à busca do melhor modelo, partindo-se da representação por Denavit-Hartemberg ou assemelhados, (4 parâmetros) fixados também os referenciais. Ou seja, estamos isolando os ramos origem, considerando que o problema parte dos parâmetros de Denavit-Hartemberg ou Craig-Khalil.

Veremos adiante que, mesmo com as restrições "econômicas" feitas anteriormente, o número de alternativas continua muito grande.

2.2.3. SEQUÊNCIA DE MULTIPLICAÇÃO DAS MATRIZES DE TRANSFORMAÇÃO ELEMENTARES

Vamos tratar agora de um dos pontos relevantes no desenvolvimento do nosso trabalho.

Na multiplicação de várias matrizes numéricas não há modificação significativa no resultado se começarmos a multiplicar a penúltima e última, pré-multiplicando sequencialmente as demais (expressão(2.36.)) ou se começarmos pelas primeira e segunda, pós-multiplicando as demais (expressão (2.37.)).

$$(A_1(A_2(A_3(A_4(A_5\ A_6)))))) \qquad (2.36.)$$

$$(((((A_1\ A_2)A_3)A_4)A_4)A_6) \qquad (2.37.)$$

Porém quando estamos tratanto de multiplicação de matrizes simbólicas, que é o caso da obtenção de modelos simbólicas, sequências de multiplicação diferentes resultam em modelos com número de operações diferentes.

Os dois casos acima foram explorados por [Megahed 84]. Porém observamos que o número de sequências de multiplicação possíveis é muito maior que dois, sendo que todas elas devem ser geradas para cada robô (conforme apresentado no exemplo do quadro (2.1.)), devendo então ser computadas as operações aritiméticas e os cálculos de funções trigonométricas para que se posa escolher o melhor modelo.

O exemplo do quadro (2.1.) mostra de todas as sequências de multiplicação possíveis para quatro matrizes (A_1, A_4, A_3, e A_4):

$$(((A_1\ A_2)\ A_3)\ A_4)$$
$$((A_1\ (A_2\ A_3))\ A_4)$$
$$(A_1\ ((A_2\ A_3)\ A_4))$$
$$(A_1\ (A_2\ (A_3\ A_4)))$$
$$((A_1\ A_2)\ (A_3\ A_4))$$

Quadro (2.1.) - Seqüências de Multiplicação de Quatro Matrizes

Através de técnicas de recorrência e partição de números inteiros, chegamos ao número de sequências de multiplicação de "n" matrizes ("f_n"):

$$f_n = \sum_{i=1}^{n-1} f_{n,i} \qquad n \geq 2 \qquad (2.38.)$$

onde:

$$f_{n,i} = \begin{cases} 1 \text{ para } i=1 \text{ ou} \\ f_{n-1,i} + f_{n,i-1} \quad p/\ 2 \leq i \leq n-1, \text{ ou} \\ \qquad\qquad n > 2, \\ f_{n,i} = 0 \text{ para } i = n \end{cases} \qquad (2.39.)$$

que fornece para até n = 7 os valores da tabela (2.2.):

n	fn
2	1
3	2
4	5
5	14
6	42
7	132

Tabela (2.2.) - Número de Seqüências de Multiplicação

Para geração de todas as sequências possíveis, desenvolvemos um algoritmo que tem como entrada o número de matrizes a serem multiplicadas e como saída as "n-1" multiplicações de cada uma das "fn" sequências.

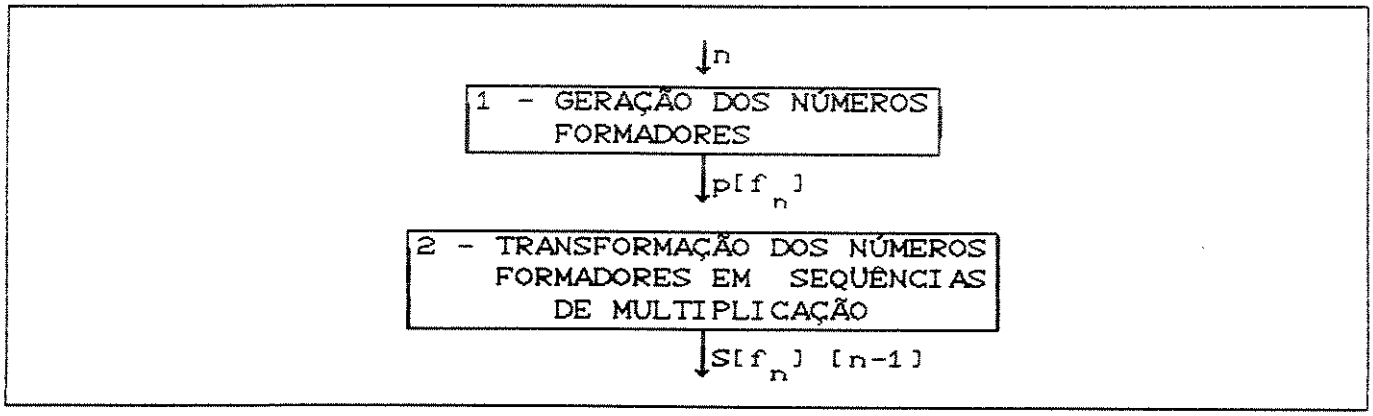
2.2.3.1. ALGORITMO PARA OBTENÇÃO DA SEQUÊNCIA DE MULTIPLICAÇÃO DE N MATRIZES

A construção deste algoritmo foi baseada nas posições de abertura dos parênteses, observando que um conjunto destas "n-1" posições corresponde biunivocamente a uma sequência de multiplicação.

O quadro (2.2.) mostra a seqüência de funções do algoritmo e suas entradas e saídas, que serão descritas as seguir:

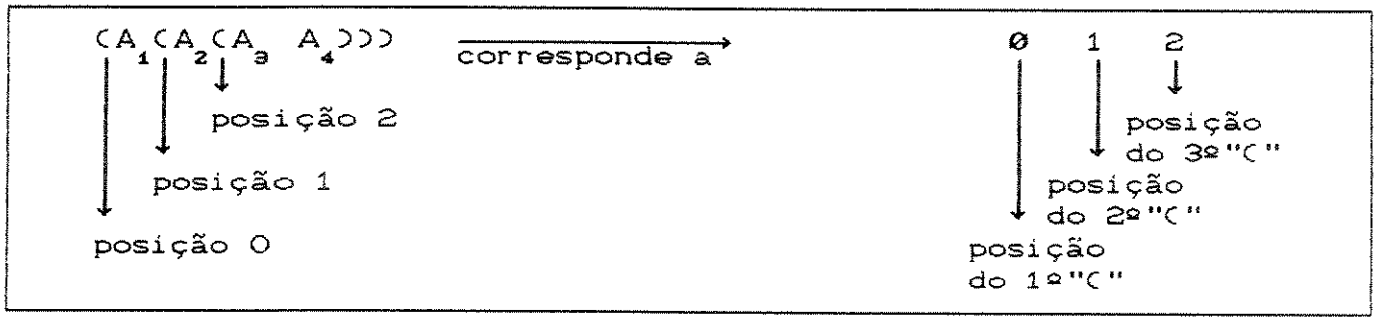
1 - n = número de matrizes

2 - Geração dos Números Formadores: correspondem à posição dos abre-parênteses.



Quadro (2.2.) - Algoritmo para Obtenção das Seqüências de Multiplicação de "n" Matrizes

O exemplo do quadro (2.3.) mostra os números formadores correspondentes a uma determinada seqüência de multiplicação de quatro matrizes (A_1, A_2, A_3 e A_4).



Quadro (2.3.) - Geração de Números Formadores para Quatro Matrizes

3 - $p[f_n]$ = vetor com as "f_n" sequências de "n-1" caracteres numéricos correspondentes aos números formadores. No caso acima para algum "i", p[i] = 0 1 2.

4 - Transformação dos números formadores em sequências de multiplicação: obtem as "n-1" sequências de multiplicação de cada um dos "f_n" vetores "p[k]", onde as multiplicações vão sendo efetuadas e substituídas por cada um dos "n-2" caracteres alfabéticos de uma cadeia de caracteres "c" fornecida.

O exemplo do quadro (2.4.) mostra a seqüência de multiplicação correspondente aos números formadores obtidos no exemplo do quadro (2.3.).

c = [bd]	multiplicação	falta multiplicar
		1234
1ª multiplicação: (A_3A_4)	34 ⟶ b	12b
2ª multiplicação: $(A_2(A_3\ A_4))$	2b ⟶ d	1d
3ª multiplicação: $(A_1(A_2(A_3A_4)))$	1d	

Quadro (2.4.) - Transformação dos Números Formadores em Seqüências de Multiplicação

5 - S [f_n] [n-2] = matriz com as (n-1) sequências de multiplicação de cada um dos f_n vetores p [k]. Para p[i] = Ø 1 2 temos S[i] = [34 2b 1d].

Os quadros (2.5.) e (2.7.) mostram os passos para se obter respectivamente os números formadores e as seqüências de multiplicação. Os quadros (2.6.) e (2.8.) apresentam exemplos destas duas funções para multiplicação de quatro matrizes.

GERAÇÃO DOS NÚMEROS FORMADORES

$p_1 = p_2 = \ldots\ldots\ p_{n-1} =$ Ø (p_i é caracter numérico)

k = 0 (k é inteiro)

p[k = k + 1] = $p_1p_2\ldots p_{n-1}$ (p[k] é a cadeia de caracteres p_i)

para i variando de (n-1) até 2 com decremento unitário

l = i

(*) para jl variando de max (1, p_{l-1}) até (l - 1)

pl = jl

se l < (n-1), faça recorrência em (*) com l = l + 1

caso contrário p[k = k + 1] = $p_1p_2\ldots\ldots p_{n-1}$

f_n = último valor de k (fn é o número de sequências)

Quadro (2.5.) - Geração dos Números Formadores

```
EXEMPLO PARA QUATRO MATRIZES (N = 4):

p [1] = 0

i variando de 3 até 2

i = 3

     l = 3

     j3 variando de 1 até 2

     j3 = 1

          p3 = 1

          (3 < (4-1)) = falso  →  p[2] = 001

     j3 = 2

          p3 = 2

          (3 < (4-1)) = falso  →  p[3] = 002

i = 2

     l =2

     j2 variando de 1 até 1

     j2 = 1

          p2 = 1

          (2 < 3) = verdadeiro  →  recorrência com l = 3

          l = 3

          j3 variando de 1 até 2

          j3 = 1

               p3 = 1

               (3 < (4-1)) = falso  ⇒  p[4] = 011

          j3 = 2

               p3 = 2

               (3 < (4-1)) = falso  ⇒  p[5] = 012

f4 = 5
```

Quadro (2.6.) - Geração dos Números Formadores para Quatro Matrizes

O resultado final do exemplo do quadro (2.6.) é: p[5] = [000 001 002 011 012], que correspondem às posições dos 3 abre-parênteses das cinco sequências de multiplicação de $A_1A_2A_3A_4$.

TRANSFORMAÇÃO DOS NÚMEROS FORMADORES
EM SEQÜÊNCIAS DE MULTIPLICAÇÃO

Arbitre c = $c_1c_2 \ldots c_{n-2}$ (c é uma cadeia de caracteres c_i)

Construa a = $a_1a_2 \ldots a_n$ (onde $a_1 = 1$, $a_2 = 2 \ldots a_n = n$)

(sendo "a" a cadeia de caracteres correspondentes aos números das matrizes)

para k variando de 1 até fn

p = p[k] (cadeia de caracteres numéricos p_j)

para i variando de 0 até n-2

leia p_{n-1-i}

construa S[k] [i + 1] = $a_{p_{n-1-i}+1}$ $a_{p_{n-1-i}+2}$

se (i < (n-2)) construa "a", substituindo a sequência acima pelo caracter c_{i+1} : a = $a_1a_2 \ldots a_{n-1-i}$, onde $a_l = c_{i+1}$ para algum l entre 1 e (n-1)

caso contrário reconstrua a seqüência original de "a"

Quadro (2.7.) - Transformação dos Números Formadores em Seqüências de Multiplicação

O exemplo do quadro (2.8.), mostrado a seguir, é continuação do exemplo do quadro (2.6.).

EXEMPLO PARA QUATRO MATRIZES (N = 4)

c = bd

a = 1234

para k variando de 1 até 5

k = 1

p = 000

para i variando de 0 até 2

i = 0

p_3 = 0

S[1] [1] = 12

c_1 = b

a = b34

i = 1

p_2 = 0

S[1] [2] = b3

c_2 = d

a = d4

i = 2

p_1 = 0

S[1] [3] = d4

a = 1234

k = 2

...

Quadro (2.8.) - Transformação dos Números Formadores para Quatro Matrizes

2.2.4. EQUAÇÕES AUXILIARES, SIMPLIFICAÇÕES TRIGONOMÉTRICAS E ARITMÉTICAS, USO DE PROPRIEDADES DE BASES ORTONORMAIS

Os aspectos tratados neste item também foram objetos prioritários no desenvolvimento do nosso trabalho.

Fixada a representação por matrizes de transformação homogêneas e adotados os sistemas de coordenadas (itens 2.2.1. e 2.2.2), existem ainda várias formas de obtenção equivalentes quantitativamente mas que podem diferir muito quanto ao número de operações aritméticas e mesmo quanto ao número de avaliações de senos e cossenos. Esta diferença está relacionada não só com a sequência de multiplicação das matrizes de transformação elementares que acabamos de ver (item 2.2.3.) mas também com a utilização de equações auxiliares, utilização de simplificações aritméticas e trigonométricas, e a utilização de propriedades de vetores ortonormais.

2.2.4.1. EQUAÇÕES AUXILIARES

No intento de minimizar o número de operações aritméticas necessárias para avaliar o modelo geométrico, pode-se utilizar a idéia [Megahed 84] de, durante o processo de geração do modelo, substituir-se toda expressão aritmética parcial que surgir no decorrer da multiplicação das matrizes elementares, por variáveis auxiliares, de modo a evitar-se a redundância de cálculos no modelo.

Para mostrar sua importância daremos aqui um exemplo de modelo direto com uso de equações auxiliares e sem, com as comparações do número de avaliação de funções trigonométricas. Estes resultados foram gerados pelo SIS_MGD (gerador de modelos geométricos doGMROB), ferramenta que será descrita no item 2.3..

Foi utilizado o robô K16, com representação de Denavit-Hartemberg, cujos parâmetros estão mostrados na tabela (2.3.), com ordem direta de multiplicação (método 1 de GMROB), ou seja $(((((T_{01} * T_{12}) * T_{23}) * T_{34}) * T_{45}) * T_{56})$.

Para simplificar, adotaremos a partir daqui a notação: $C_i = \cos\theta_i$, $S_i = \sin\theta_i$, $C_{\alpha i} = \cos\alpha_i$ e $S_{\alpha i} = \sin\alpha_i$.

Convenção D-H adotada: Tradicional

junta	tipo	θ	α	a	d
1	R	θ_2	90	0	di
2	R	θ_2	0	a2	0
3	R	θ_3	0	a3	0
4	R	θ_4	90	0	0
5	R	θ_5	-90	0	d5
6	R	θ_6	90	0	0

Tabela (2.3.) - Parâmetros de Denavit-Hartemberg para o Robô K16

O quadro (2.9.) mostra o modelo geométrico (matriz $T_{on}[\ ][\ ]$) obtido sem utilização das equações auxiliares:

```
T on[0][0] = ((((C1C2C3-C1S2S3)C4 + (-C1C2S3-C1S2C3)S4)C5 + S1S5)C6 +
           - ((C1C2C3-C1S2S3)S4 - (-C1C2S3-C1S2C3)C4)S6);

T on[0][1] = (-((C1C2C3-C1S2S3)C4 + (-C1C2S3-C1S2C3)S4)S5 + S1C5);

T on[0][2] = ((((C1C2C3-C1S2S3)C4 + (-C1C2S3-C1S2C3)S4)C5 + S1S5)S6 +
           + ((C1C2C3-C1S2S3)S4 - (-C1C2S3-C1S2C3)C4)C6);

T on[0][3] =  (((C1C2C3-C1S2S3)S4 - (-C1C2S3-C1S2C3)C4)d5 +
           +  (C1C2a3C3 - C1S2a3S3 + C1a2C2));

T on[1][0] = ((((S1C2C3-S1S2S3)C4 + (-S1C2S3-S1S2C3)S4)C5 - C1S5)C6 +
           - ((S1C2C3-S1S2S3)S4 - (-S1C2S3-S1S2C3)C4)S6);

T on[1][1] = (-((S1C2C3-S1S2S3)C4 + (-S1C2S3-S1S2C3)S4)S5 - C1C5);

T on[1][2] = ((((S1C2C3-S1S2S3)C4 + (-S1C2S3-S1S2C3)S4)C5 - C1S5)S6 +
           + ((S1C2C3-S1S2S3)S4 - (-S1C2S3-S1S2C3)C4)C6);

T on[1][3] =  (((S1C2C3-S1S2S3)S4 - (-S1C2S3-S1S2C3)C4)d5 +
           +  (S1C2a3C3 - S1S2a3S3 + S1a2C2));

T on[2][0] =  (((S2C3+C2S3)C4 + (-S2S3+C2C3)S4)C5C6 - ((S2C3+C2S3)S4
           - (-S2S3+C2C3)C4)S6);

T on[2][1] = - ((S2C3+C2S3)C4 + (-S2S3+C2C3)S4)S5;

T on[2][2] =  (((S2C3+C2S3)C4 + (-S2S3+C2C3)S4)C5S6 + ((S2C3+C2S3)S4
           - (-S2S3+C2C3)C4)C6);

T on[2][3] =  (((S2C3+C2S3)S4 - (-S2S3+C2C3)C4)d5 + (S2a3C3+C2a3S3 +
           +  (a2S2+d1)));
```

Quadro (2.9.) - Modelo Geométrico do K16 sem Utilização das Equações Auxiliares

O quadro (2.10.) mostra o modelo geométrico obtido com uso de equações auxiliares:

```
D[1]  =  a2C2                          D[28] = -D[21]S5+S1C5
D[2]  =  a2S2                          D[29] =  D[22]d5+D[14]
D[3]  =  C1C2                          D[30] =  D[23]C5-C1S5
D[4]  = -C1S2                          D[31] = -D[23]S5-C1C5
D[5]  =  C1D[1]                        D[32] =  D[24]d5+D[17]
D[6]  =  S1C2                          D[33] =  D[25]C5
D[7]  = -S1S2                          D[34] = -D[25]S5
D[8]  =  S1D[1]                        D[35] =  D[26]d5+D[20]
D[9]  =  D[2]+d1                       D[36] =  D[27]C6-D[22]S6
D[10] =  a3C3                          D[37] =  D[27]S6+D[22]C6
D[11] =  a3S3                          D[38] =  D[30]C6-D[24]S6
D[12] =  D[3]C3+D[4]S3                 D[39] =  D[30]S6+D[24]C6
D[13] = -D[3]S3+D[4]C3                 D[40] =  D[33]C6-D[26]S6
D[14] =  D[3]D[10]+D[4]D[11]+D[5]      D[41] =  D[33]S6+D[26]C6
D[15] =  D[6]C3+D[7]S3
D[16] = -D[6]S3+D[7]C3                 Ton[0][0] = D[36]
D[17] =  D[6]D[10]+D[7]D[11]+D[8]      Ton[0][1] = D[28]
D[18] =  S2C3+C2S3                     Ton[0][2] = D[37]
D[19] = -S2S3+C2C3                     Ton[0][3] = D[29]
D[20] =  S2D[10]+C2D[11]+D[9]          Ton[1][0] = D[38]
D[21] =  D[12]C4+D[13]S4               Ton[1][1] = D[31]
D[22] =  D[12]S4-D[13]C4               Ton[1][2] = D[39]
D[23] =  D[15]C4+D[16]S4               Ton[1][3] = D[32]
D[24] =  D[15]S4-D[16]C4               Ton[2][0] = D[40]
D[25] =  D[18]C4+D[19]S4               Ton[2][1] = D[34]
D[26] =  D[18]S4-D[19]C4               Ton[2][2] = D[41]
D[27] =  D[21]C5+S1S5                  Ton[2][3] = D[35]
```

Quadro (2.10.) - Modelo Geométrico do K16 com Utilização das Equações Auxiliares

O quadro (2.11) mostra o total de operações requeridas para os cálculos dos dois modelos sem e com equações auxiliares (quadros (2.9.) e (2.10.) respectivamente).

Método	E.A.	+	-	*	Total de Operações	Sin-Cos
1	sem	35	41	207	283	12
1	com	24	8	65	97	12

Quadro (2.11) - Número de Operações com e sem o Emprego de Equações Auxiliares

Uma observação que deve ser frisada é que, para o caso do auxílio à inversão do modelo, não seria recomendável a utilização de equações auxiliares pois, como já foi dito, o método utilizado exige a visualização completa dos termos das matrizes.

2.2.4.2. SIMPLIFICAÇÕES ARITMÉTICAS

Outro recurso para simplificação de modelos é o das simplificações aritméticas, que abrange a evidenciação de termos (explícita ou por parênteses), efetuação das somas e multiplicações por zero e da multiplicação por um.

No caso da efetuação de operações com zero e uns e evidenciação de termos que aparecem numa mesma equação auxiliar, está claro que a multiplicação simbólica já deve levar isto em consideração. Mas no caso da evidenciação de termos pertencentes a duas equações auxiliares diferentes, há uma contraposição entre o uso de equações auxiliares e o uso das evidenciações.

No entanto observamos que as evidenciações são automaticamente consideradas quando fazemos o produto das matrizes através de outra sequência utilizando equações auxiliares. Ou seja, o próprio uso das equações auxiliares é um procedimento de evidenciação implícita.

Com a análise de diversos casos, notamos que uma simplificação que diferencia intrinsecamente as sequências de multiplicações é a relativa às evidenciações.

Na questão das simplificações aritméticas, consideramos que duas abordagens estruturais distintas (excludentes) podem ser adotadas: 1) construção de um analisador sintático para execução das simplificações "a posteriori" e 2) previsão das possíveis simplificações e execução "a priori" e simultaneamente à obtenção de cada matriz intermediária. O nosso trabalho adota a segunda abordagem (não só para simplificações aritméticas como também para simplificações trigonométricas).

A justificativa para esta abordagem é que, no caso de uso de um analisador sintático para simplificações "a posteriori", duas opções poderiam ser adotadas: 1ª) analisador geral, excluído por causa do custo impraticável, 2ª) analisador dedicado, cuja construção dependeria, de qualquer maneira, do conhecimento acumulado nas simplificações "a priori", sem o qual não se teriam elementos para construí-lo.

2.2.4.3. SIMPLIFICAÇÕES TRIGONOMÉTRICAS

As simplificações trigonométricas, no caso de modelagem geométrica se referem a soma de ângulos (cosθ_i cosθ_j $\mp$ senθ_i senθ_j) = cos ($\theta_i \pm \theta_j$), (cosθ_i senθ_j $\pm$ senθ_i cosθ_j) = sen ($\theta_i \pm \theta_j$)) e somas de quadrados de seno e cosseno do mesmo ângulo ((cos$^2\theta_i$ + sen$^2\theta_i$) = 1).

Na obtenção do modelo geométrico através de multiplicação das matrizes de transformação elementares, observamos por construção, que não aparecem casos de somas de quadrados de senos e cosenos do mesmo ângulo. Quanto ao caso de somas de ângulos, existe a mesma contraposição a equações auxiliares que no caso da evidenciação, sendo que no caso das simplificações trigonométricas, algumas vezes se mostrou "econômico" abandonar parcialmente o recurso das equações auxiliares e utilizar as simplificações trigonométricas, como no caso apresentado a seguir.

Foram avaliados dois modelos do robô Puma, cujos parâmetros de Denavit-Hartemberg estão apresentados na tabela (2.4.). Os dois modelos, obtidos também com o auxílio do gerador de modelos geométricos do GMROB e apresentados nos quadros (2.12.) e (2.13.), utilizam pós multiplicação (((((T$_{01}$ * T$_{12}$) * T$_{23}$) * T$_{34}$) * T$_{45}$) * T$_{56}$), sendo que na obtenção do primeiro modelo foi apenas utilizado o recurso de equações auxiliares (método 1 do GMROB com equações auxiliares) e no segundo foram utilizadas também simplificações trigonométricas (método 5 do GMROB com equações auxiliares). O resultado das avaliações está apresentado no quadro (2.14.).

Convenção D-H adotada : tradicional

junta	tipo	θ	α	a	d
1	R	θ_1	-90	0	0
2	R	θ_2	0	a2	d2
3	R	θ_3	90	a3	0
4	R	θ_4	-90	0	d4
5	R	θ_5	90	0	0
6	R	θ_6	0	0	d6

Tabela (2.4.) - Parâmetros de Denavit-Hartemberg para o Robô Puma

```
D[1]  =  a2C2                          D[29] =  D[20]C5-D[12]S5
D[2]  =  a2S2                          D[30] =  D[20]S5+D[12]C5
D[3]  =  C1C2                          D[31] =  D[23]C5-D[15]S5
D[4]  = -C1S2                          D[32] =  D[23]S5+D[15]C5
D[5]  =  C1D[1]-S1d2                   D[33] =  D[26]C5-D[18]S5
D[6]  =  S1C2                          D[34] =  D[26]S5+D[18]C5
D[7]  = -S1S2                          D[35] =  D[29]C6+D[21]S6
D[8]  =  S1D[1]+C1d2                   D[36] = -D[29]S6+D[21]C6
D[9]  =  a3C3                          D[37] =  D[30]d6+D[22]
D[10] =  a3S3                          D[38] =  D[31]C6+D[24]S6
D[11] =  D[3]C3+D[4]S3                 D[39] = -D[31]S6+D[24]C6
D[12] =  D[3]S3-D[4]C3                 D[40] =  D[32]d6+D[25]
D[13] =  D[3]D[9]+D[4]D[10]+D[5]       D[41] =  D[33]C6+D[27]S6
D[14] =  D[6]C3+D[7]S3                 D[42] = -D[33]S6+D[27]C6
D[15] =  D[6]S3-D[7]C3                 D[43] =  D[34]d6+D[28]
D[16] =  D[6]D[9]+D[7]D[10]+D[8]
D[17] = -S2C3-C2S3                     Ton[0][0] = D[35]
D[18] = -S2S3+C2C3                     Ton[0][1] = D[36]
D[19] = -S2D[9]-C2D[10]-D[2]           Ton[0][2] = D[30]
D[20] =  D[11]C4-S1S4                  Ton[0][3] = D[37]
D[21] =  D[11]S4-S1C4                  Ton[1][0] = D[38]
D[22] =  D[12]d4+D[13]                 Ton[1][1] = D[39]
D[23] =  D[14]C4+C1S4                  Ton[1][2] = D[32]
D[24] = -D[14]S4+C1C4                  Ton[1][3] = D[40]
D[25] =  D[15]d4+D[16]                 Ton[2][0] = D[41]
D[26] =  D[17]C4                       Ton[2][1] = D[42]
D[27] = -D[17]S4                       Ton[2][2] = D[34]
D[28] =  D[18]d4+D[19]                 Ton[2][3] = D[43]
```

Quadro (2.12.) - Modelo Geométrico do Puma sem Utilização das Simplificações Trigonométricas

Da mesma forma que nas simplificações aritméticas, nas simplificações trigonométricas a abordagem é a simplificação simultânea ou "a priori" das multiplicações das matrizes intermediárias. Para isto desenvolvemos um novo método para obtenção das matrizes intermediárias (ver item 2.4), que facilita a dedução analítica dos casos de simplificações trigonométricas (ver item 2.5) e nos permite visualizar a contraposição das equações auxiliares com as simplificações trigonométricas.

Uma observação importante é quanto ao auxílio à inversão do modelo em que, ao contrário das equações auxiliares, as simplificações trigonométricas auxiliam bastante na visualização das expressões.

```
D[1]  =  a2C2                    D[26] =  D[17]S5+D[11]C5
D[2]  =  a2S2                    D[27] =  D[20]C5-C23S5
D[3]  =  C1D[1]-S1d2             D[28] =  D[20]S5+C23C5
D[4]  =  S1D[1]+C1d2             D[29] =  D[23]C6+D[15]S6
D[5]  =  0                       D[30] = -D[23]S6+D[15]C6
D[6]  =  0                       D[31] =  D[24]d6+D[16]
D[7]  =  C1C23                   D[32] =  D[25]C6+D[18]S6
D[8]  =  S1C23                   D[33] = -D[25]S6+D[18]C6
D[9]  =  C1S23                   D[34] =  D[26]d6+D[19]
D[10] =  a3D[7]+D[3]             D[35] =  D[27]C6+D[21]S6
D[11] =  S1S23                   D[36] = -D[27]S6+D[21]C6
D[12] =  a3D[8]+D[4]             D[37] =  D[28]d6+D[22]
D[13] = -a3S23-D[2]
D[14] =  D[7]C4-S1S4             Ton[0][0] = D[29]
D[15] = -D[7]S4-S1C4             Ton[0][1] = D[30]
D[16] =  D[9]d4+D[10]            Ton[0][2] = D[24]
D[17] =  D[8]C4+C1S4             Ton[0][3] = D[31]
D[18] = -D[8]S4+C1C4             Ton[1][0] = D[32]
D[19] =  D[11]d4+D[12]           Ton[1][1] = D[33]
D[20] = -S23C4                   Ton[1][2] = D[26]
D[21] =  S23S4                   Ton[1][3] = D[34]
D[22] =  C23d4+D[13]             Ton[2][0] = D[35]
D[23] =  D[14]C5-D[9]S5          Ton[2][1] = D[36]
D[24] =  D[14]S5+D[9]C5          Ton[2][2] = D[28]
D[25] =  D[17]C5-D[11]S5         Ton[2][3] = D[37]
```

Quadro (2.13.) - Modelo Geométrico do Puma com Utilização das Simplificações Trigonométricas

Método	E.A.	+	-	*	Total de Operações	Sin-Cos
1	com	25	11	70	106	12
5	com	21	7	53	81	14

Quadro (2.14) - Número de Operações com e sem o Emprego de Simplificações Trigonométricas

2.2.4.4. USO DE PROPRIEDADES DE BASES ORTONORMAIS

Outra forma de cálculo das matrizes intermediárias é a obtenção de apenas duas colunas da matriz-partição de orientação R (cf. 2.1.4.) com obtenção da terceira pelo produto vetorial das duas anteriores (propriedade de bases ortonormais). A vantagem de uso ou não deste recurso deve ser verificado a cada matriz intermediária calculada.

Este recurso foi utilizado por [MEGAHED 84] em dois dos seus quatro métodos de cálculos do modelo geométrico.

Veremos no item 2.3 os métodos utilizados nos [MEGAHED 84] e, também os utilizados no gerador de modelos de GMROB. O presente

trabalho, mais os dois citados indicam a necessidade de se dispor de uma bateria de métodos de cálculos do modelo geométrico e geométricos intermediários, que devem ser executados para cada robô, pois não há como determinar "a priori" o melhor método para um dado robô.

Para o robô TH8, cujos parâmetros de Denavit-Hartemberg estão mostrados na tabela (2.5.), apresentamos no quadro (2.15.), as avaliações do número de operações dos modelos geométricos obtidos pela sequência de multiplicação $(T_{01}(T_{12}(T_{23}(T_{34}(T_{45}\ T_{56})))))$, primeiramente sem o emprego das propriedades de bases ortonormais (método 3 do GMROB) e depois com utilização das propriedades de bases ortonormais (método 4 do GMROB).

Convenção D-H adotada : tradicional

junta	tipo	θ	α	a	d
1	R	θ_1	0	0	0
2	P	0	90	a2	d2
3	P	0	0	0	d3
4	R	θ_4	90	0	0
5	R	θ_5	90	0	0
6	R	θ_6	90	0	d6

Tabela (2.5.) - Parâmetros de Denavit-Hartemberg para o Robô Th8

Método	E.A.	+	-	*	Total de Operações	Sin-Cos
3	com	19	6	34	49	8
4	com	7	7	30	44	8

Quadro (2.15) - Número de Operações sem e com o Emprego das Propriedades de Bases Ortonormais

2.3. GERAÇÃO AUTOMÁTICA DE MODELOS GEOMÉTRICOS

2.3.1. CÁLCULO SIMBÓLICO

Como já vimos nos itens 2.1 e 2.2, embora a definição do modelo geométrico seja única, a maneira de obter a matriz de passagem homogênea do robô não o é. Existem diversas alternativas, e 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. Tendo em vista as possíveis aplicações do modelo em sistemas de controle em tempo real e na obtenção do modelo dinâmico, é importante que se obtenha um modelo com o menor número possível de operações matemáticas, permitindo sua avaliação numérica no menor

Em [MEGAHED 84] foram apresentados quatro métodos, todos utilizando equações auxiliares:

O método 1 monta as matrizes elementares simbólicas e faz pós-multiplicações matriciais simbólicas a partir da matriz relativa à primeira ligação da cadeia cinemática.

$$\text{Método 1} \Rightarrow ((.((T_{01}\ T_{12})T_{23})...)T_{n-1\ n}) \qquad (2.40.)$$

O método 2 é idêntico ao método 1, porém utilizando propriedades de bases ortonormais.

O método 3 monta as matrizes elementares simbólicas e faz pré-multiplicações matriciais simbólicas a partir da matriz relativa à última ligação da cadeia cinemática.

$$\text{Método 3} \Rightarrow (T_{01}(T_{12}(...(T_{n-1\ n-2}\ T_{n-1n}))).)) \qquad (2.41.)$$

O método 4 é idêntico ao método 3, porém utilizando propriedades de bases ortonormais.

O Gerador de Modelos Geométricos do GMROB (SIS_MGD) apresenta 5

métodos mais as opções de utilizar ou não:

a) A convenção de CRAIG (na qual a montagem das matrizes elementares é feita de forma diferente) e

b) Equações auxiliares,

totalizando vinte maneiras diferentes de obtenção do modelo.

Os quatro primeiros métodos são idênticos aos utilizados por [Megahed 84]. O Método 5 é um método híbrido, mesclando equações auxiliares com simplificações trigonométricas, em casos de paralelismos de eixos em eixos consecutivos ou entremeados por ligações prismáticas apenas.

2.3.2. MODELAGEM AUTOMÁTICA

O processo de obtenção manual de modelos geométricos é bastante moroso e muito susceptivel a erros; só este fato já justificaria uma ferramenta para modelagem automática. Além disso, considerando que estamos a procura do melhor modelo, com tantas alternativas na busca, seria impraticável executar esta tarefa sem o auxilio de ferramenta para modelagem automática.

Achamos oportuno ressaltar aqui a diferença entre método de geração automática e ferramenta de geração automática, utilizando como exemplo o caso dos métodos de [Megahed 84], onde os programas disponíveis eram para o estrito uso pessoal do autor, não constituindo portanto uma ferramenta de uso geral. O que buscamos foi tornar estes métodos (e outros) ferramenta utilizável por outras pessoas para obtenção de modelos.

O SIS_MGD foi desenvolvido com a filosofia de ferramenta de modelagem automática, mas já superou em muito sua expectativa inicial de utilização, tendo não só reduzido etapas de outros trabalhos e validado modelos diversos (com o uso vulgarizado pelas III^{a} e IV^{a} EBAI), mas também servido de ferramenta para ampliar o próprio escopo, fornecendo subsídios para a proposição de uma metodologia mais genérica do que a contida no GMROB, que é um dos objetivos desta dissertação.

O Módulo acima foi implementado baseado em quatro funções:

1) Supervisão: responsável pela coordenação das demais funções;

2) Inicialização: responsável pela geração e deleção de arquivos de dados. Este módulo permite que os parâmetros de Denavit-Hartemberg de um robô, uma vez especificados, possam ser acessados posteriormente referenciando-se apenas ao nome do robô. Permite também a visualização de arquivos de dados específicos e do diretório de arquivos.

3) Geração do modelo: responsável pela geração do modelo segundo as diversas alternativas disponíveis e pela geração dos arquivos de saída, que apresentam os resultados e servem como interface com o módulo de geração de modelos dinâmicos.

4) Funções básicas de computação simbólica: responsável pela execução de todas as operações com cadeias de caracteres ou números.

Este módulo apresenta ainda as seguintes características:

- Codificado em linguagem "C", com uso dos recursos de alocação dinâmica;
- Implementação em máquinas compatíveis com o IBM PC-XT;
- Capacidade de tratamento tanto de cadeias de caracteres quanto de valores numéricos. Isto quer dizer que pode-se tratar matrizes de transformação totalmente literais, numéricas ou híbridas, onde os parâmetros de Denavit-Hartemberg que são fixos podem ser representados por seus valores numéricos. Qualquer operação entre dois valores numéricos é efetuada e seu resultado substitui a operação antes indicada. Ainda neste aspecto, é feita a simplificação automática de produtos por "1", "-1" ou zero e de somas com zero.
- Geração de resultados na forma compilável (linguagem "C"). Esta característica permite que os resultados obtidos possam ser avaliados numericamente sem necessidade de digitação das expressões obtidas em um outro programa.

Além disto, como já citamos, este módulo possibilita a representação das matrizes elementares segundo a convenção normal de Denavit-Hartemberg ou segundo a convenção modificada apresentada por [CRAIG 85], bem como a geração dos modelos usando as variáveis auxiliares ou sem elas. No caso da não utilização de equações auxiliares, o sistema apenas justapõe os termos separando-os com o uso de parênteses.

O módulo para Geração de Modelos Geométricos foi validado com diversos modelos geométricos obtidos à mão, dentre eles os dos robôs Puma, Stanford, K15 e K16 da Volkswagen, Esférico, Prismático, Cartesiano etc.

2.4. UM NOVO MÉTODO DE OBTENÇÃO PARA AS MATRIZES INTERMEDIÁRIAS

Apresentaremos neste item um novo método de obtenção para as matrizes intermediárias, que desenvolvemos com o objetivo de facilitar a identificação analítica dos casos de simplificação trigonométrica bem como a visualização da contraposição entre equações auxiliares e simplificações trigonométricas ou evidenciações.

Antes de apresentarmos o método, faremos um resumo de seus principais resultados:

- O método permite a obtenção de forma separada, das expressões da rotação e translação da matriz final.

- A constatação de que a problemática das simplificações trigonométricas está relacionada unicamente com a expressão da rotação global (matriz de rotação na matriz de transformação final).

- O método possibilitou a identificação dos casos em que aparecem simplificações trigonométricas (veja item 2.5., onde também são mostradas as expressões simplificadas), bem como a localização das simplificações na matriz de transformação final.

- A expressão de translação de robô K16 utilizado no exemplo 2.4.2., nos "indicou" um conjunto de sequências mais "naturais" para execução das multiplicações (que resultou número de operações menor do que o melhor caso encontrado pelo GMROB), sugerindo que o método leva a identificação da classe de sequências ótimas equivalentes.

- Quanto à contraposição das equações auxiliares com as simplificações trigonométricas, não podemos afirmar que esta última sempre traga diminuição no número de operações, embora nos casos observados a "economia" tenha sido grande. Esta contraposição efetivamente aparece: "a priori" temos que gerar nas opções com e sem simplificações trigonométricas, nos dois casos sempre utilizando equações auxiliares onde for possível.

- Quanto à contraposição das equações auxiliares com as evidenciações, observamos que esta inexiste. Para a mesma equação auxiliar a evidenciação deve ser feita sempre; para equações auxiliares diferentes, como já foi dito no item 2.2.4., o próprio uso das equações auxiliares constitui um processo de evidenciação implícita, sendo que para cada sequência, temos uma forma de evidenciação (no caso geral). No caso de cada robô, uma classe de sequências corresponderá a cada forma de evidenciação.

2.4.1. MÉTODO PARA OBTENÇÃO DOS MODELOS GEOMÉTRICOS INTERMEDIÁRIOS E FINAL

A busca do método foi iniciada com a verificação empírica de que a separação dos movimentos de rotação levava a uma melhor explicitação das expressões trigonométricas nas cadeias de caracteres resultantes, sendo que nas translações aparecia uma composição das expressões de rotação com os parâmetros de Denavit-Hartemberg, relativas a translação.

Esta observação, facilmente dedutível, nos sugeriu a utilização de uma partição nas matrizes elementares e intermediárias resgatando as informações parciais da matriz de transformação clássica (cf 2.2.1.) e do vetor de translação, sem sair da forma de matriz de transformação homogênea.

Também nos ocorreu tentar escrever as matrizes elementares como um produto de duas outras, separando os parâmetros de Denavit-Hartemberg de tal maneira que pudessemos vizualizar o seu efeito na matriz de transformação final. A definição destas duas matrizes está amarrada então à composição dos movimentos relativos a cada parâmetro e a um critério de complexidade, isto é, estas matrizes têm que ser as mais simples. A análise dos casos possíveis nos conduziu, forçosamente, à solução seguinte:

$$T_{i-1,\,i} = \left|\begin{array}{c|c} R\theta_i & 0 \\ \hline 0 & 1 \end{array}\right| \left|\begin{array}{c|c} R\alpha_i & P_i \\ \hline 0 & 1 \end{array}\right| = A(\theta)\ A(\alpha,\ a,\ r) \quad (2.42.)$$

onde:

$$R_{\theta i} = \begin{vmatrix} C_i & -S_i & 0 \\ S_i & C_i & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.43.)$$

$$R_{\alpha i} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & C_{\alpha i} & -S_{\alpha i} \\ 0 & S_{\alpha i} & C_{\alpha i} \end{vmatrix} \qquad (2.44.)$$

$$\text{e } P_i = \begin{vmatrix} a_i \\ 0 \\ r_i \end{vmatrix} \qquad (2.45.)$$

Observamos que os movimentos relativos a translação se agruparam num único vetor, e as informações de rotação estão cada uma numa matriz diferente.

Substituindo, por simplificação, as matrizes elementares $T_{i-1,i}$ por A_i, podemos escrever:

$$A_1A_2 = \begin{bmatrix} R\theta_1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R\alpha_1 & P_1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R\theta_2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R\alpha_2 & P_2 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} R\theta_1R\alpha_1R\theta_2 & R\theta_1P_1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R\alpha_2 & P_2 \\ 0 & 1 \end{bmatrix} \qquad (2.46.)$$

$A_1A_2A_3 = A' * \begin{bmatrix} R\alpha_3 & P_3 \end{bmatrix}$, onde A' é:

$$\begin{bmatrix} R\theta_1R\alpha_1R\theta_2R\alpha_2R\theta_3 & R\theta_1R\alpha_1R\theta_2P_2+R\theta_1P_1 \\ 0 & 1 \end{bmatrix} \qquad (2.47.)$$

Definindo:

$$R_i = \begin{cases} R\theta_1\left(\prod_{j=1}^{i-1}(R\alpha_jR\theta_{j+1})\right) \text{ para } i>1 \\ R\theta_1 \text{ para } i=1 \end{cases} \qquad (2.48.)$$

então:

$$T_{on} = \begin{bmatrix} R_n R_{an} & \sum_{i=1}^{n} (R_i P_i) \\ 0 & 1 \end{bmatrix} \qquad (2.49.)$$

Da mesma forma podemos definir uma matriz intermediária qualquer:

$$T_{mn} = \begin{bmatrix} R_{mn} R_{an} & \sum_{i=m+1}^{n} (R_{mi} P_i) \\ 0 & 1 \end{bmatrix} \qquad (2.50.)$$

onde:

$$R_{mi} = \begin{cases} R_{\theta m+1} \left(\prod_{j=m+1}^{i-1} (R_{\alpha j} R_{\theta j+1}) \right) & \text{para } i > m+1 \\ R_{\theta i} & \text{para } i = m+1 \end{cases} \qquad (2.51.)$$

2.4.2 - EXEMPLO DE APLICAÇÃO DO MÉTODO

O método acima foi utilizado para identificação analítica dos casos possíveis de simplificação trigonométrica e também para investigação da estrutura do modelo geométrico em relação às possíveis simplificações trigonométricas e aritméticas e utilização de equações auxiliares, que mostraremos a seguir através de um exemplo, utilizando os parâmetros de Denavit-Hartemberg do robô K16 (cf tabela (2.3.)).

Utilizando a expressão de Ton para n = 6:

$$T_{o6} = \left| \begin{array}{c|c} R_6 R_{a6} & \sum_{i=1}^{6} R_i P_i \\ \hline 0 & 1 \end{array} \right| \qquad (2.52.)$$

Onde $R_6 = R_{\theta 1} R_{\alpha 1} R_{\theta 2} R_{\alpha 2} R_{\theta 3} R_{\alpha 3} R_{\theta 4} R_{\alpha 4} R_{\theta 5} R_{\alpha 5} R_{\theta 6}$ (2.53.)

$$R_i = R_{\theta 1} R_{\alpha 1} \ldots\ldots\ldots R_{\alpha i-1} R_{\theta i} \qquad i > 1 \qquad (2.54.)$$

$$\text{e } R_1 = R_{\theta 1} \qquad (2.55.)$$

No caso do robô K16, existe simplificação trigonométrica entre os ângulos das ligações 2, 3 e 4 (confira no item seguinte que a condição de $\alpha_2 = 0$ e $\alpha_3 = 0$ resulta nesta simplificação).

A simplificação resultante se dá em $R_{14} = R_{\theta 2} R_{\alpha 2} R_{\theta 3} R_{\alpha 3} R_{\theta 4}$, da seguinte maneira:

$$R_{14} = R_{\theta 234} = \begin{vmatrix} C_{234} & -S_{234} & 0 \\ S_{234} & C_{234} & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.56.)$$

onde $C_{234} = \cos(\theta_2 + \theta_3 + \theta_4)$ (2.57.)

e $S_{234} = \text{sen}(\theta_2 + \theta_3 + \theta_4)$ (2.58.)

2.4.2.1. EXPRESSÕES DE ROTAÇÃO

Substituindo, R_{14} na expressão de R_6, temos:

$$R_6 = R_{\theta 1} R_{\alpha 1} R_{14} R_{\alpha 4} R_{\theta 5} R_{\alpha 5} R_{\theta 6} \qquad (2.59.)$$

Então a matriz de Rotação de T_{06} é:

$$R_6 R_{\alpha 6} = R_{01}(R_{14} R_{\alpha 4}) R_{45} R_{56} \qquad (2.60.)$$

2.4.2.2. EXPRESSÕES DE TRANSLAÇÃO:

O desenvolvimento do vetor de translação, resulta:

$$R_{\theta 1} P_1 + R_{\theta 1} R_{\alpha 1} R_{\theta 2} P_2 + R_{\theta 1} R_{\alpha 1} R_{\theta 2} R_{\alpha 2} R_{\theta 3} P_3 + R_{\theta 1} R_{\alpha 1} R_{\theta 2} R_{\alpha 3} R_{\theta 3} R_{\alpha 4} R_{\theta 4} P_4 +$$
$$+ R_{\theta 1} R_{\alpha 1} R_{\theta 2} R_{\alpha 2} R_{\theta 3} R_{\alpha 3} R_{\theta 4} R_{\alpha 4} R_{\theta 5} P_5 + R_{\theta 1} R_{\alpha 1} R_{\theta 2} R_{\alpha 2} R_{\theta 3} R_{\alpha 3} R_{\theta 4} R_{\alpha 4} R_{\theta 5} R_{\alpha 5} R_{\theta 6} P_6$$
(2.61.)

Para simplificar esta expressão, lançamos mão das simplificações de R_{14} (θ_2, θ_3 e θ_4) e R_{13} (θ_2 e θ_3), e da evidenciação (ou fatoração) dos termos repetidos, resultando:

$$R_{\theta_1}(R_{\alpha_1}(R_{14}(R_{\alpha_4}R_{\theta_5}(R_{\alpha_5}R_{\theta_6}P_6 + P_5) + P_4 + R_{13}P_3 + R_{\theta_2}P_2) + P_1) \quad (2.62.)$$

Como P_4 e P_6 são nulos, temos a expressão do vetor final de translação:

$$\sum_{i=1}^{6} R_i P_i = R_{\theta_1}(R_{\alpha_1}(R_{14}R_{\alpha_4}(R_{\theta_5}P_5) + R_{13}P_3 + R_{\theta_2}P_2) + P_1) \quad (2.63.)$$

Da expressão acima podemos deduzir o seguinte:

A) Todas as simplificações trigonométricas estão feitas e, com certeza, o seu uso traduz grande "economia". Obviamente, o uso de equações auxiliares é necessário onde não são usadas as simplificações trigonométricas.

B) O método nos fornece a localização exata das simplificações trigonométricas levando, no caso de decisão "a priori" do uso das simplificações, à determinação de um elenco mais restrito de possibilidades de sequências de multiplicação.

C) Apesar de serem decididas na matriz de rotação, as simplificações também aparecem nas expressões de translação, que são mais complexas, impedindo-nos de afirmar categoricamente que o uso de simplificações trigonométricas seja mais vantajoso.

D) A expressão de translação já mostra um "peso" diferenciado para as evidenciações, indicando que existe uma classe de sequências mais naturais para execução da multiplicação, pelo menos no que diz respeito à translação.

E) A expressão de rotação não mostra peso diferenciado para as evidenciações, indicando que um caminho a seguir em trabalhos futuros,

seria o de trabalhar com sequências distintas para rotação e translação, e estudar a composição dos efeitos no número de operações das expressões finais.

O item E) acima reforça a necessidade de construção de uma ferramenta mais genérica para investigação desta e de outras questões levantadas neste trabalho. Nesta ferramenta será fundamental a inclusão da partição proposta neste item pois acreditamos, embora sem prova analítica, que este método leva à identificação da classe de sequências ótimas equivalentes para um dado robô, pelo menos com respeito à translação.

2.4.2.3. OBTENÇÃO DO MODELO SEGUNDO UMA SEQUÊNCIA

Prosseguindo o cálculo do exemplo, adotando uma das sequências correspondentes à posição dos parentes na expressão de translação, chegamos a uma forma final com número de operações menor do que qualquer dos 5 métodos do GMROB com diferença percentual da ordem de 10%, nos mostrando que nosso caminho está correto.

O quadro (2.16.) mostra os passos para obtenção do modelo na seqüência escolhida.

	translação	rotação
1º		$R_{\theta 6} R_{\alpha 6} = M_A$
2º	$R_{\theta 5} P_5 = V_A$	$R_{\theta 5} R_{\alpha 5} M_A = M_B$
3º	$(R_{14} R_{\alpha 4}) V_A = V_B$	$R_{14} R_{\alpha 4} M_B = M_C$
4º	$R_{13} P_3 + V_B = V_C$	
5º	$R_{\theta 2} P_2 + V_C = V_D$	
6º	$R_{\alpha 1} V_D + P_1 = V_E$	$(R_{\theta 1} R_{\alpha 1}) M_C = M_D$
7º	$R_{\theta 1} V_E = V_F$	

Quadro (2.16.) - Passos para o Cálculo do Modelo

Observe que, no cálculo automático, os 3º, 4º e 5º passos para obtenção das expressões de translação, poderiam ser executados no mesmo cálculo de matriz intermediária (como é no GMROB).

Para cálculo da translação, montamos as matrizes:

$$R_{\theta 1} = \begin{vmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.64.)$$

$$R_{\theta 2} = \begin{vmatrix} C_2 & -S_2 & 0 \\ S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.65.)$$

$$R_{\theta 5} = \begin{vmatrix} C_5 & -S_5 & 0 \\ S_5 & C_5 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.66.)$$

$$R_{\theta 6} = \begin{vmatrix} C_6 & -S_6 & 0 \\ S_6 & C_6 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.67.)$$

$$R_{14} = R_{\theta 234} = \begin{vmatrix} C_{234} & -S_{234} & 0 \\ S_{234} & C_{234} & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.68.)$$

$$R_{13} = R_{\theta 23} = \begin{vmatrix} C_{23} & -S_{23} & 0 \\ S_{23} & C_{23} & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.69.)$$

$$R_{\theta 1} R_{\alpha 1} = \begin{vmatrix} C_1 & 0 & S_1 \\ S_1 & 0 & -C_1 \\ 0 & 1 & 0 \end{vmatrix} \qquad (2.70.)$$

$$R_{14} R_{\alpha 4} = \begin{vmatrix} C_{234} & 0 & S_{234} \\ S_{234} & 0 & -C_{234} \\ 0 & 1 & 0 \end{vmatrix} \qquad (2.71.)$$

$$R_{\theta 5} R_{\alpha 5} = \begin{vmatrix} C_5 & 0 & -S_5 \\ S_5 & 0 & C_5 \\ 0 & -1 & 0 \end{vmatrix} \qquad (2.72.)$$

$$R_{\theta 6} R_{\alpha 6} = \begin{vmatrix} C_6 & 0 & S_6 \\ S_6 & 0 & -C_6 \\ 0 & 1 & 0 \end{vmatrix} \qquad (2.73.)$$

$$P_1 = \begin{vmatrix} 0 \\ 0 \\ r_1 \end{vmatrix} \qquad (2.74.)$$

$$P_2 = \begin{vmatrix} a_2 \\ 0 \\ 0 \end{vmatrix} \qquad (2.75.)$$

$$P_3 = \begin{vmatrix} a_3 \\ 0 \\ 0 \end{vmatrix} \qquad (2.76.)$$

$$P_4 = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} \qquad (2.77.)$$

$$P_5 = \begin{vmatrix} 0 \\ 0 \\ r_5 \end{vmatrix} \qquad (2.78.)$$

$$P_6 = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} \qquad (2.79.)$$

Prosseguindo o cálculo, temos:

$$V_A = R_{\theta 5} P_5 = \begin{vmatrix} C_5 & -S_5 & 0 \\ S_5 & C_5 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0 \\ 0 \\ r_5 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ r_5 \end{vmatrix} \qquad (2.80.)$$

$$V_B = (R_{14} R_{\alpha 4})\, V_A = \begin{vmatrix} C_{234} & 0 & S_{234} \\ S_{234} & 0 & -C_{234} \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 \\ 0 \\ r_5 \end{vmatrix} = \begin{vmatrix} S_{234} r_5 \\ -C_{234} r_5 \\ 0 \end{vmatrix} \qquad (2.81.)$$

$$R_{13} P_3 = \begin{vmatrix} C_{23} & -S_{23} & 0 \\ S_{23} & C_{23} & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} a_3 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} C_{23} a_3 \\ S_{23} a_3 \\ 0 \end{vmatrix} \qquad (2.82.)$$

da mesma forma:

$$R_{\theta 2} P_2 = \begin{vmatrix} C_2 a_2 \\ S_2 a_2 \\ 0 \end{vmatrix} \qquad (2.83.)$$

$$V_D = V_B + R_{13} P_3 + R_{\theta 2} P_2 = \begin{vmatrix} S_{234} r_5 + C_{23} a_3 + C_2 a_2 \\ -C_{234} r_5 + S_{23} a_3 + S_2 a_2 \\ 0 \end{vmatrix} \qquad (2.84.)$$

$$V_E = R_{\alpha 1} V_D + P_1 =$$

$$= \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} S_{234} r_5 + C_{23} a_3 + C_2 a_2 \\ -C_{234} r_5 + S_{23} a_3 + S_2 a_2 \\ 0 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ r_1 \end{vmatrix} =$$

$$= \begin{vmatrix} S_{234} r_5 + C_{23} a_3 + C_2 a_2 \\ 0 \\ -C_{234} r_5 + S_{23} a_3 + S_3 a_2 + r_1 \end{vmatrix} \qquad (2.85.)$$

$$V_F = R_{\theta 1} V_E =$$

$$= \begin{vmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} S_{234} r_5 + C_{23} a_3 + C_2 a_2 \\ 0 \\ -C_{234} r_5 + S_{23} a_3 + S_2 a_2 + r_1 \end{vmatrix} =$$

$$= \begin{vmatrix} C_1 * (S_{234} r_5 + C_{23} a_3 + C_2 a_2) \\ S_1 * (S_{234} r_5 + C_{23} a_3 + C_2 a_2) \\ -C_{234} r_5 + S_{23} a_3 + S_2 a_2 + r_1 \end{vmatrix} \qquad (2.86.)$$

Com equações auxiliares:

$$D[1] = S_{234}\ r_5 \qquad (2.87.)$$

$$D[2] = -C_{234}\ r_5 \qquad (2.88.)$$

$$D[3] = C_{23}\ a_3 \qquad (2.89.)$$

$$D[4] = S_{23}\ a_3 \qquad (2.90.)$$

$$D[5] = C_2\ a_2 \qquad (2.91.)$$

$$D[6] = S_2\ a_2 \qquad (2.92.)$$

$$D[7] = D[1] + D[2] + D[3] + D[5] \quad (2.93.)$$

$$D[8] = D[2] + D[4] + D[6] \quad (2.94.)$$

$$D[9] = D[8] + r_1 \quad (2.95.)$$

$$D[10] = C_1 \; D[7] \quad (2.96.)$$

$$D[11] = S_1 \; D[7] \quad (2.97.)$$

$$\text{Translação} = \begin{vmatrix} D[10] \\ D[11] \\ D[9] \end{vmatrix} \quad (2.98.)$$

Para a rotação calculamos $R_{01}(R_{14}R_{04})R_{45}R_{56} =$

$$= \begin{vmatrix} C_1 & 0 & S_1 \\ S_1 & 0 & -C_1 \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} C_{234} & 0 & S_{234} \\ S_{234} & 0 & -C_{234} \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} C_5C_6 & -S_5 & C_5S_6 \\ S_5C_6 & C_5 & S_5S_6 \\ -S_6 & 0 & C_6 \end{vmatrix} =$$

$$= \begin{vmatrix} C_1 & 0 & S_1 \\ S_1 & 0 & -C_1 \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} C_{234}C_5C_6 - S_{234}S_6 & -C_{234}S_5 & C_{234}C_5S_6 + S_{234}C_6 \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}S_5 & S_{234}C_5S_6 - C_{234}C_6 \\ S_5C_6 & C_5 & S_5S_6 \end{vmatrix}$$

(2.99.)

Escrevendo as equações auxiliares relativas aos resultados parciais acima:

$$D[1] = C_5C_6 \quad (2.100.)$$

$$D[2] = C_5S_6 \quad (2.101.)$$

$$D[3] = S_5C_6 \quad (2.102.)$$

$$D[4] = S_5S_6 \quad (2.103.)$$

$$D[5] = C_{234} \; D[1] - S_{234} \; S_6 \quad (2.104.)$$

$$D[6] = -C_{234} \; S_5 \quad (2.105.)$$

$$D[7] = C_{234} \; D[2] + S_{234} \; C_6 \quad (2.106.)$$

$$D[8] = S_{234} \; D[1] + C_{234} \; S_6 \quad (2.107.)$$

$$D[10] = S_{234} \; D[2] - C_{234} \; C_6 \quad (2.108.)$$

Rotação =

$$= \begin{vmatrix} C_1 D[5] + S_1 D[3] & C_1 D[6] + S_1 C_5 & C_1 D[7] + S_1 D[4] \\ S_1 D[5] - C_1 D[3] & S_1 D[6] - C_1 C_5 & S_1 D[7] - C_1 D[4] \\ D[8] & D[9] & D[10] \end{vmatrix} \qquad (2.109.)$$

Os números de operações resultantes estão na tabela (2.6.). O quadro (2.17.) compara estes números com os obtidos por outros métodos.

avaliações	soma de ângulos	multiplicações	adições	
12	2	34	15	total
		8	5	translação
		26	10	rotação

Tabela (2.6.) - Número de Operações para o Cálculo do Modelo

Total de operações aritméticas:	51
Para o melhor método do GMROB (método 5):	57
Para o melhor método de [MEGAHED 84]:	80

Quadro (2.17.) - Comparação entre Vários Métodos

Vejamos o cálculo para uma outra seqüência qualquer na rotação, ex: $((R_{\theta 1} R_{\alpha 1})(R_{14} R_{\alpha 4}))((R_{\theta 5} R_{\alpha 6})(R_{\theta 6} R_{\alpha 6}))$:

Rotação:

$$\begin{vmatrix} C_1 & 0 & S_1 \\ S_1 & 0 & -C_1 \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} C_{234} & 0 & S_{234} \\ S_{234} & 0 & -C_{234} \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} C_5 & 0 & -S_5 \\ S_5 & 0 & C_5 \\ 0 & -1 & 0 \end{vmatrix} \begin{vmatrix} C_6 & 0 & S_6 \\ S_6 & 0 & -C_6 \\ 0 & 1 & 0 \end{vmatrix} =$$

$$= \begin{vmatrix} C_1 C_{234} & S_1 & C_1 S_{234} \\ S_1 C_{234} & -C_1 & S_1 S_{234} \\ S_{234} & 0 & -C_{234} \end{vmatrix} \begin{vmatrix} C_5 C_6 & -S_5 & C_5 S_6 \\ S_5 C_6 & C_5 & S_5 S_6 \\ -S_6 & 0 & C_6 \end{vmatrix} \qquad (2.110.)$$

Substituindo por equações auxiliares:

$$D[1] = C_1 C_{234} \qquad (2.111.)$$

$$D[2] = C_1 S_{234} \qquad (2.112.)$$

$$D[3] = S_1 C_{234} \qquad (2.113.)$$

$$D[4] = S_1 S_{234} \qquad (2.114.)$$

$$D[5] = C_5 C_6 \qquad (2.115.)$$

$$D[6] = C_5 S_6 \qquad (2.116.)$$

$$D[7] = S_5 C_6 \qquad (2.117.)$$

$$D[8] = S_5 S_6 \qquad (2.118.)$$

Fazendo o produto da equação (2.110.):

$$\left.\begin{aligned} D[9] &= D[1]\,D[5] + S_1\,D[7] - D[2]\,S_6 \\ D[10] &= -D[1]\,S_5 + S_1\,C_5 \\ D[11] &= D[1]\,D[6] + S_1\,D[8] + D[2]\,C_6 \end{aligned}\right\} \text{1ª linha} \qquad (2.119.)$$

$$\left.\begin{aligned} D[12] &= S_{234}\,D[5] + C_{234}\,S_6 \\ D[13] &= -S_{234}\,S_5 \\ D[14] &= S_{234}\,D[6] - C_{234}\,C_6 \end{aligned}\right\} \text{2ª linha} \qquad (2.120.)$$

Obtendo a terceira linha através de produto vetorial da primeira pela segunda:

$$\left.\begin{aligned} D[15] &= D[10]\,D[14] - D[11]\,D[13] \\ D[16] &= D[11]\,D[12] - D[9]\,D[14] \\ D[17] &= D[9]\,D[13] - D[10]\,D[12] \end{aligned}\right\} \text{3ª linha} \qquad (2.121.)$$

Os números de operações resultantes estão na tabela (2.7.). A comparação destes números com os obtidos por outros métodos estão no quadro (2.18.).

avaliações	soma de ângulos	multiplicações	adições	
12	2	35	15	total
		8	5	translação
		27	10	rotação

Tabela (2.7.) - Número de Operações para o Cálculo do Modelo

Total de operações aritméticas:	52
Para o melhor método do GMROB (método 5):	57
Para o melhor método de [MEGAHED 84]:	80

Quadro (2.18.) - Comparação entre Vários Métodos

2.5. SIMPLIFICAÇÕES TRIGONOMÉTRICAS

O objetivo deste item é a identificação de um elenco de situações onde aparecem simplificações trigonométricas, umas fáceis de visualizar, outras menos, e para cada situação qual a simplificação a ser feita.

A utilização do método do item anterior foi muito importante pois observamos que, para identificar a ocorrência de simplificações entre duas matrizes (m+1) e n, só precisamos estudar a composição da R_{mn}, ou seja, nem R_{an} nem as expressões de translação são necessárias.

2.5.1. SITUAÇÕES ONDE OCORREM SIMPLIFICAÇÕES TRIGONOMÉTRICAS

Para auxílio às análises, suponhamos uma orientação qualquer R_q entre R_{ei} e R_{ej}, onde R_q é representada por uma matriz de orientação genérica da forma:

$$R_q = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \qquad (2.122.)$$

Neste caso, analisando os elementos de $R_{ei}R_qR_{ej}$, chegamos a:

Elemento (1,1):

$$C_iC_jn_x - S_iC_jn_y + C_iS_jo_x - S_iS_jo_y \qquad (2.123.)$$

Há simplificação trigonométrica (doravante denominada ST) parcial quando $n_x = \pm o_y$ (1ª condição) ou $o_x = \pm n_y$ (2ª condição). ST total aparece quando as duas condições são satisfeitas.

Outros elementos da matriz: caem nas mesmas condições.

2.5.1.1. CASO DE DUAS LIGAÇÕES ROTACIONAIS CONSECUTIVAS

Na expressão (2.123.), se fizermos i e j duas ligações seguidas então $R_q = R_{ai}$, e temos da 1ª condição que $C_{ai} = \pm 1$, e a 2ª é sempre satisfeita (multiplicação por zero, analisada automaticamente em GMROB). Então a simplificação total existe sempre que a 1ª condição é

satisfeita, ou seja, $C\alpha_i = \pm 1$. A condição de ST resulta:

$$\alpha_i = K\pi \quad (K \text{ é um inteiro}) \qquad (2.124.)$$

2.5.1.2. CASO DE DUAS LIGAÇÕES ROTACIONAIS INTERCALADAS POR UMA TAMBÉM ROTACIONAL

Para análise em duas juntas i e j, intercaladas por uma junta k rotacional, fazemos $R_q = R_{\alpha i} R_{\theta k} R_{\alpha k}$. A 1ª condição significa $C\alpha_i C\alpha_k = \pm 1$ e $S\alpha_i S\alpha_k = 0$. A 2ª condição significa $C\alpha_k = \pm C\alpha_i$. Daí:

$$\text{ST total: } \alpha_i = K_i\pi \text{ e } \alpha_k = K_k\pi \qquad (2.125.)$$

$$\text{ST parcial: } \alpha_k = \pm(\alpha_i + K\pi) \qquad (2.126.)$$

2.5.1.3. CASO DE DUAS LIGAÇÕES ROTACIONAIS INTERCALADAS POR UMA PRISMÁTICA

A ST em duas juntas i e j, intercaladas por outra prismática k, cai no mesmo caso de intercalação por junta rotacional com $C_k = 1$ e $S_k = 0$, se $R_{\theta k} = I$ (matriz identidade). Neste caso, da 1ª condição: $C\alpha_i C\alpha_k - S\alpha_i S\alpha_k = \pm 1$ (ou $C\alpha_{ik} = \cos(\alpha_i + \alpha_k) = \pm 1$), sendo a 2ª sempre satisfeita, resultando:

$$\text{ST total: } \alpha_i + \alpha_k = K\pi. \qquad (2.127.)$$

2.5.1.4. CASO DE DUAS LIGAÇÕES ROTACIONAIS INTERCALADAS POR OUTRAS DUAS ROTACIONAIS

A ST em duas juntas i e 1, intercaladas por duas rotacionais j e k pode ser obtida por análise de $R_{\theta i} R_{\alpha i} R_{\theta j} R_{\alpha j} R_{\theta k} R_{\alpha k} R_{\theta 1}$, fazendo substituições sucessivas de $R_{q1} = R_{\alpha j}$ em $R_{q2} = R_{\theta j} R_{\alpha j} R_{\theta k} = R_{\theta j} R_{q1} R_{\theta k}$, de R_{q2} em $R_{q3} = R_{\alpha i} R_{\theta j} R_{q1} R_{\theta k} R_{\alpha k} = R_{\alpha i} R_{q2} R_{\alpha k}$, e finalmente de R_{q3} em $R_{\theta i} R_{\alpha i} R_{\theta j} R_{\alpha j} R_{\theta k} R_{\alpha k} R_{\theta 1} = R_{\theta i} R_{q3} R_{\theta 1}$.

Da 1ª condição de ST, tiramos: $C\alpha_i C\alpha_j C\alpha_k = \pm 1$, $C\alpha_j = \pm C\alpha_i C\alpha_k$, $S\alpha_i S\alpha_j C\alpha_k = 0$, $C\alpha_i S\alpha_j S\alpha_k = 0$ e $S\alpha_i C\alpha_j S\alpha_k = 0$; daí chegamos a $\alpha_i = K_i\pi$ e $\alpha_j = K_j\pi$ e $\alpha_k = K_k\pi$. Da 2ª condição tiramos $C\alpha_i = \pm C\alpha_j C\alpha_k$, $C\alpha_i C\alpha_j = \pm C\alpha_k$, $S\alpha_i S\alpha_j = 0$ e $S\alpha_j S\alpha_k = 0$; daí chegamos a $\alpha_i = K_i\pi$ e $\alpha_j = K_j\pi$ e α_k

$= K_k\pi$, que é a mesma da 1ª. Portanto:

$$\text{ST total: } \alpha_i = K_i\pi \text{ e } \alpha_j = K_j\pi \text{ e } \alpha_k = K_k\pi. \qquad (2.128.)$$

Observe que neste caso não temos somente simplificação entre as juntas i e l, mas também entre as juntas intermediárias. Conclusão: se tivermos condições de ST para duas juntas intercaladas de duas rotacionais então teremos condições para ST entre as intermediárias.

2.5.1.5. CASO DE DUAS LIGAÇÕES ROTACIONAIS INTERCALADAS POR DUAS PRISMÁTICAS

No caso de duas juntas rotacionais intercaladas por duas prismáticas, podemos utilizar o resultado obtido para quatro juntas rotacionais i, j, k, l. Se $R\theta_j = R\theta_k = I$, então a 1ª condição sempre se verifica e a 2ª ($C\alpha_{ijk} = \pm 1$) é condição para ST total.

$$\text{ST total: } \alpha_i + \alpha_j + \alpha_k = K\pi \qquad (2.129.)$$

2.5.2. EXECUÇÃO DAS SIMPLIFICAÇÕES

Até aqui analisamos quais as condições para que apareçam as ST, porém não mostramos como ficariam os elementos das matrizes resultantes nos casos anteriores. Com o auxílio do GMROB, sem a utilização do método 5 (que tem as ST mais usuais), foram estudados alguns casos (veja no Anexo 2) cujos principais resultados estão mostrados a seguir:

Neste ponto é importante frisar que estamos trabalhando com as expressões de rotação. Para emprego das ST no modelo direto, têm que ser analisadas as expressões de translação para se decidir onde e quais simplificações utilizar.

2.5.2.1. CASO DE QUATRO LIGAÇÕES ROTACIONAIS SEGUIDAS

No caso de quatro juntas rotacionais (1, 2, 3 e 4), para $\alpha_1 = \alpha_2 = \alpha_3 = 0$, chegamos a $R_4\alpha_4 = R\theta_{1234}R\alpha_4$. Num outro caso em que $\alpha_1 = \alpha_3 = \pi$ e $\alpha_2 = 0$, chegamos a $R_4R\alpha_4 = R\theta_{1-2-34}R\alpha_{1234}$ ($\theta_{1-2-34} = \theta_1-\theta_2-\theta_3+\theta_4$).

Para quaisquer valores de α_1, α_2 e α_3, desde que obedecidas as condições para ST, temos a expressão geral:

$$R_{14}R_{\alpha 4} = R_{\theta\varepsilon 4}R_{\alpha 1234}, \text{ onde } \theta_{\varepsilon 4} = \theta_1 + C_{\alpha 1}\theta_2 + C_{\alpha 1}C_{\alpha 2}\theta_3 + C_{\alpha 1}C_{\alpha 2}C_{\alpha 3}\theta_4 \qquad (2.130.)$$

Há uma forma de representação mais conveniente do que esta, pois isola o $R_{\alpha 4}$, sendo mais adequada à obtenção do modelo dentro da proposição do item 2.4.. Esta forma foi originada da observação de que o valor de K da expressão $\sum\alpha_i = K_i$ resultando número ímpar, significava que $R_{\theta\varepsilon 4}R_{\alpha 1234}$ podia ser obtido com a matriz $R_{\bar{\theta}\varepsilon 4}$ ($R_{\theta\varepsilon 4}$ com o sinal das duas últimas colunas modificado), multiplicada por $R_{\alpha 4}$. Se o valor de K resultasse par, o valor da expressão seria $R_{\theta\varepsilon 4}R_{\alpha 4}$. Isolado o $R_{\alpha 4}$, temos a expressão:

$$R_i = \begin{matrix} R_{\theta\varepsilon i} \text{ se } \sum \alpha_i \text{ for par} \\ R_{\bar{\theta}\varepsilon i} \text{ se } \sum \alpha_i \text{ for ímpar} \end{matrix} \qquad (2.131.)$$

2.5.2.2. CASO DE DUAS LIGAÇÕES ROTACIONAIS SEGUIDAS

Voltando ao caso de duas juntas rotacionais seguidas, $\alpha_i = K\pi$ é o caso de ST. Obtém-se portanto o mesmo resultado que no item anterior.

2.5.2.3. CASO DE TRÊS LIGAÇÕES ROTACIONAIS SEGUIDAS

Para o caso de três juntas rotacionais seguidas temos dois casos: o primeiro se refere à simplificação total, que obviamente obedece à mesma lei de formação acima. Porém o caso de simplificação parcial ($C_{\alpha k} = \pm C_{\alpha i}$) mereceu uma análise em separado com utilização do GMROB. Dos resultados sem equações auxiliares obtivemos 8 multiplicações e 4 somas/subtrações a menos, porém com 2 avaliações de senos/cossenos a mais. Para o caso com equações auxiliares o resultado foi 2 somas/subtrações a mais, duas avaliações de senos/cossenos a mais e ainda 2 equações a mais, ou seja para o caso de uso de equações auxiliares a ST parcial não vale a pena em nenhum caso.

2.5.2.4. CASO DE DUAS LIGAÇÕES ROTACIONAIS INTERCALADAS POR UMA PRISMÁTICA

Quando há ST em duas juntas rotacionais intercaladas por uma prismática (i, k e j, nesta ordem) para o caso em que $R_{\theta k} = I$, o produto $R_{\theta i}R_{\alpha i}R_{\alpha k}R_{\theta j}$ é igual a $R_{\theta i}R_{\alpha ik}R_{\theta j}$, ou seja, os termos são idênticos aos termos do produto sem considerar a intercalação e considerando não mais os ângulos α_i e α_k separadamente, porém a soma deles.

2.6. AS MATRIZES INTERMEDIÁRIAS ÓTIMAS E O CÁLCULO DO MODELO DINÂMICO

Nos itens anteriores tratamos da obtenção do modelo geométrico e dos modelos geométricos intermediários e de sua otimização, explicitando todo o leque de opções da literatura e propondo novas soluções.

Neste item vamos sintetizar os passos para obtenção dos modelos ótimos, dentro dos critérios estabelecidos no item 2.2, e discutir a aplicação destes modelos na otimização do modelo dinâmico.

Nós consideramos imprescindível para a obtenção de modelos ótimos, que se disponha de uma ferramenta apropriada para geração automática de modelos. Recomendamos, para trabalhos futuros, que seja construída uma ferramenta para levantamento sistemático de todas as alternativas, de cada caso em estudo. Esta deverá permitir as seguintes opções aos usuários:

- Geração em todas as sequências, sempre com opção de uso de equações auxiliares, realizando todas as simplificações. aritméticas dentro de uma mesma equação auxiliar.

- Geração com e sem simplificações trigonométricas com uso de equações auxiliares onde for possível.

- Obtenção diferenciada das expressões de translação e rotação.

- Uso ou não de propriedades de bases ortonormais.

- Construção diferenciada para matrizes elementares, de acordo com a convenção adotada: [DENAVIT 85] ou [CRAIG 85]; e possibilidade de construção de matrizes elementares segundo outras convenções a quatro parâmetros.

Esta ferramenta, além de busca do modelo ótimo para cada robô, terá função de subsidiar:

- Pesquisa sobre otimização dos modelos considerando variações de origens, posicionamento de eixos e convenção tipo 4 parâmetros.

- Pesquisa sistemática das alternativas para construção de um sistema baseado em conhecimento para redução do elenco de alternativas a serem pesquisadas, para um determinado tipo de robô.

- Validação dos algorítmos de busca em árvore que garantam o caminho da otimalidade, que venham a ser desenvolvidos.

2.6.1. AS MATRIZES INTERMEDIÁRIAS ÓTIMAS NO CÁLCULO DO MODELO DINÂMICO

A geometria intervem no modelo dinâmico (formalismo de Lagrange), onde são utilizadas, além da matriz de transformação final, as matrizes de transformação intermediárias.

Poder-se-ia pensar que a otimização do Modelo Dinâmico com respeito às expressões do modelo geométrico seria obtida utilizando-se todas as matrizes intermediárias ótimas. Porém, esta afirmação não é verdadeira pois existe uma contraposição entre uso das matrizes intermediárias ótimas e o aproveitamento das equações auxiliares de uma matriz por outra, quando estas são geradas por sequências diferentes.

Isto se deve ao fato de que na obtenção de uma matriz ótima de ordem maior, a sua seqüência de geração pode não passar pelo caminho ótimo da de ordem menor. No caso de serem geradas por sequências diferentes, não se poderia, "a priori", aproveitar as equações auxiliares de uma matriz intermediária para outra de ordem maior.

Com relação a esta questão, nós acreditamos com muita convicção, embora sem prova analítica, que a "economia" gerada pelo aproveitamento das equações auxiliares e uso da matriz ótima quando a contraposição não existir, é maior que a gerada pelo uso de todas as matrizes ótimas.

2.6.2. EXEMPLO

Suponhamos que os parêntes de $(((T_{01}\ T_{12})(T_{23}\ T_{34}))\ T_{45})$ representem a sequência ótima de multiplicação das cinco matrizes, sendo este um caso onde não aparecerão simplificações trigonométricas e não foi utilizado o recurso de produto vetorial.

Então, no caso dos modelos geométricos intermediários, utilizaremos esta sequência para gerar o que for possível, otimizando o restante com esta restrição.

No caso, seja o conjunto completo dos modelos geométricos intermediários (definindo $A_i = T_{i-1,i}$ por simplificação):

A_1	(A_1A_2)	$(A_1A_2)A_3$	$((A_1A_2)(A_3A_4))$	$(((A_1A_2)(A_3A_4))A_5)$
	A_2	A_2A_3	$A_2(A_3A_4)$	$A_2(A_3A_4)A_5$
		A_3	(A_3A_4)	$(A_3A_4)A_5$
			A_4	A_4A_5
				A_5

Quadro (2.19.) - Conjunto de Modelos Geométricos Intermediários

Os produtos entre parênteses devem constar como equações auxiliares comuns aos modelos intermediários e o final.

CAPÍTULO 3

COORDENAÇÃO DE MOVIMENTOS - DESENVOLVIMENTO DE UMA FERRAMENTA PARA AUXÍLIO À OBTENÇÃO DO MODELO INVERSO

3.1. INTRODUÇÃO

O objetivo deste capítulo é apresentar o problema da inversão do modelo geométrico, as soluções existentes na literatura, em especial a inversão analítica, apresentar uma ferramenta desenvolvida (SIS_INV) para auxiliar a inversão por este método, propor a extensão desta ferramenta, e tratar da validação dos modelos inversos obtidos desta forma.

Sejam "$\underline{x}$" o vetor de coordenadas operacionais (posição e orientação) e "$\underline{q}$" o vetor de coordenadas generalizadas (ângulos e deslocamentos).

Para um movimento "$\underline{x}(t)$", relativo a uma tarefa definida no espaço operacional, um movimento correspondente "$\underline{q}(t)$" no espaço das coordenadas generalizadas é obtido pela inversão do modelo geométrico:

$$\underline{q}(t) = f^{-1}(\underline{x}(t)) \qquad (3.1.)$$

No controle de robôs manipuladores o modelo geométrico inverso serve para fornecer uma referência no mesmo espaço onde estão os acionadores, ou seja, no espaço das coordenadas generalizadas.

A obtenção de uma solução $\underline{q}(t)$ é denominada "coordenação de movimentos".

O cálculo do modelo inverso em tempo real pode ser feito através de expressões analíticas, ou através de métodos numéricos.

No caso da obtenção analítica, como "f" é não linear, não podemos garantir a existência e/ou a unicidade de uma função inversa "f^{-1}". No caso geral, só se pode determinar o número máximo de prováveis soluções ([GORLA 84]). Mas, uma vez obtidas ([PAUL 81]) e validadas as expressões analíticas, o seu cálculo é simples e rápido, sendo possível também a fácil localização e tratamento de regiões próximas a pontos de singularidade.

No caso de métodos numéricos, a obtenção das coordenadas generalizadas é em geral complexo, demandando recursos sofisticados para a sua execução em tempo real, além de problemas como a falta de garantia de convergência e a não explicitação de todas as soluções (converge para uma única solução).

Existe uma outra forma de realizar a coordenção de movimentos: a aprendizagem, onde o operador conduz o órgão terminal e os correspondentes pontos de referência são registrados pelos sensores, no nível das ligações. Neste caso uma solução inversa é fornecida implicitamente.

Porém, quando as referências relativas a uma determinada tarefa são geradas em tempo real por um sistema de visão, ou quando é necessária a programação "off-line" do robô (programação textual) devido à complexidade ou precisão inerentes à tarefa, o processo de aprendizagem perde o sentido, sendo necessária uma coordenação de movimentos adequada via modelagem.

3.2. O PROBLEMA DA INVERSÃO DO MODELO GEOMÉTRICO

A função "f" que compõe os movimentos das ligações para resultar no movimento do órgão terminal (modelo geométrico) é não linear, composta de somas de produtos das coordenadas generalizadas prismáticas e senos e cossenos das coordenadas generalizadas rotacionais. Por isto sua inversão não é trivial, envolvendo uma série de dificuldades, como por exemplo: existência e número de soluções, generalidade dos métodos, singularidades, redundâncias, escolha da representação do modelo direto mais apropriada à inversão (quando a inversão é analítica), e cálculo do modelo em tempo real (quando a inversão é numérica).

Para ilustrarmos a complexidade das expressões do modelo geométrico, usaremos as expressões da matriz de transformação final do robô TH8 em termos das variáveis de juntas. A representação utilizada está na figura (3.1.) e os parâmetros de Denavit-Hartemberg na tabela

(3.1.).

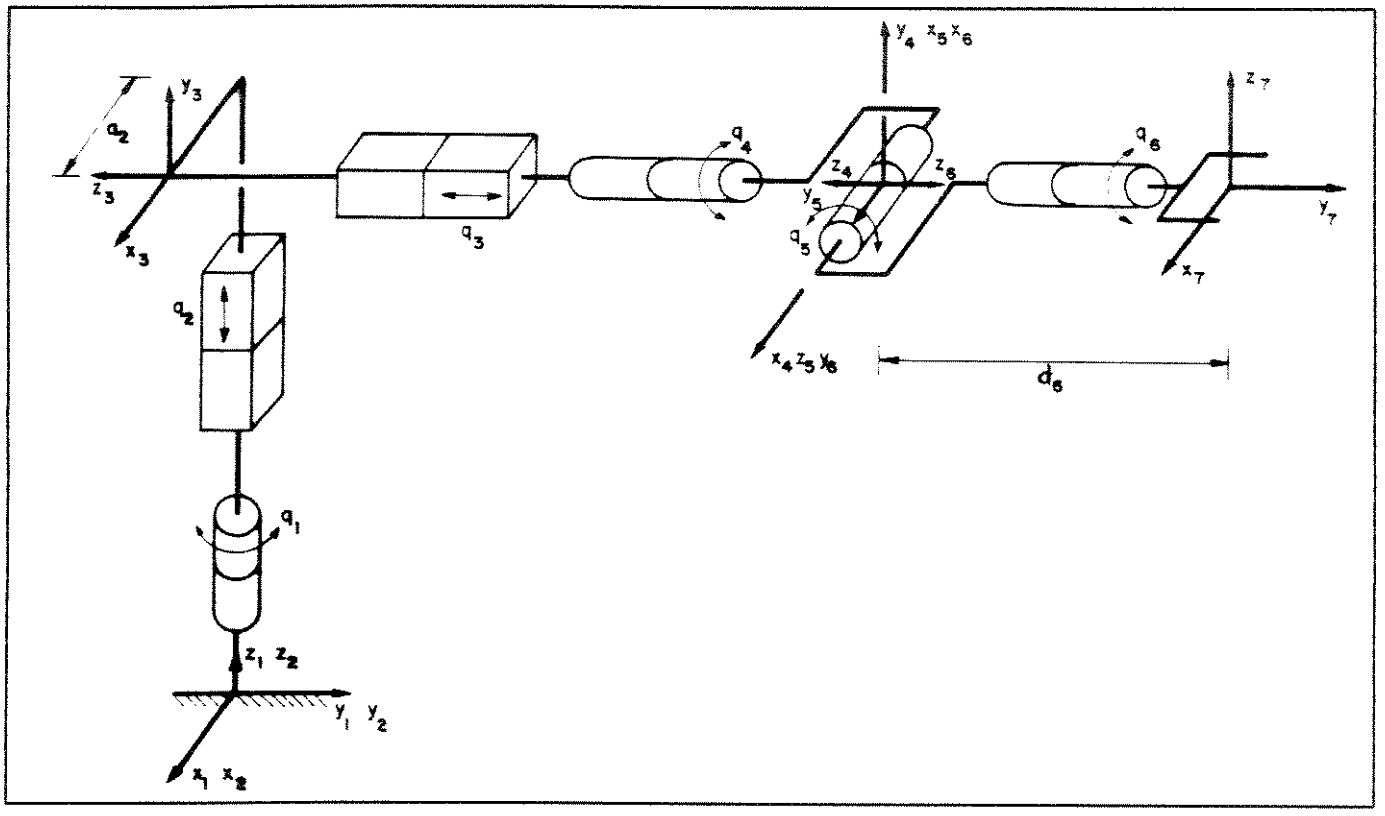


Figura (3.1.) - Representação do Robô TH8 [MEGAHED 84]

junta	tipo	θ	α	a	d
1	R	θ_1	0	0	0
2	P	0	90	a2	d2
3	P	0	0	0	d3
4	R	θ_4	90	0	0
5	R	θ_5	90	0	0
6	R	θ_6	90	0	d6

Tabela (3.1.) - Parâmetros de Denavit-Hartemberg do robô TH8

As expressões ((3.3.) a (3.14.)) da matriz de transformação final do TH8 ([To6]) foram obtidas através do SIS_MGD, sem a utilização de equações auxiliares (para visualização de todas as variáveis).

$$[To6] = \begin{bmatrix} nx & ax & ox & px \\ ny & ay & oy & py \\ nz & az & oz & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.2.)$$

onde:

$nx = (+(+C_1*C_4*C_5+S_1*S_5)*C_6+C_1*S_4*S_6);$ (3.3.)

$ox = (+C_1*C_4*S_5-S_1*C_5);$ (3.4.)

$ax = (+(+C_1*C_4*C_5+S_1*S_5)*S_6-C_1*S_4*C_6);$ (3.5.)

$px = (+(+C_1*C_4*S_5-S_1*C_5)*d_6+(+S_1*D_3+C_1*a_2));$ (3.6.)

$ny = (+(+S_1*C_4*C_5-C_1*S_5)*C_6+S_1*S_4*S_6);$ (3.7.)

$oy = (+S_1*C_4*S_5+C_1*C_5);$ (3.8.)

$ay = (+(+S_1*C_4*C_5-C_1*S_5)*S_6-S_1*S_4*C_6);$ (3.9.)

$py = (+(+S_1*C_4*S_5+C_1*C_5)*d_6+(-C_1*D_3+S_1*a_2));$ (3.10.)

$nz = (+S_4*C_5*C_6-C_4*S_6);$ (3.11.)

$oz = S_4*S_5;$ (3.12.)

$az = (+S_4*C_5*S_6+C_4*C_6);$ (3.13.)

$pz = (+S_4*S_5*d_6+D_2).$ (3.14.)

Como podemos ver por estas expressões, não é trivial a explicitação de $\underline{q} = [\theta_1, d_2, d_3, \theta_4, \theta_5, \theta_6]$. Veremos no item 3.3.1. que existem maneiras de escrever as identidades acima de uma forma mais conveniente para a explicitação de "$\underline{q}$". Veremos também que a explicitação pode ser feita de diversas maneiras, sendo algumas formas de utilização inconvenientes (ex: arco-cosseno e arco-seno) por problemas numéricos e de indefinição de quadrantes para os ângulos.

3.2.1. SOLUÇÕES MÚLTIPLAS E RESOLUBILIDADE

Um outro problema encontrado na coordenação de movimentos é o das soluções múltiplas, ou seja, o fato de que podem existir várias configurações do robô resultando na mesma posição e orientação do órgão terminal.

A figura (3.2.) mostra um manipulador planar com três graus de liberdade, em uma determinada posição e orientação. A linha tracejada mostra uma segunda configuração que fornece as mesmas posição e orien-

tação.

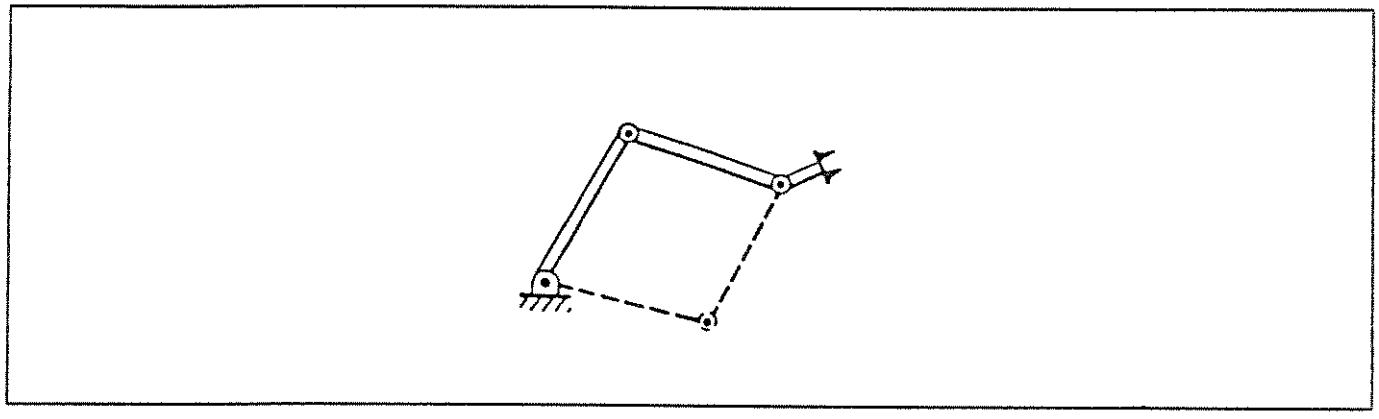


Figura (3.2.) - Manipulador Planar com Três Graus de Liberdade

O fato do manipulador ter múltiplas soluções pode causar problemas porque o sistema tem que ser capaz de escolher uma delas. O critério de escolha pode variar, porém uma escolha razoável é a da solução mais próxima, isto é, minimizar a amplitude de deslocamento das diversas juntas.

Na figura (3.3.) o manipulador está na posição "A". Para atingir a posição e orientação determinada em "B", a melhor escolha seria a solução tracejada superior, se não houvesse o obstáculo.

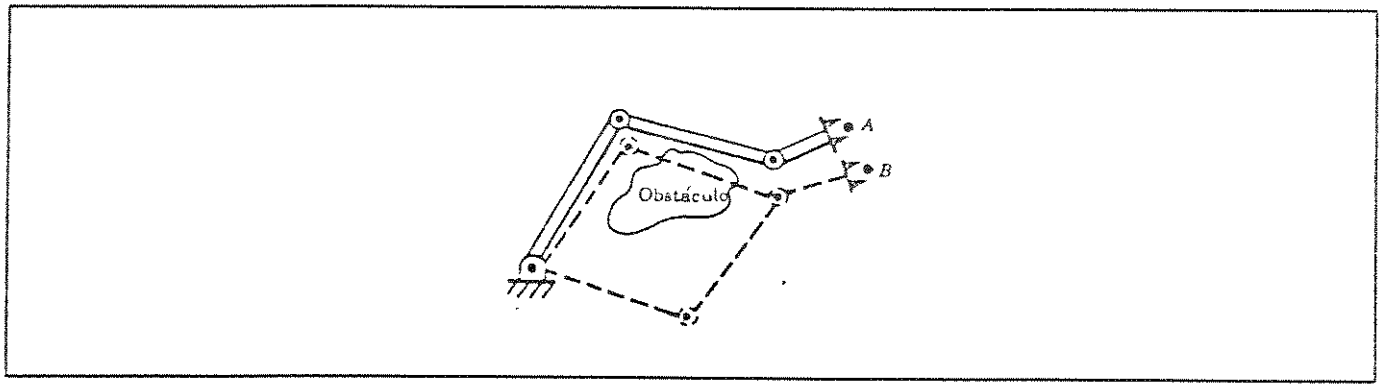


Figura (3.3.) - Manipulador Planar com Obstáculo

Isto sugere que um dado relevante para a rotina de inversão deve ser a configuração atual do manipulador. Neste caso, se há múltiplas soluções, o algoritmo pode escolher a mais próxima.

No entanto o conceito de proximidade deve ser bem definido. Por exemplo, alguns robôs podem ter elementos grandes seguidos de três menores de orientação. Neste caso, pesos devem ser atribuídos a cada ligação para a definição de proximidade, de tal forma que a escolha favoreça o movimento dos elementos menores.

No âmbito da III^{a} EBAI, sob nossa orientação, foi desenvolvido um programa computacional para avaliação de gastos "energéticos" de uma trajetória no espaço operacional (definida pelo usuário), ao se adotar cada uma das soluções de configuração do robô TH8. Neste programa, o usuário define também os pesos relativos a cada ligação do robô (O relatório deste trabalho está no Anexo 3).

Quanto à resolubilidade, um robô é resolúvel se é possível determinar todas as configurações "q" correspondentes a uma dada situação "x". Segundo [GORLA 84], são resolúveis os robôs manipuladores com seis graus de liberdade que possuem ao menos duas ligações coaxiais, (P) e (P) ou (R) e (P), ou dois pares de ligação (R) concorrentes, ou ainda três ligações (R) concorrentes. Os robôs com menos de seis graus de liberdade são todos resolúveis.

Quanto ao número de soluções, se "x" pertence ao volume de trabalho do robô e o número de ligações é inferior ao número de graus de liberdade da tarefa, então não existe solução; se o número de ligações é igual ao número de graus de liberdade da tarefa, então existe número finito de soluções para as configurações não singulares, e número infinito de soluções para as configurações singulares; se o número de ligações é maior que o número de graus de liberdade da tarefa, então existe número infinito de soluções (redundância ou degenerescência).

Uma tarefa qualquer, definida no espaço operacional de um robô, exige no mínimo seis graus de liberdade para sua execução. Contudo, se existirem obstáculos a contornar, o número de graus de liberdade necessários pode aumentar.

Ainda com respeito ao número de soluções, um mesmo robô pode ter número de soluções diversos para duas situações do órgão terminal

distintas, devido aos limites de juntas. As expressões analíticas do modelo inverso não mostram isto, por isto devemos sempre incluir no algoritmo de inversão a eliminação das soluções fora dos limites das juntas.

Na figura (3.4.), são mostradas quatro configurações possíveis do robô PUMA para uma determinada situação do órgão terminal.

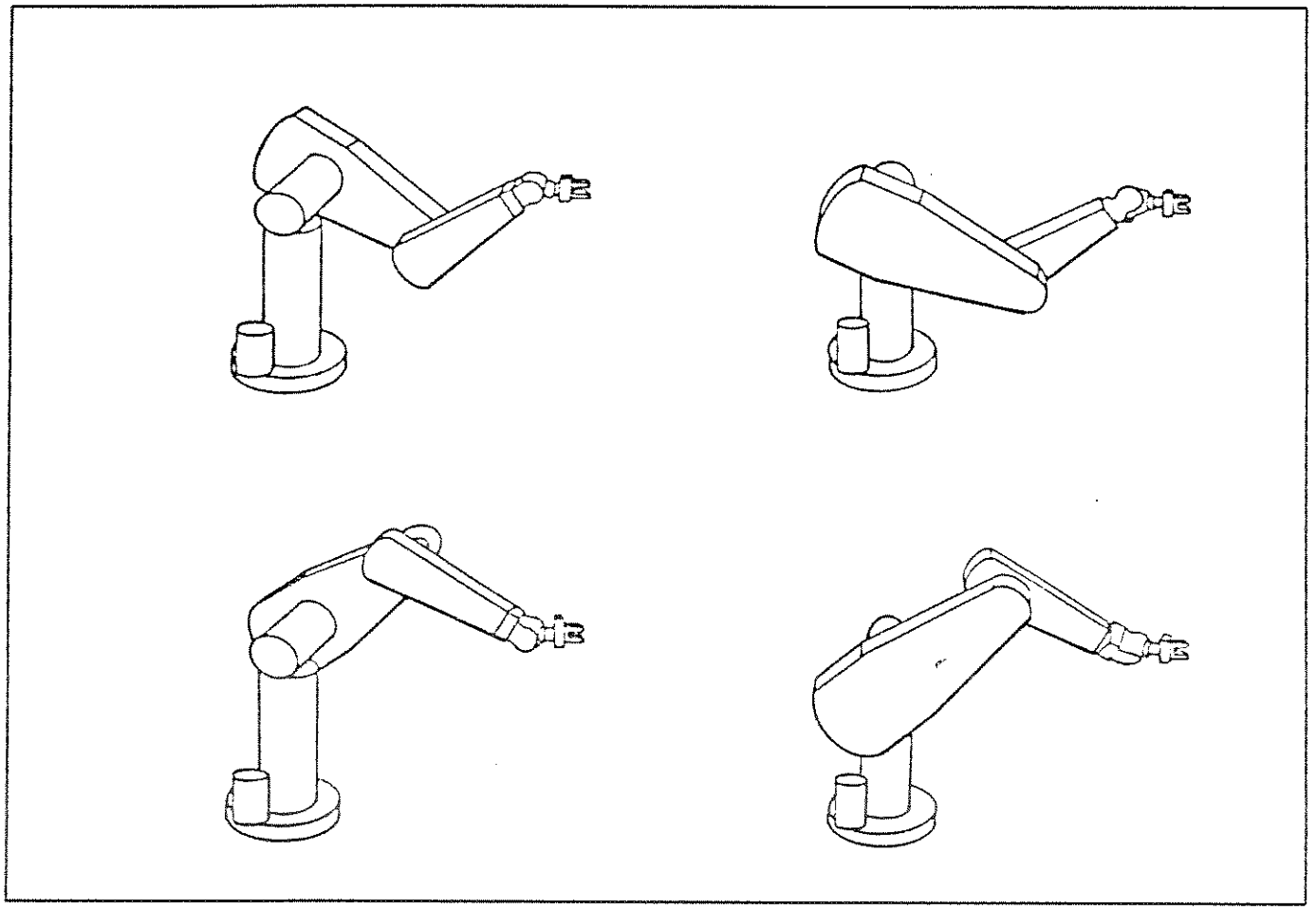


Figura (3.4.) - Configurações Possíveis do Robô PUMA

3.2.2. SINGULARIDADE E DEGENERESCÊNCIA (OU REDUNDÂNCIA)

Mais complexos que os problemas de soluções múltiplas são os problemas de degenerescência e singularidade.

Degenerescência ocorre quando o robô tem mais graus de liberdade

do que exige a tarefa, o que fornece infinitas soluções para cada situação (alguns autores chamam isto de degenerescência infinita, chamando simplesmente degenerescência ao que chamamos soluções múltiplas). Neste caso utilizam-se outras técnicas tais como minimização da energia cinética, para escolha de uma configuração. É claro que sempre se pode fixar uma junta, reduzindo o número de graus de liberdade do robô.

Sigularidade ocorre quando, num determinado ponto do espaço operacional, as configurações possíveis são infinitas, ou seja, quando se pode escrever alguma variável de junta como combinação linear das outras. Pode ser entendida como uma degenerescência pontual.

Observe a relatividade destes conceitos: um robô de seis graus de liberdade será redundante em relação a uma tarefa cuja trajetória só contenha pontos de singularidade, pois para esta tarefa poderia ser fixada uma das juntas, movimentando-se as demais.

Resumindo, o problema da coordenação de movimentos pode visto como: dada uma tarefa no espaço operacional, determinar se existe solução, determinar as configurações possíveis para cada ponto, escolher uma delas, tratar os pontos de singularidade e, no caso de robôs redundantes, escolher uma solução segundo algum critério (exemplo: minimização da energia cinética).

3.3. MÉTODOS DE INVERSÃO

Neste ítem iremos apresentar uma breve comparação entre métodos numéricos e métodos analíticos para obtenção do modelo inverso, e apresentar o método descrito em [PAUL 81], método adotado neste trabalho por ser analítico, trazendo vantagens para o cálculo do modelo inverso em tempo real, e por utilizar a representação por matrizes de transformação homogêneas, também adotada neste trabalho.

3.3.1. MÉTODOS ANALÍTICOS

Vimos, no item anterior que, por ser o modelo geométrico um conjunto de equações transcedentais com somas de produtos de senos e cossenos de ângulos, a inversão analítica não é trivial, e não há garantia de que seja possível fazê-lo para um robô qualquer, a menos que se enquadre nos casos mencionados em [GORLA 84] ou pertença a alguma família cuja solução conste na literatura, por exemplo em [DUFFY 80], [RAMOS 88], [PAUL 81] etc.

Adotamos a inversão analítica por apresentar evidentes vantagens de utilização em relação à inversão numérica, e por sua única desvantagem - a falta de generalidade - ser de menor importância, já que a maioria dos robôs industriais é simples, resultando em esparsidade nas matrizes de transformação elementares, e portanto em modelos geométricos de fácil inversão.

Características do modelo inverso analítico:

a) Explicitação de todas as configurações relativas a uma mesma posição e orientação do órgão terminal: esta explicitação aparece ainda durante a dedução do modelo, gerando "famílias" de soluções para o mesmo modelo (cada "família" é obtida de uma equação diferente).

b) Cálculo do modelo em tempo real: o tipo de expressão que aparece no modelo inverso analítico (veja exemplo no quadro (3.1.)) é relativamente simples, sendo fácil o seu cálculo em tempo real. Obviamente devemos verificar se temos um processador que calcule o modelo segundo as necessidades de execução em tempo real para o problema em questão.

c) Tratamento de regiões próximas a pontos singulares: os pontos singulares são também obtidos durante a dedução do modelo, fornecendo uma identificação "a priori" destes, possibilitando o chaveamento para modelos alternativos não singulares nas regiões próximas às singularidades.

$d_2 = -o_z d_6 + p_z$;

$r_1 = ((p_x - o_x d_6)^2 + (p_y - o_y d_6)^2)^{1/2}$;

$\phi_1 = atar(p_x - o_x d_6,\ p_y - o_y d_6)$;

$d_3 = -r_1 \cos(\phi_1 + \theta_1) = -(r_1^2 - a_2^2)^{1/2}$;

$\theta_1 = atar(a_2,\ -d_3) - \phi_1$;

para & = ±1

$\theta_5 = atar(\& (1 - (-S_1 o_x + C_1 o_y)^2)^{1/2},\ -S_1 o_x + C_1 o_y)$;

se $S_5 > 0$, $\theta_4 = atar(o_z,\ C_1 o_x + S_1 o_y)$ e

$\theta_6 = atar(S_1 a_x - C_1 a_y,\ S_1 n_x - C_1 n_y)$;

se $S_5 < 0$, $\theta_4 = atar(-o_z,\ -C_1 o_x - S_1 o_y)$ e

$\theta_6 = atar(-S_1 a_x + C_1 a_y,\ -S_1 n_x + C_1 n_y)$;

se $S_5 = 0$, $\theta_4 = \theta_{4a}$ e $\theta_6 = \theta_{4a} - atar(n_x,\ a_x)$.

Quadro (3.1.) - Modelo Analítico do Robô TH8

3.3.2. MÉTODOS NUMÉRICOS

Os métodos numéricos são mais gerais, porém necessitam de grande tempo de cálculo, dificultando sua utilização para tempo real. Também são delicados os problemas de convergência e tratamento em torno dos pontos de singularidade, além de necessitarem que se estime uma solução inicial. Sobre o problema de deteção de pontos de singularidade, o que se faz geralmente é determinar à priori as situações que resultem em configurações de singularidade, e evitá-las. Estes métodos convergem para uma única solução, não mostrando todas as configurações possíveis, impedindo a escolha da melhor solução.

Os dois métodos numéricos mais utilizados são os seguintes:

1) Método da linearização da matriz de transformação do robô, que é uma adaptação do método clássico de Newton-Raphson para a solução de um sistema de equações não lineares.

2) Método recursivo, que utiliza o cálculo do modelo geométrico e da matriz Jacobiana inversa (veja diagrama na figura (3.5.)).

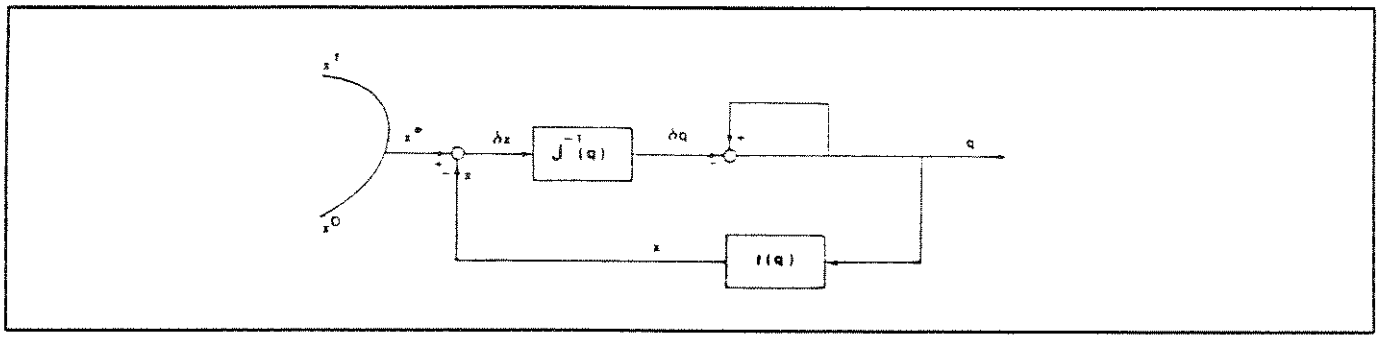


Figura (3.5.) - Método Recursivo de Inversão do Modelo

3.3.3. O MÉTODO ANALÍTICO DE [PAUL 81]

Este método, de natureza geométrica, é baseado na utilização das matrizes de transformação homogêneas elementares do robô. Com estas matrizes, suas inversas e mais a matriz literal montada com a posição e orientação genéricas do órgão terminal (que denominaremos [T_{on}]), são construídas duas matrizes de transformação intermediárias equivalentes entre cada ligação e o órgão terminal determinadas das seguintes maneiras:

Produto das matrizes de transformação elementares a partir da ligação em questão e o órgão terminal.

$$T_{i,i+1} * T_{i+1,i+2} * \ldots * T_{n-1,n} \qquad (3.15.)$$

Pré multiplicação da matriz de transformação final pelas inversas das matrizes de transformação elementares das ligações anteriores à ligação em questão.

$$T^{-1}_{i-1,i} * T^{-1}_{i-2,i-1} * \ldots * T^{-1}_{base,1} * T_{base,n} \qquad (3.16.)$$

A idéia do método é: explicitar sucessivamente as coordenadas generalizadas a partir das expressões geradas quando igualamos os termos de (3.15.) aos termos de (3.16.).

Um exemplo para um robô de seis ligações é mostrado no quadro

(3.2.):

base:	$T_{0,1} * T_{1,2} * T_{2,3} * T_{3,4} * T_{4,5} * T_{5,6} = [T_{0,6}]$
junta 1:	$T_{1,2} * T_{2,3} * T_{3,4} * T_{4,5} * T_{5,6} = T_{0,1}^{-1} * [T_{0,6}]$
junta 2:	$T_{2,3} * T_{3,4} * T_{4,5} * T_{5,6} = T_{1,2}^{-1} * T_{0,1}^{-1} * [T_{0,6}]$
junta 3:	$T_{3,4} * T_{4,5} * T_{5,6} = T_{2,3}^{-1} * T_{1,2}^{-1} * T_{0,1}^{-1} * [T_{0,6}]$
junta 4:	$T_{4,5} * T_{5,6} = T_{3,4}^{-1} * T_{2,3}^{-1} * T_{1,2}^{-1} * T_{0,1}^{-1} * [T_{0,6}]$
junta 5:	$T_{5,6} = T_{4,5}^{-1} * T_{3,4}^{-1} * T_{2,3}^{-1} * T_{1,2}^{-1} * T_{0,1}^{-1} * [T_{0,6}]$

Quadro (3.2.) - Expressões para Auxílio à Inversão com n = 6

Se é possível a obtenção analítica do modelo geométrico inverso do robô, as expressões acima podem fornecer as coordenadas generalizadas diretamente ou mediante alguma manipulação algébrica.

Em geral para uma ligação rotacional, o cálculo da coordenada generalizada passa pela resolução de uma equação do tipo $a*\cos\theta + b*\sin\theta = c$, que tem solução se $(a^2 + b^2) \geq c^2$ (será visto com detalhes no ítem seguinte).

A expressão acima determina duas soluções. Porém, se aparecem expressões isoladas para o seno e o cosseno do ângulo, então este estará unicamente determinado. Em qualquer caso, deve-se utilizar a função que obtém o arco-tangente no quadrante especificado pelos senos e cossenos do ângulo. As funções arco-seno e arco-cosseno não devem ser utilizadas por problemas de precisão e indefinição do quadrante.

Um exemplo deste método será apresentado no ítem seguinte, exemplificando também a utilização do módulo de auxílio à inversão do GMROB.

3.4. FERRAMENTA DE AUXÍLIO À INVERSÃO MODELO GEOMÉTRICO

Como já vimos, uma das principais dificuladades da obtenção do modelo inverso analítico é a dedução das espressões das matrizes intermediárias. Se feito manualmente, demanda um tempo grande e perde em confiabilidade.

Uma ferramenta de auxílio à obtenção do modelo inverso para robôs de cadeia simples foi desenvolvida de modo a gerar automaticamente estas expressões e além disso facilitar a seleção dos elementos das matrizes mais indicados para a manipulação ou que já contenham as expressões individuais das coordenadas generalizadas.

Esta ferramenta (SIS_INV), é um dos módulos de geração de modelos do GMROB, de estrutura interna similar ao módulo de geração do modelo geométrico (SIS_MGD). A diferença básica é que o SIS_INV gera facilidades para obtenção do modelo inverso, ao passo que o SIS_MGD gera o próprio modelo geométrico.

Da forma como foi concebida, toda a dedução do modelo geométrico inverso deveria ser feita interativamente, com uma interface com o usuário que lhe mostrasse os caminhos possíveis, permitisse que ele entrasse com resultados intermediários que o sistema não pudesse determinar, e prosseguisse o processo selecionando elementos para manipulação de acordo com as coordenadas ainda não resolvidas.

Porém verificamos que para a construção de uma ferramenta assim, seria necessário o uso do conhecimento sobre quais termos seriam indicados para manipulação de acordo com a estrutura do robô, conhecimento somente obtenível a partir de amplo e sistematizado estudo de casos.

Para possibilitar este estudo sistemático, além de auxiliar a inversão, implementamos o SIS_INV, que se mostrou extremamente útil na inversão de modelos. Para um estudo sistemático recomendamos, ao invés do SIS_INV, a utilização de uma ferramenta a ser implementada com utilização de método descrito no ítem 3.5., que é ao mesmo tempo uma

extensão do método descrito em [PAUL 81] e uma extensão do método de obtenção das matrizes intermediárias descrito no ítem 2.4..

Sobre o SIS_INV, na sua forma atual, a partir dos parâmetros de Denavit-Hartemberg, são geradas automaticamente as expressões (sem equações auxiliares) no modo pré-multiplicado como descrito no ítem anterior (utilizado largamente na literatura), e também no modo pós-multiplicado:

$$[T_{0,n}] * T_{n-1,n}^{-1} * \ldots * T_{i,i+1}^{-1} = T_{0,1} * T_{1,2} * \ldots * T_{i-1,i} \qquad (3.17.)$$

Esta expressão se mostrou muito útil, gerando em alguns casos, equações de solução mais imediata que o modo pré-multiplicado. Ainda não está implementado no SIS_INV o recurso de simplificações trigonométricas.

Também está implementado um sistema de varredura das expressões para localizar as coordenadas ainda não resolvidas e montar uma cadeia de caracteres com os números destas ligações.

Sem heurísticas para seleção dos elementos e manipulação das expressões, a ferramenta por enquanto gera listagens com todos os elementos das duas matrizes equivalentes colocados lado a lado, mais a cadeia de caracteres com as coordenadas não resolvidas. Isto facilita enormemente a visualização dos elementos para dedução das heurísticas de seleção e manipulação dos mesmos.

Para deteção dos pontos de singularidade e seu tratamento (dedução de equações não singulares), a ferramenta se mostrou muito prática, bastando fixar convenientemente as coordenadas generalizadas rotacionais no arquivo de parâmetros de Denavit-Hartemberg e observar as expressões resultantes.

Para validação das expressões das matrizes intermediárias obtidas das matrizes elementares diretas, foi utilizado o SIS_MGD nos modos 1 e 3, sem equações auxiliares. Para validação das matrizes equivalentes foram utilizados os modelos resolvidos manualmente em [RAMOS 86],

[MEGAHED 84] e [PAUL 81].

Toda a geração de modelos implementada no GMROB só se aplica a robôs manipuladores de cadeias simples. Está em fase de concepção a geração de modelos para robôs de cadeias complexas com paralelogramos, onde ainda se aplicam os parâmetros de Denavit-Hartemberg.

A seguir veremos a resolução do modelo inverso do robô TH8 com auxílio das expressões geradas pelo módulo de auxílio do GMROB.

Os referenciais estabelecidos para o robô, bem como parâmetros de Denavit-Hartemberg, estão mostrados no ítem 3.2..

Os parâmetros de Denavit-Hartemberg foram introduzidos no módulo de inicialização do GMROB e depois foi acionado o módulo de auxílio à obtenção do modelo inverso.

Neste módulo foram gerados os arquivos Th-8.gi1 e Th-8.gi2, que contêm as expressões a serem utilizadas para determinação do modelo inverso. O arquivo Th-8.gi1 contem as expressões geradas com o modo de obtenção por pré-multiplicação das matrizes; o arquivo Th-8.gi2 contém as expressões geradas por pós-multiplicação.

Encerrada a gravação dos arquivos pelo GMROB, pôde-se abandonar o programa e comandar sua impressão. Estes arquivos estão mostrados nos quadros (3.3.) a (3.16.).

SISTEMA DE AUXILÍO À OBTENCÃO DO
MODELO GEOMÉTRICO INVERSO

MATRIZES DE TRANSFORMAÇÃO HOMOGÊNEA ENTRE
SISTEMAS DE COORDENADAS DAS JUNTAS

ROBÔ TH8

MODO PRÉ-MULTIPLICADO

Ex: para robô de 6 juntas (T' = inversa)

$$T_{06} = T_{01} * T_{12} * T_{23} * T_{34} * T_{45} * T_{56}$$

$$T'_{01} * T_{06} = T_{12} * T_{23} * T_{34} * T_{45} * T_{56}$$

$$T'_{12} * T'_{01} * T_{06} = T_{23} * T_{34} * T_{45} * T_{56}$$

$$T'_{23} * T'_{12} * T'_{01} * T_{06} = T_{34} * T_{45} * T_{56}$$

$$T'_{34} * T'_{23} * T'_{12} * T'_{01} * T_{06} = T_{45} * T_{56}$$

$$T'_{45} * T'_{34} * T'_{23} * T'_{12} * T'_{01} * T_{06} = T_{56}$$

Quadro (3.3.) - Modo Pré-multiplicado do Auxílio à Inversão

MATRIZ-T06

```
elemento(0,0)                                   variáveis  1456
+nx
= +C1(+C4C5C6+S4S6)+S1S5C6

elemento(0,1)                                   variáveis  145
+ox
= +C1C4S5-S1C5

elemento(0,2)                                   variáveis  1456
+ax
= +C1(+C4C5S6-S4C6)+S1S5S6

elemento(0,3)                                   variáveis  1453
+px
= +C1(+C4S5d6+a2)+S1(-C5d6+d3)

elemento(1,0)                                   variáveis  1456
+ny
= +S1(+C4C5C6+S4S6)-C1S5C6

elemento(1,1)                                   variáveis  145
+oy
= +S1C4S5+C1C5

elemento(1,2)                                   variáveis  1456
+ay
= +S1(+C4C5S6-S4C6)-C1S5S6

elemento(1,3)                                   variáveis  1453
+py
= +S1(+C4S5d6+a2)-C1(-C5d6+d3)

elemento(2,0)                                   variáveis  456
+nz
= +(+S4C5C6-C4S6)

elemento(2,1)                                   variáveis  45
+oz
= +S4S5

elemento(2,2)                                   variáveis  456
+az
= +(+S4C5S6+C4C6)

elemento(2,3)                                   variáveis  452
+pz
= +(+S4S5d6+d2)
```

Quadro (3.4.) - Matriz T06

MATRIZ T_{16}

elemento(0,0) variáveis 1456

+C1nx+S1ny
= +(+C4C5C6+S4S6)

elemento(0,1) variáveis 145

+C1ox+S1oy
= +C4S5

elemento(0,2) variáveis: 1456

+C1ax+S1ay
= +(+C4C5S6-S4C6)

elemento(0,3) variáveis: 145

+C1px+S1py
= +C4S5d6+a2

elemento(1,0) variáveis: 156

-S1nx+C1ny
= -S5C6

elemento(1,1) variáveis: 15

-S1ox+C1oy
= +C5

elemento(1,2) variáveis: 156

-S1ax+C1ay
= -S5S6

elemento(1,3) variáveis: 153

-S1px+C1py
= -(-C5d6+d3)

elemento(2,0) variáveis: 456

+nz
= +(+S4C5C6-C4S6)

elemento(2,1) variáveis: 45

+oz
= +S4S5

elemento(2,2) variáveis: 456

+az
= +(+S4C5S6+C4C6)

elemento(2,3) variáveis: 452

+pz
= +S4S5d6+d2

Quadro (3.5.) - Matriz T_{16}

MATRIZ—Tz6

```
elemento(0,0)                                   variáveis: 1456

+(+C1nx+S1ny)
= +(+C4C5C6+S4S6)

elemento(0,1)                                   variáveis: 145

+(+C1ox+S1oy)
= +C4S5

elemento(0,2)                                   variáveis: 1456

+(+C1ax+S1ay)
= +(+C4C5S6-S4C6)

elemento(0,3)                                   variáveis: 145

+(+C1px+S1py)-a2
= +C4S5d6

elemento(1,0)                                   variáveis: 456

+nz
= +(+S4C5C6-C4S6)

elemento(1,1)                                   variáveis: 45

+oz
= +S4S5

elemento(1,2)                                   variáveis: 456

+az
= +(+S4C5S6+C4C6

elemento(1,3)                                   variáveis: 245

+pz-d2
= +S4S5d6

elemento(2,0)                                   variáveis: 156

-(-S1nx+C1ny)
= +S5C6

elemento(2,1)                                   variáveis: 15

-(-S1ox+C1oy)
= -C5

elemento(2,2)                                   variáveis: 156

-(-S1ax+C1ay)
= S5S6

elemento(2,3)                                   variáveis: 153

-(-S1px+C1py)
= -C5d6+d3
```

Quadro (3.6.) - Matriz Tz6

MATRIZ-Tα6

```
elemento(0,0)                                   variáveis: 1456
+(+C1nx+S1ny)
= +C4C5C6+S4S6

elemento(0,1)                                   variáveis: 145
+(+C1ox+S1oy)
= +C4S5

elemento(0,2)                                   variáveis: 1456
+(+C1ax+S1ay)
= +C4C5S6-S4C6

elemento(0,3)                                   variáveis: 145
+(+(+C1px+S1py)-a2)
= +C4S5d6

elemento(1,0)                                   variáveis: 456
+nz
= +S4C5C6-C4S6

elemento(1,1)                                   variáveis: 45
+oz
= +S4S5

elemento(1,2)                                   variáveis: 456
+az
= +S4C5S6+C4C6

elemento(1,3)                                   variáveis: 245
+(+pz-d2)
= +S4S5d6

elemento(2,0)                                   variáveis: 156
-(-S1nx+C1ny)
= +S5C6

elemento(2,1)                                   variáveis: 15
-(-S1ox+C1oy)
= -C5

elemento(2,2)                                   variáveis: 156
-(-S1ax+C1ay)
= +S5S6

elemento(2,3)                                   variáveis: 135
-(-S1px+C1py)-d3
= -C5d6
```

Quadro (3.7.) - Matriz Tα6

MATRIZ $T_{4\sigma}$

```
elemento(0,0)                                   variáveis: 4156
+C4(+C1nx+S1ny)+S4nz
= +C5C6

elemento(0,1)                                   variáveis: 415
+C4(+C1ox+S1oy)+S4oz
= +S5

elemento(0,2)                                   variáveis: 4156
+C4(+C1ax+S1ay)+S4az
= +C5S6

elemento(0,3)                                   variáveis: 4125
+C4(+(+C1px+S1py)-a2)+S4(+pz-d2)
= +S5d6

elemento(1,0)                                   variáveis: 156
-(-S1nx+C1ny)
= +S5C6

elemento(1,1)                                   variáveis: 15
-(-S1ox+C1oy)
= -C5

elemento(1,2)                                   variáveis: 156
-(-S1ax+C1ay)
= +S5S6

elemento(1,3)                                   variáveis: 135
+(-(-S1px+C1py)-d3)
= -C5d6

elemento(2,0)                                   variáveis: 416
+S4(+C1nx+S1ny)-C4nz
= +S6

elemento(2,1)                                   variáveis: 41
+S4(+C1ox+S1oy)-C4oz
= 0

elemento(2,2)                                   variáveis: 416
+S4(+C1ax+S1ay)-C4az
= -C6

elemento(2,3)                                   variáveis: 412
+S4(+(+C1px+S1py)-a2)-C4(+pz-d2)
= 0
```

Quadro (3.8.) - Matriz $T_{4\sigma}$

MATRIZ-T56

```
elemento(0,0)                                          variáveis: 5416
+C5(+C4(+C1nx+S1ny)+S4nz)-S5(-S1nx+C1ny)
= +C6

elemento(0,1)                                          variáveis: 541
+C5(+C4(+C1ox+S1oy)+S4oz)-S5(-S1ox+C1oy)
= 0

elemento(0,2)                                          variáveis: 5416
+C5(+C4(+C1ax+S1ay)+S4az)-S5(-S1ax+C1ay)
= +S6

elemento(0,3)                                          variáveis: 54123
+C5(+C4(+(+C1px+S1py)-a2)+S4(+pz-d2))+S5(-(-S1px+C1py)-d3)
= 0

elemento(1,0)                                          variáveis: 416
+(+S4(+C1nx+S1ny)-C4nz)
= +S6

elemento(1,1)                                          variáveis: 41
+(+S4(+C1ox+S1oy)-C4oz)
= 0

elemento(1,2)                                          variáveis: 416
+(+S4(+C1ax+S1ay)-C4az)
= -C6

elemento(1,3)                                          variáveis: 412
+(+S4(+(+C1px+S1py)-a2)-C4(+pz-d2))
= 0

elemento(2,0)                                          variáveis: 541
+S5(+C4(+C1nx+S1ny)+S4nz)+C5(-S1nx+C1ny)
= 0

elemento(2,1)                                          variáveis: 541
+S5(+C4(+C1ox+S1oy)+S4oz)+C5(-S1ox+C1oy)
= +1

elemento(2,2)                                          variáveis: 541
+S5(+C4(+C1ax+S1ay)+S4az)+C5(-S1ax+C1ay)
= 0

elemento(2,3)                                          variáveis: 54123
+S5(+C4(+(+C1px+S1py)-a2)+S4(+pz-d2))-C5(-(-S1px+C1py)-d3)
= +d6
```

Quadro (3.9.) - Matriz T56

SISTEMA DE AUXILÍO À OBTENCÃO DO
MODELO GEOMÉTRICO INVERSO

MATRIZES DE TRANSFORMAÇÃO HOMOGÊNEA ENTRE
SISTEMAS DE COORDENADAS DAS JUNTAS

ROBÔ TH8

MODO PÓS-MULTIPLICADO

Ex: para robô de 6 juntas (T' = inversa)

$$T_{06} = T_{01} * T_{12} * T_{23} * T_{34} * T_{45} * T_{56}$$

$$T_{06} * T'_{56} = T_{01} * T_{12} * T_{23} * T_{34} * T_{45}$$

$$T_{06} * T'_{56} * T'_{45} = T_{01} * T_{12} * T_{23} * T_{34}$$

$$T_{06} * T'_{56} * T'_{45} * T'_{34} = T_{01} * T_{12} * T_{23}$$

$$T_{06} * T'_{56} * T'_{45} * T'_{34} * T'_{23} = T_{01} * T_{12}$$

$$T_{06} * T'_{56} * T'_{45} * T'_{34} * T'_{23} * T'_{12} = T_{01}$$

Quadro (3.10.) - Modo Pós-multiplicado do Auxílio à Inversão

MATRIZ-T06

```
elemento(0,0)                                   variáveis: 1456

+nx
= +(+C1C4C5+S1S5)C6+C1S4S6

elemento(0,1)                                   variáveis: 145

+ox
= +(+C1C4S5-S1C5)

elemento(0,2)                                   variáveis: 1456

+ax
= +(+C1C4C5+S1S5)S6-C1S4C6

elemento(0,3)                                   variáveis: 1453

+px
= +(+C1C4S5-S1C5)d6+(+S1d3+C1a2)

elemento(1,0)                                   variáveis: 1456

+ny
= +(+S1C4C5-C1S5)C6+S1S4S6

elemento(1,1)                                   variáveis: 145

+oy
= +(+S1C4S5+C1C5)

elemento(1,2)                                   variáveis: 1456

+ay
= +(+S1C4C5-C1S5)S6-S1S4C6

elemento(1,3)                                   variáveis: 1453

+py
= +(+S1C4S5+C1C5)d6+(-C1d3+S1a2)

elemento(2,0)                                   variáveis: 456

+nz
= +S4C5C6-C4S6

elemento(2,1)                                   variáveis: 45

+oz
= +S4S5

elemento(2,2)                                   variáveis: 456

+az
= +S4C5S6+C4C6

elemento(2,3)                                   variáveis: 452

+pz
= +S4S5d6+d2
```

Quadro (3.11.) - Matriz T06

MATRIZ-T06

elemento(0,0) variáveis: 6145

+nxC6+axS6
= +C1C4C5+S1S5

elemento(0,1) variáveis: 614

+nxS6-axC6
= +C1S4

elemento(0,2) variáveis: 145

+ox
= +C1C4S5-S1C5

elemento(0,3) variáveis: 13

-oxd6+px
= +(+S1d3+C1a2)

elemento(1,0) variáveis: 6145

+nyC6+ayS6
= +S1C4C5-C1S5

elemento(1,1) variáveis: 614

+nyS6-ayC6
= +S1S4

elemento(1,2) variáveis: 145

+oy
= +S1C4S5+C1C5

elemento(1,3) variáveis: 13

-oyd6+py
= +(-C1d3+S1a2)

elemento(2,0) variáveis: 645

+nzC6+azS6
= +S4C5

elemento(2,1) variáveis: 64

+nzS6-azC6
= -C4

elemento(2,2) variáveis: 45

+oz
= +S4S5

elemento(2,3) variáveis: 2

-ozd6+pz
= +d2

Quadro (3.12.) - Matriz T06

MATRIZ-To4

elemento(0,0) variáveis: 6514

+(+nxC6+axS6)C5+oxS5
= +C1C4

elemento(0,1) variáveis: 651

+(+nxC6+axS6)S5-oxC5
= +S1

elemento(0,2) variáveis: 614

+(+nxS6-axC6)
= +C1S4

elemento(0,3) variáveis: 13

+(-oxd6+px)
= +(+S1d3+C1a2)

elemento(1,0) variáveis: 6514

+(+nyC6+ayS6)C5+oyS5
= +S1C4

elemento(1,1) variáveis: 651

+(+nyC6+ayS6)S5-oyC5
= -C1

elemento(1,2) variáveis: 614

+(+nyS6-ayC6)
= +S1S4

elemento(1,3) variáveis: 13

+(-oyd6+py)
= +(-C1d3+S1a2)

elemento(2,0) variáveis: 654

+(+nzC6+azS6)C5+ozS5
= +S4

elemento(2,1) variáveis: 65

+(+nzC6+azS6)S5-ozC5
= 0

elemento(2,2) variáveis: 64

+(+nzS6-azC6)
= -C4

elemento(2,3) variáveis: 2

+(-ozd6+pz)
= +d2

Quadro (3.13.) - Matriz To4

MATRIZ-Toa

```
elemento(0,0)                                    variáveis: 6541
+(+(+nxC6+axS6)C5+oxS5)C4+(+nxS6-axC6)S4
= +C1

elemento(0,1)                                    variáveis: 654
+(+(+nxC6+axS6)C5+oxS5)S4-(+nxS6-axC6)C4
= 0

elemento(0,2)                                    variáveis: 651
+(+(+nxC6+axS6)S5-oxC5)
= +S1

elemento(0,3)                                    variáveis: 13
+(-oxd6+px)
= +S1d3+C1a2

elemento(1,0)                                    variáveis: 6541
+(+(+nyC6+ayS6)C5+oyS5)C4+(+nyS6-ayC6)S4
= +S1

elemento(1,1)                                    variáveis: 654
+(+(+nyC6+ayS6)C5+oyS5)S4-(+nyS6-ayC6)C4
= 0

elemento(1,2)                                    variáveis: 651
+(+(+nyC6+ayS6)S5-oyC5)
= -C1

elemento(1,3)                                    variáveis: 13
+(-oyd6+py)
= -C1d3+S1a2

elemento(2,0)                                    variáveis: 654
+(+(+nzC6+azS6)C5+ozS5)C4+(+nzS6-azC6)S4
= 0

elemento(2,1)                                    variáveis: 654
+(+(+nzC6+azS6)C5+ozS5)S4-(+nzS6-azC6)C4
= +1

elemento(2,2)                                    variáveis: 65
+(+(+nzC6+azS6)S5-ozC5)
= 0

elemento(2,3)                                    variáveis: 2
+(-ozd6+pz)
= +d2
```

Quadro (3.14.) - Matriz Toa

MATRIZ-Toz

elemento(0,0) variáveis: 6541

+(+(+(+nxC6+axS6)C5+oxS5)C4+(+nxS6-axC6)S4)
= +C1

elemento(0,1) variáveis: 654

+(+(+(+nxC6+axS6)C5+oxS5)S4-(+nxS6-axC6)C4)
= 0

elemento(0,2) variáveis: 651

+(+(+nxC6+axS6)S5-oxC5)
= +S1

elemento(0,3) variáveis: 6531

-(+(+nxC6+axS6)S5-oxC5)d3+(-oxd6+px)
= +C1a2

elemento(1,0) variáveis: 6541

+(+(+(+nyC6+ayS6)C5+oyS5)C4+(+nyS6-ayC6)S4)
= +S1

elemento(1,1) variáveis: 654

+(+(+(+nyC6+ayS6)C5+oyS5)S4-(+nyS6-ayC6)C4)
= 0

elemento(1,2) variáveis: 651

+(+(+nyC6+ayS6)S5-oyC5)
= -C1

elemento(1,3) variáveis: 6531

-(+(+nyC6+ayS6)S5-oyC5)d3+(-oyd6+py)
= +S1a2

elemento(2,0) variáveis: 654

+(+(+(+nzC6+azS6)C5+ozS5)C4+(+nzS6-azC6)S4)
= 0

elemento(2,1) variáveis: 654

+(+(+(+nzC6+azS6)C5+ozS5)S4-(+nzS6-azC6)C4)
= +1

elemento(2,2) variáveis: 65

+(+(+nzC6+azS6)S5-ozC5)
= 0

elemento(2,3) variáveis: 6532

-(+(+nzC6+azS6)S5-ozC5)d3+(-ozd6+pz)
= +d2

Quadro (3.15.) - Matriz Toz

MATRIZ—To1

```
elemento(0,0)                                        variáveis:  6541
+(+(+(+nxC6+axS6)C5+oxS5)C4+(+nxS6-axC6)S4)
= +C1

elemento(0,1)                                        variáveis:  651
-(+(+nxC6+axS6)S5-oxC5)
= -S1

elemento(0,2)                                        variáveis:  654
+(+(+(+nxC6+axS6)C5+oxS5)S4-(+nxS6-axC6)C4)
= 0

elemento(0,3)                                        variáveis:  65423
-(+(+(+nxC6+axS6)C5+oxS5)C4+(+nxS6-axC6)S4)a2+
-(+(+(+nxC6+axS6)C5+oxS5)S4-(+nxS6-axC6)C4)d2+
+(-(+(+nxC6+axS6)S5-oxC5)d3+(-oxd6+px))
= 0

elemento(1,0)                                        variáveis:  6541
+(+(+(+nyC6+ayS6)C5+oyS5)C4+(+nyS6-ayC6)S4)
= +S1

elemento(1,1)                                        variáveis:  651
-(+(+nyC6+ayS6)S5-oyC5)
= +C1

elemento(1,2)                                        variáveis:  654
+(+(+(+nyC6+ayS6)C5+oyS5)S4-(+nyS6-ayC6)C4)
= 0

elemento(1,3)                                        variáveis:  65423
-(+(+(+nyC6+ayS6)C5+oyS5)C4+(+nyS6-ayC6)S4)a2+
-(+(+(+nyC6+ayS6)C5+oyS5)S4-(+nyS6-ayC6)C4)d2+
+(-(+(+nyC6+ayS6)S5-oyC5)d3+(-oyd6+py))
= 0

elemento(2,0)                                        variáveis:  654
+(+(+(+nzC6+azS6)C5+ozS5)C4+(+nzS6-azC6)S4)
= 0

elemento(2,1)                                        variáveis:  65
-(+(+nzC6+azS6)S5-ozC5)
= 0

elemento(2,2)                                        variáveis:  654
+(+(+(+nzC6+azS6)C5+ozS5)S4-(+nzS6-azC6)C4)
= +1

elemento(2,3)                                        variáveis:  65423
-(+(+(+nzC6+azS6)C5+ozS5)C4+(+nzS6-azC6)S4)a2+
-(+(+(+nzC6+azS6)C5+ozS5)S4-(+nzS6-azC6)C4)d2+
+(-(+(+nzC6+azS6)S5-ozC5)d3+(-ozd6+pz))
= 0
```

Quadro (3.16.) - Matriz To1

3.4.1. OBTENÇÃO DO MODELO INVERSO DO TH8

Notação: a = atar(b,c) significa "a" é igual ao arco cuja tangente é "b/c", com quadrante determinado pelos sinais de "b" e "c", que correspondem respectivamente aos sinais do seno e do cosseno de "a".

Analisando detidamente as expressões dos arquivos apresentados e as variáveis que aparecem em cada uma delas, localizamos a variável "d2" (=q2) explícita:

MATRIZ To5	
elemento(2,3)	variáveis: 2
-ozd6+pz = +d2	(3.18.)

Quadro (3.17.) - Variável da Junta 2 Explícita

Para resolução das variáveis das juntas 1 e 3 não encontramos nenhuma expressão que forneça qualquer das duas isoladamente ou junto com a variável da junta 2 (variável já resolvida). Conseguimos a solução das duas juntas com o auxílio de duas expressões, apresentadas no quadro (3.18.):

MATRIZ To5	
elemento(0,3)	variáveis: 13
-oxd6+px = +(+S1d3+C1a2)	(3.19.)
elemento(1,3)	variáveis: 13
-oyd6+py = +(-C1d3+S1a2)	(3.20.)

Quadro (3.18.) - Expressões com Variáveis das Juntas 1 e 3

Manipulando as expressões:

S1*(3.19.) - C1*(3.20.):

d3 = S1(px - oxd6) - C1(py - oyd6) (3.21.)

C1*(3.19.) + S1*(3.20.):

a2 = C1(px - oxd6) + S1(py - oyd6) (3.22.)

Fazendo a transformação de variáveis:

$$px - oxd6 = r1 \sin(\phi 1) \quad (3.23.)$$

$$py - oyd6 = r1 \cos(\phi 1) \quad (3.24.)$$

$$\text{restrito a } r1 > 0 \quad (3.25.)$$

Escrito de outra maneira:

$$r1 = ((px - oxd6)^2 + (py - oyd6)^2)^{1/2} \quad (3.26.)$$

$$\phi 1 = atar(px - oxd6,\ py - oyd6) \quad (3.27.)$$

(3.27.) em (3.21.), resulta:

$$d3 = S1\ r1 \sin(\phi 1) - C1\ r1 \cos(\phi 1) = -r1 \cos(\phi 1 + \theta 1) \quad (3.28.)$$

$$\text{Como } d3 < 0 \text{ e } r1 > 0 \text{ então } \cos(\phi 1 + \theta 1) > 0 \quad (3.29.)$$

(3.27.) em (3.22.), resulta:

$$C1\ r1 \sin(\phi 1) + S1\ r1 \cos(\phi 1) = a2 \quad (3.30.)$$

então:

$$\sin(\phi 1 + \theta 1) = a2/r1 \quad (3.31.)$$

$$\cos(\phi 1 + \theta 1) = +(r1^2 - a2^2)^{1/2}/\ r1 \quad (3.32.)$$

Portanto, temos resolvidas as variáveis das juntas 1 e 3, com:

$$d3 = -r1 \cos(\phi 1 + \theta 1) = -(r1^2 - a2^2)^{1/2} \quad (3.33.)$$

$$\theta 1 = atar(a2,\ -d3) - \phi 1 \quad (3.34.)$$

Para resolução das outras variáveis, encontramos informações adequadas nas matrizes "$T_{1\sigma}$" e "$T_{0\sigma}$" (quadros (3.19.) e (3.20.), respectivamente).

MATRIZ $T_{1\sigma}$

elemento(1,1) variáveis: 15

$$-S1ox+C1oy = +C5 \quad (3.35.)$$

Quadro (3.19.) - Expressão com Variável da Junta 5

A expressão (3.35.) fornece duas soluções para "$\theta 5$", pois só a informação do cosseno não determina unicamente o ângulo. Como não há outras expressões que dêem mais informações, concluimos que realmente temos aqui uma duplicidade de solução.

Neste caso, as soluções para "$\theta 5$" são:

$$\theta 5 = atar(\& (1 - (-S1ox + C1oy)^2)^{1/2},\ -S1ox + C1oy), \qquad (3.36.)$$
$$\& = \pm 1$$

Para resolução da variável da junta 4, lançamos mão da matriz "$To6$":

MATRIZ $To6$

elemento(0,1) variáveis 145

$$+ox = +C1C4S5 - S1C5 \qquad (3.37.)$$

elemento(1,1) variáveis 145

$$+oy = +S1C4S5 + C1C5 \qquad (3.38.)$$

elemento(2,1) variáveis 45

$$+oz = +S4S5 \qquad (3.39.)$$

Quadro (3.20.) - Expressões com Variável da Junta 4

(3.37.) * C1 + (3.38.) * S1:

$$C1ox + S1oy = C4\ S5 \qquad (3.40.)$$

De (3.39.) e (3.40.) tiramos "$\theta 4$" caso "$\theta 5$" seja diferente de 0:

$$\text{se } S5 > 0, \quad \theta 4 = atar(oz,\ C1ox + S1oy) \qquad (3.41.)$$
$$\text{se } S5 < 0, \quad \theta 4 = atar(-oz,\ -C1ox - S1oy) \qquad (3.42.)$$

Da mesma forma, lançamos mão da matriz "T_{16}" para resolver "$\theta 6$":

MATRIZ T_{16}

elemento(1,0) variáveis: 156

$$-S1nx + C1ny = -S5C6 \qquad (3.43.)$$

elemento(1,2) variáveis: 156

$$-S1ax + C1ay = -S5S6 \qquad (3.44.)$$

Quadro (3.21.) - Expressões com Variável da Junta 6

Caso "$\theta 5$" seja diferente de 0:

$$\text{se } S5 > 0, \quad \theta 6 = \text{atar}(S1ax - C1ay,\ S1nx - C1ny) \qquad (3.45.)$$

$$\text{se } S5 < 0, \quad \theta 6 = \text{atar}(-S1ax + C1ay,\ -S1nx + C1ny) \qquad (3.46.)$$

Em $\theta 5 = 0$ temos um ponto de singularidade. Podemos estudá-lo utilizando as expressões do quadro (3.22.):

MATRIZ T_{16}

elemento(2,0) variáveis 456

$$+nz = +(+S4C5C6 - C4S6) \qquad (3.47.)$$

elemento(2,2) variáveis 456

$$+az = +(+S4C5S6 + C4C6) \qquad (3.48.)$$

Quadro (3.22.) - Expressões para Análise de Singularidade

Com uma análise nos limites de juntas, se S5=0 temos C5=1. As expressões (3.47.) e (3.48.) se tornam, respectivamente:

$$nz = \text{sin}(\theta 4 - \theta 6) \qquad (3.49.)$$

$$az = \cos(\theta 4 - \theta 6) \qquad (3.50.)$$

Daí, temos: $\theta 4 - \theta 6 = \text{atar}(nz, az)$ (3.51.)

Um critério para resolver esta singularidade pode ser a fixação de "$\theta 4$" no valor anterior ($\theta 4 = \theta 4a$). Neste caso:

$$\theta 4 = \theta 4a \qquad (3.52.)$$

$$\theta 6 = \theta 4a - \text{atar}(nz, az) \qquad (3.53.)$$

O resumo do modelo está apresentado no quadro (2.23.):

$d_2 = -o_z d_6 + p_z;$

$r_1 = ((p_x - o_x d_6)^2 + (p_y - o_y d_6)^2)^{1/2};$

$\phi_1 = atar(p_x - o_x d_6,\ p_y - o_y d_6);$

$d_3 = -r_1 \cos(\phi_1 + \theta_1) = -(r_1^2 - a_2^2)^{1/2};$

$\theta_1 = atar(a_2, -d_3) - \phi_1;$

para & = ±1

$\theta_5 = atar(\& (1 - (-S_{10}x + C_{10}y)^2)^{1/2},\ -S_{10}x + C_{10}y);$

se $S_5 > 0$, $\theta_4 = atar(o_z, C_{10}x + S_{10}y)$ e

$\theta_6 = atar(S_{1a}x - C_{1a}y,\ S_{1n}x - C_{1n}y);$

se $S_5 < 0$, $\theta_4 = atar(-o_z, -C_{10}x - S_{10}y)$ e

$\theta_6 = atar(-S_{1a}x + C_{1a}y,\ -S_{1n}x + C_{1n}y);$

se $S_5 = 0$, $\theta_4 = \theta_{4a}$ e $\theta_6 = \theta_{4a} - atar(n_z, a_z)$.

Quadro (2.23.) - Modelo Inverso do Robô TH8

3.5. EXTENSÃO DO MÉTODO DE OBTENÇÃO DAS MATRIZES INTERMEDIÁRIAS PARA AUXÍLIO À INVERSÃO DO MODELO

Neste item extenderemos o método visto no item 2.4., para obtenção das matrizes intermediárias inversas, de tal forma que seja possível a geração das expressões de auxílio à inversão do modelo pelo método de [PAUL 81], não só pelos modos pré e pós multiplicados, mas também por identidades correspondentes a cada uma n(n+1)/2 matrizes intermediárias e suas n(n+1)/2 inversas.

No método de [PAUL 81], são utilizadas como expressões de auxílio à inversão, identidades do tipo:

$$T_{0i}^{-1} * [T_{0n}] = T_{in} \qquad \begin{cases} 0 \leq i \leq (n-1) \\ T_{oo} = I \text{ (matriz identidade)} \\ [T_{0n}] \text{ como em (3.55.)} \end{cases} \qquad (3.54.)$$

$$[T_{0n}] = \begin{bmatrix} R(3x3) & P(3x1) \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} nx & ox & ax & px \\ ny & oy & ay & py \\ nz & oz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.55.)$$

Em GMROB, para auxílio à inversão do modelo, obtemos expressões de dois tipos:

1) Modo pré-multiplicado - obtido segundo a mesma expressão acima

2) Modo pós-multiplicado - obtido segundo a expressão:

$$[T_{0n}] * T_{jn}^{-1} = T_{0j} \qquad 0 \leq j \leq (n-1) \qquad (3.56.)$$

Qualquer dos dois conjuntos de identidades acima, correspondentes a um valor de "j", possui a mesma informação que o modelo geométrico final, apenas apresentado de maneira diversa.

A filosofia de nossa abordagem é a de apresentar as identidades sob a forma de mais fácil visualização. Propomos assim, uma nova forma de definição destas, onde são construídas todas as identidades, da seguinte maneira: de um lado colocamos os modelos geométricos

intermediários, e do outro as expressões geradas pela pré-multiplicação e/ou pós-multiplicação da matriz [Ton] pelas inversas das matrizes elementares convenientes.

Exemplo: para um robô com seis graus de liberdade, o modelo geométrico intermediário "T35" fornece a seguinte identidade:

$$\begin{aligned} T_{35} &= T_{23}^{-1}\, T_{12}^{-1}\, T_{01}^{-1}\, [T_{06}]\, T_{56}^{-1} \\ &= T_{08}^{-1}\, [T_{06}]\, T_{56}^{-1} \end{aligned} \qquad (3.57.)$$

Ou, para um modelo geométrico intermediário qualquer:

$$T_{ij} = T_{0i}^{-1}\, [T_{0n}]\, T_{jn}^{-1} \qquad (3.58.)$$

Com a mesma filosofia de melhor visualização, podemos também obter expressões equivalentes com as inversas destas matrizes:

$$T_{ij}^{-1} = T_{jn}\, [T_{0n}^{*}]\, T_{0i} \qquad (3.59.)$$

Onde $[T_{0n}^{*}]$ é a matriz literal correspondente a $[T_{0n}]^{-1}$, podendo ser facilmente obtida desta, pelas expressões a seguir:

$$[T_{0n}^{*}] = \begin{bmatrix} R_{(3x3)}^{*} & P_{(3x1)}^{*} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R^{*} = R^{t} & \begin{matrix} -n.p \\ -o.p \\ -a.p \end{matrix} \\ 0 & 1 \end{bmatrix} \qquad (3.60.)$$

Onde "t" significa transposição e "." significa produto interno dos vetores.

O método descrito em [PAUL 81] se limitou ao uso do modo pré-multiplicado (utilizando parcialmente o modo pós-multiplicado em alguns casos) por dois motivos:

a) A tarefa de obtenção das expressões, se feita manualmente, é extremamente árdua

b) Só havia interesse em chegar num ponto onde se pudesse

visualizar a forma de inversão para um determinado robô.

Como o objetivo do nosso trabalho é propiciar maneiras de se ir além em generalidade e economia, visando o levantamento das heurísticas que permitam a localização automática (quem sabe a inversão automática) dos termos de melhor visualização de acordo com a estrutura do robô, então o módulo de auxílio à inversão do GMROB deve ser extendido para gerar as n(n+1) matrizes, sendo n(n+1)/2 correspondentes aos modelos geométricos intermediários, e o mesmo tanto correspondente às suas inversas.

Veremos a seguir, como o método proposto no item 2.4 pode ser extendido para a obtenção das expressões acima, sempre observando que agora nos interessa somente expressões sem equações auxiliares e com todas as simplificações trigonométricas pertinentes.

Primeiramente vamos apresentar as matrizes intermediárias inversas, procedendo como no item 2.4.: representando as matrizes elementares inversas com uso de partições, e como um produto de duas matrizes mais simples, de forma que os parâmetros de Denavit-Hartemberg fiquem evidentes:

$$T_{i-1,i}^{-1} = \begin{bmatrix} R\alpha_i^t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R\theta_i^t & -P_i \\ 0 & 1 \end{bmatrix} \qquad (3.61.)$$

Onde $R\theta_i$, $R\alpha_i$ e P_i têm a mesma definição que no item 2.4.

Para simplificar a notação, definimos $A_i = T_{i-1,i}$ (3.62.)

Assim, podemos escrever:

$$(A_1A_2)^{-1} = A_2^{-1}A_1^{-1} = \begin{bmatrix} R\alpha_2^t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R\theta_2^t & -P_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R\alpha_1^t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R\theta_1^t & -P_1 \\ 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} R\alpha_2^t R\theta_2^t R\alpha_1^t R\theta_1^t & -R\alpha_2^t P_2 - R\alpha_2^t R\theta_2^t R\alpha_1^t P_1 \\ 0 & 1 \end{bmatrix} \qquad (3.63.)$$

Podemos facilmente deduzir a expressão para uma matriz

intermediária qualquer:

$$(Tmn)^{-1} = \begin{bmatrix} (RmnRan)^t & -\left(\sum_{i=m+1}^{n} (P_i^t R_{i-1,n}^*) \right)^t \\ 0 & 1 \end{bmatrix} \qquad (3.64.)$$

Onde:

$$R_{i-1,n}^* = \begin{cases} \left(\prod_{j=i+1}^{n-1} (Ra_j R\theta_{j+1}) \right) Ran \text{ p/ } n>i \\ Ran \text{ p/ } n=i \end{cases} \qquad (3.65.)$$

Poderíamos obter uma expressão equivalente à apresentada acima utilizando a inversão da matriz intermediária já calculada, porém, isto resultaria numa complexidade muito maior para os termos de translação, prejudicando o nosso maior objetivo, que é a melhor visualização das expressões.

Para os termos de rotação, basta executar a transposição da matriz de rotação intermediária, e teremos a sua inversa.

Para análise e ilustração da forma de obtenção dos termos de translação, usaremos o exemplo da matriz intermediária "T_{15}":

Translação de "T_{15}" ($A_2A_3A_4A_5$):

$$(R\theta_2 (Ra_2R\theta_3 (Ra_3R\theta_4 (Ra_4R\theta_5P_5 + P_4) + P_3) + P_2)) \qquad (3.66.)$$

Translação de "T_{15}^{-1}":

$$-((P_5^t + (P_4^t + (P_3^t + P_2^t Ra_2R\theta_3) Ra_3R\theta_4) Ra_4R\theta_5) Ra_5)^t \qquad (3.67.)$$

Observe que os grupos de matrizes a serem multiplicados pelos vetores "P" são os mesmos nos dois casos, só variando a ordem indicada para cada caso, conforme mostrado no quadro (3.24.).

T_{15}	T_{15}^{-1}
$v_1 = R\alpha_4 R\theta_5 P_5$	$u_1 = P_2^t R\alpha_2 R\theta_3$
$v_2 = R\alpha_3 R\theta_4 (v_1 + P_4)$	$u_2 = (P_3^t + u_1) R\alpha_3 R\theta_4$
$v_3 = R\alpha_2 R\theta_3 (v_2 + P_3)$	$u_3 = (P_4^t + u_2) R\alpha_4 R\theta_5$
$v_4 = R\theta_2 (v_3 + P_2)$	$u_4 = (P_5^t + u_3) R\alpha_5$
$T_{15} = v_4$	$T_{15}^{-1} = -u_4^t$

Quadro (2.24.) - Ordem de Multiplicação para "T_{15}" e "T_{15}^{-1}"

Para análise das simplificações trigonométricas, recaimos na mesma análise do item 2.5., já que as matrizes de rotação são somente transpostas, e entram na formação dos vetores de translação de maneira semelhante.

Resumindo os procedimentos para extensão do método, temos:

a) Cálculo de todas as matrizes intermediárias ótimas (veja critérios no item 2.2.) e suas inversas, considerando a não utilização de equações auxiliares e a utilização das simplificações trigonométricas;

b) Para i variando de 0 a (n-1) e j variando de (i+1) a n: executar os produtos T_{0i}^{-1} [Ton] T_{jn}^{-1} e T_{jn} [T_{on}^{*}] T_{0i} e igualar seus elementos aos das matrizes T_{ij} e T_{ij}^{-1} respectivamente.

Observe que, no passo a) acima utilizamos o critério de melhor visualização para obtenção do ótimo, que obviamente corresponde ao ótimo sob o critério de menor número de operações, no caso da geração ser feita sem equações auxiliares (ao invés disto são utilizadas evidenciações com parênteses) e com simplificações trigonométricas.

No módulo de auxílio à inversão de GMROB, as expressões são obtidas nos modos pré e pós multiplicados, mas no presente método as mesmas expressões deverão ser obtidas pela seqüência ótima.

3.6. VALIDAÇÃO DO MODELO GEOMÉTRICO INVERSO

Durante a obtenção do modelo inverso é muito importante a atenção estrita aos pontos que constituem exceção à equação deduzida, devendo ser achada uma solução específica para estes pontos. Porém isto não é trivial e requer um treinamento grande. Mesmo assim, a complexidade das equações favorecem erros de diversos tipos, que só serão detetados através de um procedimento de validação, análise e simulação gráfica, objetos deste item.

O Modelo Inverso obtido é um conjunto de equações do tipo arco-tangente real (conforme já descrito) e outras equações transcedentais, sendo que o número de soluções das equações teóricas são, não raro, infinito, por causa da multiplicidade de ângulos que atendem às equações-solução, quando não há qualquer tipo de restrição. Existe também o problema de falta de garantia da completeza do conjunto de equações. Isto é, pode ser que do modelo direto (usado na inversão), tenha sido usado um conjunto de informações insuficiente, faltando alguma informação restritiva do modelo.

Além desses, existe também o problema de pontos fora do volume de trabalho, que devem ser alvos de uma proteção a parte, pois não há nada que garanta que as equações geradas para o volume de trabalho não apresentem solução para pontos fora do volume de trabalho.

Os programas de validação das equações do modelo inverso são praticamente idênticos ao da implementação para geração de trajetórias no espaço das coordenadas generalizadas a partir dos pontos em coordenadas operacionais.

No âmbito do Laboratório de Automação e Robótica da III^{a} EBAI (Escola Brasil-Argentina de Informática), foram feitas validações de modelos de robôs, e este procedimento detectou inúmeros erros nos modelos, revelando-se essencial para prevenir condições de erro, propiciando uma programação robusta.

Como ferramenta para validação, foi desenvolvida naquele labora-

tório, sob nossa orientação, um conjunto de programas para validação do robô TH8, escritos de forma bastante modular, em linguagem C, de forma a facilitar o uso para validação de outros robôs, o que foi inclusive feito no ano seguinte, no âmbito do Laboratório de Robótica do IVº EBAI, onde foi validado o robô 4R, construído em Córdoba, Argentina.

Os programas desenvolvidos estão apresentados no Anexo 3.

CAPÍTULO 4

CONCLUSÕES/PERSPECTIVAS

O objetivo deste trabalho foi o de estudar as questões relacionadas com a otimização e auxílio à inversão de modelos geométricos de robôs com estrutura cinemática simples e corpos considerados rígidos. A finalidade da otimização do modelo geométrico é a sua aplicação em tempo real e na obtenção do modelo dinâmico com vistas também a sua otimização. A finalidade de obtenção do modelo inverso analítico é sua utilização na coordenação de movimentos de robôs.

PRINICIPAIS RESULTADOS:

Modelagem Geométrica:

1) Concluimos que, para otimização do modelo geométrico e modelos geométricos intermediários de um robô, é necessária a sua geração pelas maneiras possíveis dentro de alguns critérios restritivos. Listamos e discutimos os critérios relativos a representação, parametrização, sequência de multiplicação das matrizes elementares, uso de simplificações aritméticas, trigonométricas e propriedades de vetores ortonormais. Também desenvolvemos um algoritmo para geração de todas as sequências possíveis de multiplicação das matrizes elementares.

2) Desenvolvemos uma ferramenta eficiente e confiável para geração simbólica automática de modelos geométricos, que faz o cálculo do modelo por até vinte maneiras diferentes, que está longe de esgotar todas as possibilidades com vistas à otimização, mas cumpre muito bem as funções para as quais foi desenvolvida: ferramenta didática para uso na formação de profissionais, ferramenta que produzisse bons modelos para uso de pesquisadores e para uso posterior na obtenção de modelos dinâmicos.

3) Desenvolvemos um método de geração das matrizes intermediárias que resgata as informações parciais de rotação e translação no modelo geométrico final, e possibilita o estudo analítico de casos de simplificações trigonométricas nas expressões do modelo geométrico.

4) Fizemos um estudo analítico exaustivo para localização das possibilidades de ocorrência de simplificações trigonométricas nas expressões do modelo geométrico e, para cada caso, deduzimos os resultados das expressões simplificadas.

Coordenação de Movimentos:

1) Concluímos que a utilização do modelo inverso analítico para coordenação de movimentos apresenta nítidas vantagens sobre os métodos numéricos: menor tempo de cálculo para uso em tempo real, deteção de singularidades, possibilidade de escolha de soluções quando o robô apresenta múltiplas soluções, sendo que a única desvantagem - a falta de generalidade - não é significativa, pois a maioria dos robôs industriais é simples, possibilitando a inversão analítica mediante manipulações das expressões do modelo.

2) Implentamos uma ferramenta de auxílio à inversão que gera expressões com informações redundantes em relação ao modelo geométrico, porém de forma a permitir a visualização das manipulações necessárias à explicitação das coordenadas generalizadas. O método utilizado para geração destas expressões é o descrito em [PAUL 81].

3) Desenvolvemos também uma extensão do método de geração das matrizes intermediárias para geração das expressões de auxílio à inversão, constituindo também uma extensão do método descrito em [PAUL 81].

PERSPECTIVAS

Modelagem Geométrica:

1) Para trabalhos futuros, recomendamos a construção de uma ferramenta mais genérica que o GMROB, que gere o modelo geométrico com todas as possíveis sequencias de multiplicação das matrizes, e utilizando os demais resultados deste trabalho com relação a simplificações aritméticas, trigonométricas, usos de bases ortonormais e equações auxiliares (muitos já contidos no GMROB).

2) Em relação a representação e parametrização, recomendamos um estudo mais aprofundado em relação a: posicionamento de eixos, adoção de outros tipos de representação a quatro parâmetros e outras formas de obtenção do modelo geométrico com vistas à sua otimização.

Modelagem Inversa:

1) Sugerimos a construção de uma ferramenta que gere as expressões de auxílio à inversão correspondente a cada uma das matrizes intermediárias. Esta ferramenta poderia ser obtida facilmente com uma extensão da ferramenta sugerida para o modelo direto. Sua motivação principal é o levantamento de heurísticas que leve à identificação das expressões de mais fácil manipulação para explicitação das coordenadas generalizadas.

BIBLIOGRAFIA

[ARMADA 82] M.A. ARMADA, *Aplicaciones de los Robots Industriales.* Revista de Robotica, Espanha, 1982.

[ARMSTRONG 79] W.W. ARMSTRONG, *Recursive Solution to the Equations of Motion of n-Links Manipulator.* Proc. of 5th World Congress on Theory of Machines and Mechanisms, July 1979.

[CRAIG 85] J.J. CRAIG, *Introduction to Robotics - Mechanics and Control.* Addison-Wesley Publishing Company, USA, 1986.

[DENAVIT 55] J. DENAVIT & R.S. HARTEMBERG, *A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices.* Journal of Applied Mechanics, June 1955.

[DUFFY 80] J. DUFFY, *Analysis of Mechanisms and Robot Manipulators.* John Wiley & Sons, 1980.

[FERREIRA 84] E.P. FERREIRA, *Contribution a L'Identification de Paramètres et a la Commande Dynamique-Adaptative des Robots Manipulateurs.* Thèse de Docteur, Université Paul Sabatier, Toulouse, France, 1984.

[FERREIRA 87] E.P. FERREIRA, *Robótica Industrial: Aspectos Macroscópicos. Robôs Manipuladores: Tecnologias, Modelagem e Controle.* Editorial Kapelusz S.A., Buenos Aires, Argentina, 1987.

[FERREIRA 88] E.P. FERREIRA, J.V.L. SILVA et ali, *GMSIR - A Simulation Tool for Robot Manipulators Architecture and Control Synthesis.* IIIº Congreso Latinoamericano de Automática, Viña del Mar. Chile, 1988.

[FERREIRA 88A] E.P. FERREIRA & alii, *Laboratório de Automação Industrial.* IIIª Escola Brasileiro - Argentina de Informática, Curitiba, janeiro de 1988.

[FERREIRA 89] A.C. FERREIRA & J.P.ANDRADE FILHO, *Otimização do Modelo Geométrico de Robôs Manipuladores.* 1º Encontro Regional de Automção e Instrumentação, Vitória, Espírito Santo, julho de 1989

[GORLA 84] B. GORLA & M. RENAUD, *Modèles des Robots Manipulateurs, Application a leur Commande*. Cepadues Editions, Toulouse, France, 1984.

[HAYATI 84] S. HAYATI & M. MIRMIRANI, *Puma 600 Robot Arm Geometric Calibration*. IEEE Trans. Systems, Man, Cybernetics, 1980.

[KHALIL 86] W. KHALIL & J.F. KLEINFINGER, *A New Geometric Notation for Open and Closed Loop Robots*. Proc. IEEE Conf. on Robotics and Automation, 1986.

[KOREIN 87] J.V. KOREIN & J. ISH-SHALON, *Robotics*. IBM Systems Journal vol. 26, nº 1, 1987.

[LUH 80] J.Y.S. LUH, M.W. WALKER & R.P.C. PAUL, *On Line Computational Scheme for Mechanical Manipulators*. Journal of Dynamic Systems, Measurement and Control, June 1980.

[MEGAHED 82] S. MEGAHED & M. RENAUD, *Minimization of the Computation Time Necessary for Dynamic Control of Robot Manipulators*. 12éme I.S.I.R., Paris, France, Juin 1982.

[MEGAHED 83] S. MEGAHED & M. RENAUD, *Kinematic Modelling of Robot Manipulators Containing Closed Kinematic Chains*. 2nd Conf. of Mechanical Design and Production Engineering, Cairo, 1983.

[MEGAHED 84] S.M. MEGAHED, *Contribution à la Modélisation Géométrique et Dynamique des Robots Manipulateurs à Structure de Chaine Cinématique Simple ou Complexe*. Th. Doctorat D'État, Université Paul Sabatier, Automatique, Toulouse III, France, 1984.

[PAUL 81] R.P. PAUL, *Robots Manipulators - Mathematics, Programming and Control*. MIT Press, Cambridge, Massachusetts and London, England, 1981.

[RAMOS 88] J.J.G. RAMOS, A.C.FERREIRA & alii, *Estação de Trabalho para Robôs Industriais (ETRI)*. 8º Seminário de Comando Numérico no Brasil, São Paulo, 1988.

[RAMOS 86] J.J.G. RAMOS, *Geração de Trajetórias para Robôs Manipuladores: Aspectos Cinemáticos e Computacionais*. Tese de Mestrado, Universidade Estadual de Campinas, 1986.

[RENAUD 80] M. RENAUD, *Contribution à la Modélisation et à la Commande Dynamique des Robots Manipulateurs*. Th. de Doctorat D'État, Université Paul Sabatier, Toulouse, France, 1980.

[RENAUD 84] M. RENAUD, *Calcul du Modèle Géométrique des Robots Manipulateurs en Utilisant un Nombre Quasi Minimal D'Operations Arithmetiques. Publication LAAS du CNRS*, Toulouse, France, 1984.

[SILVA 88] R.S. SILVA, A.C. FERREIRA et ali, *Geração Automática de Modelos para Robôs Manipuladores*, 7º Congresso Brasileiro de Automática, Agosto 1988.

[SILVA 89] J.V.L. DA SILVA, E.P. FERREIRA & J.P. ANDRADE FILHO, *Um Algoritmo Eficiente para a Geração de Modelos Dinâmicos de Robôs Manipuladores*. 1º Encontro Regional de Automação e Instrumentação, Vitória, ES, Agosto 1989.

[SILVA 90] J.V.L. DA SILVA, *Otimização na Geração Automática dos Modelos Dinâmicos para o Controle e a Estimação de Parâmetros de Robôs*. Dissertação de Mestrado, Universidade Estadual de Campinas, a ser apresentada, julho 1990.

[SNYDER 85] W.E. SNYDER, *Industrial Robots: Computer Interfacing and Control*. Prentice-Hall, Inc. New Jersey, EUA, 1985.

[UICKER 65] J.J. UICKER, *On the Dynamic Analysis of Spatial Linkages Using 4 by 4 Matrices*. PhD. Thesis, Department of Mechanical Engineering and Astronautical Sciences, Northwestern University, USA, 1965.

ANEXO 1

MANUAL DO GMROB

MANUAL DO USUÁRIO

SISTEMA PARA GERAÇÃO DE MODELOS DE ROBÔS MANIPULADORES

Versão 1.0

CENTRO TECNOLÓGICO PARA INFORMÁTICA

INSTITUTO DE AUTOMAÇÃO / DIVISÃO DE ROBÓTICA

Rodovia SP 340 - Km 105,4 - Campinas - CEP 13081

CP 6162 - São Paulo - Brasil

ÍNDICE

1 - Introdução

Com a demanda crescente de robôs mais precisos, leves e rápidos, os problemas de síntese relativos aos sistemas constituintes destes robôs tornam-se extremamente complexos, exigindo para a sua solução um ferramental adequado. Apesar do aumento da demanda por atividades de síntese, constata-se uma grande lacuna no que se refere a metodologia e ferramentas adequadas a tal atividade. Este trabalho visa contribuir para o preenchimento desta lacuna.

A necessidade de se implementar um sistema de geração de modelos está associada principalmente a dois fatores: a falta de confiabilidade e o tempo gasto na obtenção manual destes modelos.

Nesta versão o módulo de geração de modelos é constituído funcionalmente dos seguintes sub-módulos:

- Gerador Automático de Modelo Geométrico Direto

- Gerador Automático de Modelo Dinâmico de Referência

- Ferramenta de Auxílio à Obtenção do Modelo Geométrico Inverso

A arquitetura deste módulo foi implementada da seguinte forma:

- Supervisão: coordena o uso da Inicialização e Modelamento.

- Inicialização : permite que sejam introduzidos dados geométricos e dinâmicos de robôs ou utilizados dados que já tenham sido introduzidos anteriormente.

- Modelamento: permite a geração de modelos geométrico direto e dinâmico de referência em sua forma analítica a partir dos parâmetros geométricos e dinâmicos do robô. Auxilia na obtenção do modelo inverso.

O modelamento dispõe de um sistema conversacional simples por menus, o qual possibilita ao usuário inicializar, executar e obter resultados dos vários sub-módulos. Foi desenvolvido em linguagem "C" usando alocação dinâmica de memória e implementado em microcomputadores tipo PC-XT.

2 - Inicialização

A inicialização centraliza todas as manipulações de criação, destruição, apresentação e listagem dos arquivos de dados necessários para a utilização do Modelamento.

Existem dois tipos de inicialização :

- Parâmetros Geométricos

- Parâmetros Dinâmicos

Uma vez inicializados, estes dados serão mantidos em arquivos que poderão ser acessados sempre que necessário. O significado e a forma de obtenção dos parâmetros geométricos são vistos em detalhes no Anexo (5.1). O significado dos parâmetros dinâmicos é dado no item (3.2.1) inicialização.

É feito um controle seguro da criação ou destruição destes arquivos. No caso de se tentar criar um arquivo já existente, o sistema dará ao usuário as opções de :

- Modificar o conteúdo do arquivo já existente;

- Modificar o nome do arquivo que está sendo criado;

- Utilizar o arquivo já existente.

Quando se deseja destruir um arquivo de inicialização, o sistema auxilia o usuário no sentido de destruir todos os arquivos que são resultantes da geração de modelos usando-se este arquivo de inicialização. A destruição dos arquivos é feita um por um, mediante autorização do usuário, de forma a destruir apenas os arquivos desejados.

3 - Geração de Modelos

Para a obtenção de modelos é preciso que os dados de entrada necessários tenham sido introduzidos através da inicialização. A falta de algum dos arquivos de dados necessários provocará o envio de mensagem elucidativa e retorno à inicialização. O Modelamento é constituído dos sub-módulos seguintes:

3.1 - Geração Automática de Modelo Geométrico Direto

Permite ao usuário obter expressões simbólicas de :

- Matrizes de transformações elementares, construídas segundo a convenção tradicional de Denavit-Hartemberg [Denavit 55] ou segundo a modificação utilizada por [Craig 85].

- Matrizes de transformações intermediárias geradas durante o processo de obtenção do modelo geométrico. Estas matrizes relacionam o sistema de coordenadas da base com os sistemas de coordenadas associados a cada elemento da cadeia cinemática.

- Matriz de transformação homogênea que relaciona o sistema de coordenadas da base com o sistema de coordenadas do órgão terminal. Esta matriz corresponde ao Modelo Geométrico e o sistema permite que seja expresso em coordenadas cartesianas (cossenos diretores) ou parâmetros de Euler.

Estes resultados podem ser obtidos para cadeias cinemáticas simples (abertas) e rígidas , com juntas que possuam apenas um grau de liberdade , sendo do tipo rotacional ou prismática.

As expressões simbólicas são geradas em código "C" compilável e nelas são adotadas as seguintes convenções :

a) Si <==> sen(i) e Ci <==> cos(i) ; i = 1,2,...,n

b) Sij <==> sen(i+j) e Cij <==> cos(i+j) ; i,j = 1,2,...,n

c) Sijk <==> sen(i+j+k) e Cijk <==> cos(i+j+k); i,j,k = 1,2,...,n

d) D[i] <==> "i"ésima variável auxiliar

e) T[i][j] <==> termo da linha "i" e coluna "j" da matriz de transformação homogênea.

A Geração Automática do Modelo Geométrico Direto, permite que o usuário teste diversas alternativas de obtenção deste modelo, visando obter aquele que necessite um menor número de operações matemáticas para ser avaliado numéricamente. Estão implementadas cinco alternativas que podem ser testadas de forma automática e sequencial caso solicitado, ou qualquer delas

individualmente. Estas alternativas são as seguintes:

Método 1 - Multiplicação das matrizes elementares completas a partir do sistema de coordenadas da base até o sistema de coordenadas do órgão terminal, isto é:

$$T_{base,1} * T_{1,2} * T_{2,3} * \ldots * T_{n-1,n} \quad \text{--------}>$$

Método 2 - Idem método 1, porém usando 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 outras duas.

Método 3 - Multiplicação das matrizes elementares completas a partir do sistema anterior ao do órgão terminal até o sistema da base, isto é :

$$T_{base,1} * (T_{1,2} * (T_{2,3} * (\ldots * (T_{n-1,n} * T_n) \quad <\text{----------}$$

Método 4 - Idem método 3, porém usando a propriedade de ortonormalidade das três primeiras colunas da matriz de transformação homogênea.

Método 5 - Idem método 1, porém efetuando simplificações trigonométricas.

Análises feitas mostraram que este método híbrido (método 5), mesclando equações auxiliares com simplificações trigonométricas poderia reduzir ainda mais o porte do modelo. Estas simplificações surgirão se houver paralelismo de eixos rotacionais do robô , desde que os eixos sejam consecutivos ou entremeados somente por ligações prismáticas.

Os cinco métodos de obtenção utilizam uma técnica de minimização de cálculos que consiste em eliminar cálculos redundantes através da substituição de toda expressão que indicar uma operação aritmética por uma variável auxiliar. O modelo gerado passa a consistir então de equações auxiliares expressando as substituições realizadas. Assim , por exemplo :

Se durante a geração simbólica do modelo surge a expressão parcial :

```
S2*C2
```

ela é então substituída pela variável auxiliar D[1] e acrescida ao modelo final a equação :

```
D[1] = S2*C2;
```

Embora do ponto de vista de redução do modelo esta técnica deva ser sempre utilizada, para fins de simples visualização das expressões do modelo com objetivos didáticos e/ou comparação com outras referências , é possível obter as expressões simbólicas na sua forma expandida. neste caso os termos são identificados com o uso de parênteses.

No caso de robôs com arquitetura que permita simplificações trigonométricas, o usuário é informado automaticamente pelo sistema. As simplificações trigonométricas que podem ocorrer são do tipo sen(i+j) , cos(i+j) , sen(i+j+k) , cos(i+j+k) e assim por diante.

3.1.1 - Inicialização de Parâmetros Geométricos

Ao entrar no sistema será apresentado o menu principal:

Voce deseja :

[I]nicializar sistema
[G]erar modelos
[A]uxilio para uso
[D]eixar o sistema

Selecione opcao desejada -->

Deve-se optar pela função **[I]nicializar sistema**, que entrará no módulo correspondente, apresentando na tela o seguinte menu :

Voce deseja inicializar:

Parametros [G]eometricos
Parametros [D]inamicos
[A]uxilio ao usuario
[V]oltar ao nivel anterior

Selecione opcao desejada -->

Deve-se selecionar a opção **Parametros [G]eometricos** para que seja possível entrar com este tipo de informação.

Na inicializacao existe à sua disposição um diretório básico contendo os arquivos de dados de entrada que já foram criados , ao qual são acrescentados os novos arquivos criados.

Este diretório permite que você use , referenciando-se apenas ao **nome** do robo , os dados referentes a parâmetros geométricos que já tenham sido introduzidos anteriormente. Pode-se listar, ver, gravar, deletar arquivos existentes e/ou

criar novos arquivos. Quando se especifica um dado robô ,o sistema fornece uma relação dos arquivos similares existentes. No caso de não existir nenhum , pode-se criar um novo arquivo, que terá o mesmo nome que você especificar para o robô, com extensão tipo ".dh".

Exemplo :

Nome = Puma
Arquivo criado = Puma.dh

O menu apresentado ao usuário neste ponto é o seguinte :

Voce deseja :

[L]istar arquivos do diretorio
[A]presentar arquivo do diretorio
[I]mprimir arquivo do diretorio
[C]riar arquivo do diretorio
[D]eletar arquivo do diretorio
[V]oltar ao menu anterior

Digite comando -->

[L]istar - Mostra todos os arquivos de dados do diretório em questão (sisdir.dh).

Exemplo:

****
Atencao

Existe(m) o(s) seguinte(s) arquivos no diretorio sisdir.dh:

- puma.dh
- th8.dh
- stanf.dh
****

[A]presentar - Apresenta o conteúdo de um arquivo de dados específico.

[I]mprime - Imprime o arquivo de dados desejado. A impressora deve estar conectada e ligada, para evitar que o sistema trave.

[C]riar - Para criação de novo arquivo de dados.

Nesta opção o usuário deve ter disponível os parâmetros geométricos obtidos pela convenção tradicional de Denavit-Hartemberg ou pela convenção modificada por Craig.

O sistema pede ao usuário o nome do arquivo que se quer criar:

Digite nome do arquivo (sem extensao) -->

E a forma como os dados deste arquivo foram obtidos:

Estes parametros foram obtidos atraves da :

Convencao [T]radicional de Denavit-Hartenberg

Convencao [C]raig (Denavit-Hartenberg modificada)

Informe convencao adotada -->

Em seguida deve-se informar o número de juntas e os parâmetros para cada junta. É informado então ao usuário quais são os parâmetros que devem ser introduzidos, assumindo automaticamente a variável de junta como rotação em torno do eixo "Z" (teta1) no caso de ligação rotacional, ou como uma translação neste mesmo eixo (D1), no caso de uma ligação translacional (prismática).

Os parâmetros são gravados em uma forma padrão ao entrar-se com caracteres não-numéricos, ou gravados numericamente quando a entrada for um caracter numérico.

Em resumo, para introdução de novos dados deve-se selecionar a função **[C]riar** e o sistema orienta ao usuário quanto às unidades e tipo dos dados, à medida que são solicitados.

[D]eletar - Para eliminar um arquivo de dados específico.

Todos os arquivos resultados de processamento sobre o arquivo que se deseja destruir têm a opção de serem também destruídos, dependendo apenas da confirmação do usuário.

[S]air - Retorna o sistema para o menu anterior.

3.1.2 - Geração de Modelos Geométricos

Para gerar Modelos Geométricos o usuário deve ter as opções do menu principal e selecionar a opção **[G]erar modelos**. Será então apresentado o seguinte menu :

Voce deseja :

gerar modelo [G]eometrico Direto
gerar modelo [I]nverso
gerar modelo [D]inamico de Referencia
[A]uxilio para uso
[V]oltar ao menu anterior

Selecione opcao desejada -->

Selecionada a opção **gerar modelo [G]eométrico Direto**, é solicitado ao usuário o nome do arquivo que contém os parâmetros geométricos do robô cujo modelo se deseja gerar :

Digite apenas o nome do robo manipulador -->

e pede confirmação do nome :

Deseja confirmar o nome do robo? (s/n) -->

É verificado então se o arquivo de dados "nome do robo".dh está disponível. Este arquivo é o que contém os parâmetros geométricos do robô escolhido. Em caso negativo a execução não é possível e o sistema retorna automáticamente para a inicialização, para que se possa providenciar a criação do arquivo necessário. Caso exista mais de um arquivo de dados que contenha o nome do robô como parte do seu nome , o sistema solicita ao usuário uma decisão quanto ao arquivo a ser utilizado. Exemplo para o robô "puma" :

****
ATENCAO

Existe(m) o(s) seguinte(s) arquivo(s) puma no diretorio :

puma.dh
pumacr.dh

****

Selecione o arquivo desejado (sem extensao) -->

A partir daí o sistema entrará na etapa de geração das matrizes de transformações intermediárias. Em seguida, pergunta ao usuário se deseja executar sequencialmente as cinco opções de cálculo.

Autoriza a escolha automatica do melhor modelo ? (s/n) -->

Neste caso o sistema executa todas as opções e elege a que exige menor número de operações matemáticas para ser avaliada.

O usuário é então informado da convenção adotada (Denavit-Hartemberg ou Craig) na inicialização do arquivo. É dada a opção de apresentar as matrizes de transformações elementares na tela.

```
**********************************************
*      Geracao das Matrizes Elementares      *
**********************************************
```

Deseja apresenta-las na tela ? (s/n) -->

Sendo então geradas estas matrizes, o sistema passará à fase de geração do modelo geométrico. Nesta etapa é perguntado ao usuário se deseja um modelo com orientação expressa em coordenadas cartesianas ou parâmetros de Euler:

Deseja expressar o Modelo Geométrico Direto por :

0 --> coordenadas cartesianas

1 --> parametros de Euler

Digite opcao desejada -->

Se a opção escolhida anteriormente foi de gerar o melhor modelo geométrico direto, aparece uma mensagem de execução, bem como o número do método que está sendo executado e a contabilidade de operações necessárias para avaliar o modelo por ele gerado. Em seguida o modelo e suas equações auxiliares (optado por gerar equações auxiliares) são apresentados na tela.

É importante lembrar que na opção de gerar o melhor modelo, o método (5) é executado somente se houver simplificações trigonométricas.

No caso de não ter sido escolhida a geração automática do melhor modelo, é dado ao usuário a opção de gerá-lo por um dos cinco métodos apresentados anteriormente:

Escolha metodo para geracao da Matriz de Transicao Homogenea(Ti,n):

'1' (metodo 1)
Tin = ti2 * t2,3 * t3,4 * ... * tn-1,n

'2' (metodo 2)
idem metodo 1 , porem usando a propriedade de ortonormalidade das colunas

'3' (metodo 3)
Ti,n = t1,2*[t2,3 * ...*[tn-2,tn-1 * tn-1,n]...]

'4' (metodo 4)
idem metodo 3 , porem usando a propriedade de ortonormalidade das colunas

'5' (metodo 5)
idem metodo 1 , porem usando simplificações trigonométricas

É possível então rever o modelo na tela ou voltar à etapa de escolha do método ou do robô e fazer nova geração de um Modelo Geométrico.

3.1.3 - Informações Geradas e Documentação

Após a geração de um Modelo Geométrico serão criados os seguintes arquivos :

- **"nome do robô".mgd**
- **"nome do robô".ea**
- **"nome do robô".mt**
- **"nome do robô".mex**
- **"nome do robô".mel**

Exemplo : Para o arquivo de dados "puma.db" terão sido gerados :

puma.ea; puma.mel; puma.mgd;

puma.mt; puma.mex;

Estes arquivos contêm as seguintes informações :

- "nome do robô".mgd

Este arquivo contém a matriz de transformação homogênea constando de expressões executáveis em linguagem "C" para todos os elementos da matriz chamada T1n[4][4] e também das expressões das equações auxiliares geradas, caso existam. As equações auxiliares são apresentadas na forma :

D[i] = [expressão executável em "C"]

- "nome do robô".mel

Este arquivo grava as matrizes elementares. Apresenta também a matriz de transformação homogênea final obtida , as equações auxiliares geradas e também a contabilidade do número de operações matemáticas necessárias para avaliar numéricamente o modelo.

- "nome do robô".ea

Este arquivo contém as expressões das equações auxiliares geradas durante a obtenção automática do modelo geométrico.

- "nome do robô".mt e "nome do robô".mex

Estes arquivos contêm as expressões dos elementos das matrizes intermediárias. A extensão ".mex" contém as expressões das matrizes intermediárias na forma compilável.

Caso seja necessário obter listagens de algum destes arquivos, isto deve ser feito antes da geração de outro modelo para o **mesmo** arquivo de dados (para outros arquivos isto não é necessário), pois o sistema destrói todo conteúdo anterior destes arquivos antes da geração de um novo modelo baseado no mesmo arquivo de dados. Este procedimento evita o congestionamento da memória de massa.

Nota Importante: Os Modelos Geométricos que servem como dados de entrada para a geração de modelos dinâmicos são os gerados pelos métodos "1", "2" ou "5" apenas. Caso isto não ocorra o Módulo de Geração de Modelos Dinâmicos identificará a imcompatibilidade e solicitará que seja gerado novo arquivo.

3.2 - Geração Automática de Modelo Dinâmico de Referência

O módulo de geração automática de modelos dinâmicos, permite ao usuário gerar equações simbólicas das:

- Forças generalizadas

- Forças de acoplamento inercial

- Forças centrífugas

- Forças de coriolis

- Forças de gravidade

Alem disso, o modelo dinâmico gerado automaticamente, permite a inclusão de forças de perturbações previamente modeladas, como por exemplo o atrito seco, atrito viscoso e etc.

As expressões simbólicas são geradas em código "C" compilável e nelas são adotadas as seguintes convenções:

A[i][j] <==> Termo da linha i e coluna j da matriz de inércia.

COR[i][j] <==> Termo da linha i e coluna j da matriz de coriolis.

CEN[i][j] <==> Termo da linha i e coluna j da matriz de centrífuga.

GRA[i] <==> Termo i do vetor de gravidade.

P[i] <==> Termo i de vetor de perturbações.

FIN[i] <==> Termo de força de inércia na junta i.

TIN[i][j] <==> Termo de acoplamento de inércia entre juntas. Produto da matriz de inércia e vetor de acelerações.

FCOR[i] <==> Força de coriolis na junta i.

FCEN[i] <==> Força centrífuga na junta i.

FGRA[i] <==> Força de gravidade na junta i.

FPER[i] <==> Forças de perturbações na junta i.

FGEN[i] <==> Força generalizada na junta i.

O método de obtenção do modelo dinâmico utiliza uma técnica de simplificação que consiste em eliminar cálculos redundantes através da substituição de toda expressão que indicar uma operação aritmética por uma variável auxiliar. O modelo gerado passa a consistir então de equações auxiliares expressando as substituições feitas. Assim , por exemplo :

Se durante a geração simbólica do modelo surge a expressão parcial :

```
+lxx2*S3+M2*C2
```

ela é então substituída pela variável auxiliar D[i] e acrescida ao modelo final a equação :

```
D[i] = +Ixx2*S3+M2*C2;
```

Embora do ponto de vista de redução do modelo esta técnica deva ser sempre utilizada, para fins de simples visualização das expressões do modelo com objetivos didáticos e/ou comparação com outras referências , é possível obter as expressões simbólicas na sua forma expandida. neste caso os termos são identificados com o uso de parênteses.

A restriçao no caso do modelamento dinâmico sem usar equações auxiliares, é a capacidade de memória da máquina utilizada, visto que as expressões geradas são extensas. Quando a capacidade de memória é insuficiente, a execução é interrompida retornando para o sistema operacional e enviando mensagem elucidativa.

3.2.1 - Inicialização dos Parâmetros Dinâmicos

Ao entrar no sistema será apresentado o seguinte menu :

```
Voce deseja :

        [I]nicializar sistema
        [G]erar modelos
        [A]uxilio ao usuario
        [D]eixar o sistema

Selecione opcao desejada -->
```

Deve-se optar pela função **[I]nicializar sistema**, que entrará no módulo correspondente.

Na Inicialização será apresentado na tela o seguinte menu :

```
Voce deseja inicializar:

            Parametros [G]eometricos
            Parametros [D]inamicos
                       [A]uxilio ao usuario
                       [V]oltar ao nivel anterior

Selecione opcao desejada -->
```

Deve-se selecionar a opção **Parametros [D]inâmicos** para que seja possível entrar com este tipo de informação.

Na inicialização , existe à sua disposição um diretório

básico contendo os arquivos de dados de entrada que já foram criados, ao qual são acrescentados os novos arquivos criados.

Este diretório permite que você use , referenciando-se apenas ao nome do robo , os dados referentes a parâmetros dinâmicos que já tenham sido introduzidos anteriormente. Pode-se listar, apresentar, imprimir, criar ou destruir arquivos. Quando se especifica um dado robô ,o sistema fornece uma relação dos arquivos similares existentes. No caso de não existir nenhum, pode-se criar um novo arquivo, que terá o mesmo nome que voce especificar para o robô, com extensão tipo ".ine".

Exemplo :

Nome = puma
Arquivo criado = puma.ine

O menu apresentado ao usuário neste ponto é o seguinte :

Voce deseja :

[L]istar arquivos do diretorio
[A]presentar arquivo do diretorio
[I]mprimir arquivo do diretorio
[C]riar arquivo do diretorio
[D]eletar arquivos do diretorio
[V]oltar ao nivel anterior

Selecione opcao desejada -->

[L]istar - Mostra todos os arquivos de dados do diretório em questão (sisdir.ine).

[V]isualizar - Apresenta o conteúdo de um arquivo de dados específico.

[I]mprime - Imprime o arquivo de dados desejado. A impressora deve estar conectada e ligada para evitar que o sistema trave.

[C]riar - Para criação de novo arquivo de dados.

[D]eletar - Para eliminar um arquivo de dados específico.

[V]oltar - Volta para o menu anterior.

Todas estas opções funcionam exatamente como no modelo geométrico direto, a única excessão é a função **[C]riar arquivo**. Neste caso deve-se ter atenção para não trocar os dados, pois a entrada é de forma sequencial. Os dados alfanuméricos serão gravados em uma forma padrão e os numéricos serão gravados numéricamente. Os parâmetros dinâmicos que devem ser introduzidos, são pedidos na seguinte ordem:

Ixx - Inércia de rotação do corpo em torno do eixo "x"

Iyy - Inércia de rotação do corpo em torno do eixo "y"

Izz - Inércia de rotação do corpo em torno do eixo "z"

Ixy - Produto de inércia xy

Ixz - Produto de inércia xz

Iyz - produto de inércia yz

rx - Raio de giração do centro de massa em torno do eixo "x"

ry - Raio de giração do centro de massa em torno do eixo "y"
rz - Raio de giração do centro de massa em torno do eixo "z"

M - Massa do corpo

3.2.2 - Geração de Modelos Dinâmicos

Para gerar Modelos Dinâmicos o usuario deve ir até o menu principal e selecionar a opção **[G]erar modelos**. Será então apresentado o seguinte menu :

Voce deseja :

gerar modelo [G]eometrico Direto
gerar modelo [D]inamico de Referencia
[A]uxilio para uso
[V]oltar ao menu anterior

Selecione opcao desejada -->

Para executar a geracão de um modelo dinâmico, o sistema solicita ao usuário o nome do robô cujo modelo se deseja gerar :

Digite apenas o nome do robo manipulador -->

e pede confirmação :

Deseja confirmar o nome do robo? (s/n) -->

é verificado então se os seguintes arquivos estão disponíveis:

- "nome do robô".ine

- "nome do robô".dh

- "nome do robô".mt

- "nome do robô".ea

Existe também uma verificação de consistência entre o número de graus de liberdade dos arquivos "nome do robô".dh e "nome do robô".ine. Em qualquer destas situações o sistema retorna mensagem elucidativa ao usuário. Quando for o caso de não existir o arquivo "nome do robô".ine, o sistema retorna automáticamente para a inicialização para que o usuário possa providenciar a criação do arquivo necessário. Caso exista mais de um arquivo de dados que contenha o nome do robô como parte do seu nome , o sistema solicita ao usuário uma decisão quanto ao arquivo a ser utilizado. Exemplo para o robô "puma" :

```
**********************************************************
                    ATENCAO

Existe(m) o(s) seguinte(s) arquivo(s) puma no diretorio :

     puma.ine
     pumacr.ine

**********************************************************

Selecione o arquivo desejado ( sem extensao ) -->
```

O sistema pergunta se deseja ou não a geração de equações auxiliares:

```
Deseja a geracao de equacoes auxiliares ? (s/n) -->
```

Logo após o sistema entrará na etapa de geração do modelo dinâmico, colocando a seguinte mensagem na tela:

```
     ********************************************************
     *                                                      *
     *  EXECUTANDO GERACAO AUTOMATICA DE MODELO DINAMICO    *
     *                                                      *
     ********************************************************
```

Quando é alcançada a etapa de geração dos termos de gravidade a execução é interrompida e o usuário é questionado se deseja modificar o vetor de gravidade em relação ao sistema de coordenadas de base, pois normalmente este vetor está na direção "Z", mas nada impede de existir um robô cuja base não esteja na posição vertical ou que esteja trabalhando fora da posição vertical, mudando o sentido do vetor de gravidade para as direções "Y" ou "X".

******************** ATENCAO ********************

Considerando o sistema de coordenadas da base,
o vetor gravidade atual esta na direcao 'ZZ'
Deseja modificar este vetor ? (s/n) -->

É então efetuado o processamento simbólico e apresentado ao usuário através da tela os nomes dos arquivos resultantes e a contabilidade das operações matemáticas necessárias para a avaliação do modelo. É possível rever estes resultados ou retornar ao menu de geração de modelos.

3.2.3 - Informações Geradas e Documentação

Após o processamento do modelo dinâmico é dada a informação do número de operações para avaliar o modelo matematicamente, número de equações auxiliares geradas e dos arquivos gerados durante este processo:

- "nome do robô".sim

Este arquivo contém todas as equações necessárias para simular os efeitos dinâmicos do robô de forma separada (inércia, centrífuga, coriolis ou gravidade) ou simular somente as forças generalizadas.

- "nome do robô".din

Este arquivo contém todas as informações referentes aos dados de entrada (tabelas contendo parâmetros geométricos e dinâmicos) e os resultados gerados, que são os termos das matrizes de inércia, coriolis, centrífuga, gravidade e perturbações.

No final deste arquivo é gravado o número de operações e o número de senos e/ou cossenos necessários para avaliar o modelo.

Este arquivo também pode ser usado para simulação, mas a sua principal finalidade é a documentação.

- "nome do robô".eac

Neste arquivo são armazenadas as equações auxiliares necessárias para avaliar o modelo numericamente.

Para cada 260 equações auxiliares do modelo, é gerado automaticamente um arquivo "nome do robô*".eac, como mostrado abaixo:

"nome-do-roboa1".eac
"nome-do-roboa2".eac
.
.
.
"nome-do-roboan".eac

São gerados estes vários arquivos, devido à limitações do PC-XT.

3.3 - Ferramenta de Auxílio à Obtenção do Modelo Geometrico Inverso

A partir dos parâmetros geométricos (convenção de Denavit-Hartemberg), são geradas automaticamente expressões nos modos pré-multiplicado e pós-multiplicado, explicadas mais detalhadamente no anexo (5.3).

Existe uma função de varredura das expressões para localizar as coordenadas ainda não resolvidas e montar uma cadeia de caracteres com os números destas ligações.

Esta ferramenta gera listagens com todos os elementos das duas matrizes equivalentes colocados lado a lado, mais a cadeia de caracteres com as coordenadas não resolvidas. Isto facilita enormemente a visualização dos elementos para dedução das heurísticas de seleção e manipulação dos mesmos.

Para deteção dos pontos de singularidade e seu tratamento, a ferramenta se mostrou muito prática, bastando fixar convenientemente as coordenadas generalizadas rotacionais no arquivo de parâmetros de Denavit-Hartemberg.

3.3.1 - Inicialização de Parâmetros Geométricos

A inicialização de parâmetros geométricos para a obtenção do Modelo Geométrico Inverso, é idêntica à inicialização de parâmetros para o Modelo Geométrico Direto. Quando o usuário faz a opção de inicializar parâmetros geométricos, é criada uma base de dados comum que pode ser acessada tanto no modelamento geométrico direto, quanto no modelamento inverso.

3.3.2 - Geração de Equações de Auxílio à Obtenção do Modelo Geométrico Inverso

Para gerar Equações de Auxílio à Obtenção do Modelo Inverso, o usuário deve ter as opções do menu principal e selecionar a opção **[G]erar modelos**. Será então apresentado o seguinte menu :

Voce deseja :

gerar modelo [G]eometrico Direto
gerar modelo [I]nverso
gerar modelo [D]inâmico de Referência
[A]uxilio para uso
[V]oltar ao menu anterior

Selecione opcao desejada -->

Selecionando a opção **gerar modelo [I]nverso**, é solicitado ao usuário o nome do arquivo que contém os parâmetros geométricos (convenção de Denavit-Hartemberg) do robô cujo modelo se deseja gerar :

Digite apenas o nome do robo manipulador -->

e pede confirmação do nome :

Deseja confirmar o nome do robo? (s/n) -->

é verificado então se o arquivo de dados "nome do robo".dh está disponível. Este arquivo é o que contêm os parâmetros geométricos (convenção de Denavit-Hartemberg) do robô escolhido. Em caso negativo a execução não é possível e o sistema retorna automáticamente para a Inicialização, para que se possa providenciar a criação do arquivo necessário.

Em seguida o sistema começa a calcular as expressões mencionadas anteriormente.

3.3.3 - Informações Geradas e Documentação

Após a execução terão sido gerados os arquivos:

"nome do robô".gi1

Este arquivo contém todas as equações da pré-multiplicação

"nome do robô".gi2

Este arquivo contém todas as equações da pós-multiplicação

4 - Saída do sistema

Para sair do sistema, estando em qualquer nível do menu , é necessário voltar nível por nível até o menu principal onde existe a opção **[D]eixar o sistema**.

Após sair do sistema o usuário deve solicitar as impressões dos arquivos que necessitar para documentação.

Se o usuário tentar gerar resultados com os arquivos de inicialização de um robô cujo modelo já foi obtido, haverá a destruição dos arquivos correspondentes a este modelo, substituindo-os pelos novos resultados obtidos.

5 - Anexos

5.1 - Matriz de Passagem Homogênea Elementar e Parâmetros Geométricos

As matrizes de passagem homogêneas elementares, representam uma sequência específica de transformações às quais deve ser submetido o sistema de coordenadas de um elemento para coincidir com o sistema de coordenadas do elemento subsequente na cadeia cinemática do robô. Embora isto possa ser feito de diversas maneiras, mostraremos somente a convenção definida por [Denavit 55] ,visto que esta gera matrizes com grande número de termos nulos ou unitários. Ela é conhecida como convenção de Denavit-Hartemberg e consiste na seguinte sequência de operações:

i - rotação de um ângulo "teta" em torno do eixo "z"

ii - translação de uma distância "d" ao longo do eixo "z" resultante

iii- rotação de um ângulo "alfa" em torno do eixo "x" resultante

iv - translação de uma distância "a" ao longo do eixo "x" resultante

Os parâmetros de rotação , "alfa" e "teta" , associados aos de translação, "a" e "d" , são conhecidos como os parâmetros geométricos obtidos pela convenção de Denavit-Hartemberg e estão ilustrados na figura (A-1).

A sequência de operações descrita é equivalente à uma matriz com a seguinte forma geral :

$$
\begin{bmatrix}
\cos(teta) & -\text{sen}(teta).\cos(alfa) & \text{sen}(teta).\text{sen}(alfa) & a.\cos(teta) \\
\text{sen}(teta) & \cos(teta).\cos(alfa) & -\cos(teta).\text{sen}(alfa) & a.\text{sen}(teta) \\
0 & \text{sen}(alfa) & \cos(alfa) & d \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

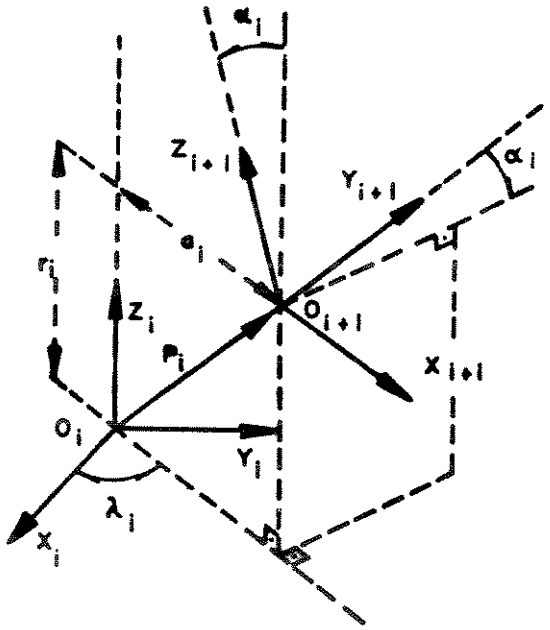


Figura (A-1) - Parâmetros de Denavit-Hartemberg

5.2 - Resultados do Modelamento Geométrico

Como exemplo da grande variação do número de operações necessárias para avaliação do modelo geométrico em função do método usado, ilustramos os resultados obtidos para o robô Puma 560 :

método	Var. Aux.	número de operações				
D-H	s/n	*	+	-	Total	sin/cos
1	s	70	25	11	106	12
1	n	205	40	37	282	12
2	s	71	23	13	107	12
2	n	1299	306	175	1780	12
3	s	70	20	14	104	12
3	n	131	41	36	208	12
4	s	62	19	12	93	12
4	n	253	66	79	253	12
5	s	53	21	07	81	14

5.3 - Pré-multiplicação e Pós-multiplicação [Paul 81]

É baseado na utilização das matrizes de passagem homogêneas elementares do robô. Com estas matrizes, suas inversas e mais a matriz literal montada com a posição e orientação genéricas do órgão terminal (matriz de passagem final), são construídas duas matrizes de passagem intermediárias equivalentes entre cada ligação e o órgão terminal determinadas das seguintes maneiras:

- produto das matrizes de passagem elementares da ligação em questão e o órgão terminal.

$$T(i,i+1) * T(i+1,i+2) * \ldots * T(n-1,n)$$

- pré-multiplicação da matriz de passagem final pelas inversas das matrizes de passagem elementares das juntas anteriores à junta em questão.

$$T(i-1,i)^{-1} * T(i-2,i-1)^{-1} * \ldots * T(base,1)^{-1} * T(base,n)$$

Exemplo para um robô de seis ligações:

base: $T(base,1)*T(1,2)*T(2,3)*T(3,4)*T(4,5)*T(5,6) = T(base,6)$

junta 1: $T(1,2)*T(2,3)*T(3,4)*T(4,5)*T(5,6) = T(base,1)^{-1}*T(base,6)$

junta 2: $T(2,3)*T(3,4)*T(4,5)*T(5,6) = T(1,2)^{-1}*T(base,1)^{-1}*T(base,6)$

junta 3: $T(3,4)*T(4,5)*T(5,6) = T(2,3)^{-1}*T(1,2)^{-1}*T(base,1)^{-1}*T(base,6)$

junta 4: $T(4,5)*T(5,6) = T(3,4)^{-1}*T(2,3)^{-1}*T(1,2)^{-1}*T(base,1)^{-1}*T(base,6)$

junta 5: $T(5,6) = T(4,5)^{-1}*T(3,4)^{-1}*T(2,3)^{-1}*T(1,2)^{-1}*T(base,1)^{-1}*T(base,6)$

Se for possível a obtenção analítica do modelo geométrico inverso do robô, as expressões acima podem fornecer as coordenadas generalizadas diretamente ou mediante alguma manipulação algébrica.

Em geral para uma ligação rotacional, o cálculo da coordenada generalizada passa pela resolução de uma equação do tipo a*cos(tet) + b*sin(tet) = c. Deve-se utilizar a função que obtém o arco-tangente no quadrante especificado pelos senos e cossenos do ângulo. As funções arco-seno e arco-cosseno não devem ser utilizadas por problemas de precisão e indefinição do quadrante.

A pós-multiplicação é análoga à pré-multiplicação, ou seja, no caso da equação da junta 1 ,teríamos a seguinte configuração da igualdade:

junta 1: $T(base,1)*T(1,2)*T(2,3)*T(3,4)*T(4,5) = T(base,6)*T(5,6)^{-1}$

E assim sucessivamente para as outras juntas.

Através de análises feitas verificou-se que em muitos casos, com a pós-multiplicação podemos obter expressões mais simples, apesar da bibliografia clássica não fazer referências à este fato.

6 - Bibliografia

[Craig 85] J.J.Craig, "Introduction to Robotics - Mechanism and Control". Addison-Wesley Publishing Company, 1985.

[Denavit 55] J. Denavit, R. S. Hartemberg. " A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices". J. Appl. Mech. 77(2), 215-221 (June 1955).

[Gorla 84] B. Gorla, M. Renaud, "Modèles des Robots Manipulateurs - Application à leur Commande". Cepadues Editions, Toulouse, 1984.

[Megahed 84] S. Megahed, "Contribution a la Modelisation Geometrique et Dynamique des Robots Manipulateurs a Structure de Chaine Cinematique Simple ou Complexe". Th. Doctorat d'Etat, Automatique, Toulouse III, France, 1984.

[Paul 81] R. Paul, "Robot Manipulators: Mathematics, Programming and Control". MIT Press, Cambridge, Massachussetts and London, England, 1981.

[Ramos 86] J.J.G. Ramos, "Geração de Trajetórias Contínuas para Robôs Manipuladores: Aspectos Cinemáticos e Computacionais". Tese de Mestrado, Unicamp 1986.

ANEXO 2

ESTUDO DAS SIMPLIFICAÇÕES TRIGONOMÉTRICAS

ESTUDO DAS SIMPLIFICAÇÕES TRIGONOMÉTRICAS

1) Caso em que $\alpha_1 = \alpha_2 = \alpha_3 = 0$ e $\alpha_4 = 70^\circ$ (substituindo um ângulo qualquer, já que no GMROB os valores de α têm que ser numéricos):

Obs 1: 0.3420 representa $C\alpha_4$ e 0.9396 representa $S\alpha_4$;

Obs 2: Apenas estão apresentadas as simplificações dos termos de rotação, já que as de translação se obtêm destas.

```
t1n[0][0] = (+(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*C4+
             +(-(+C1*C2-S1*S2)*S3+(-C1*S2-S1*C2)*C3)*S4);
```

$= C_{1234}$

```
t1n[0][1] = (-(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*S4*0.3420+
             +(-(+C1*C2-S1*S2)*S3+(-C1*S2-S1*C2)*C3)*C4*0.3420);
```

$= -S_{1234}C\alpha_4$

```
t1n[0][2] = (+(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*S4*0.9396+
             -(-(+C1*C2-S1*S2)*S3+(-C1*S2-S1*C2)*C3)*C4*0.9396);
```

$= S_{1234}S\alpha_4$

```
t1n[0][3] = (+(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*a4*C4+
             +(-(+C1*C2-S1*S2)*S3+(-C1*S2-S1*C2)*C3)*a4*S4+
             +(+(+C1*C2-S1*S2)*a3*C3+(-C1*S2-S1*C2)*a3*S3+
             +(+C1*a2*C2-S1*a2*S2+a1*C1)));
t1n[1][0] = (+(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*C4+
             +(-(+S1*C2+C1*S2)*S3+(-S1*S2+C1*C2)*C3)*S4);
```

$= S_{1234}$

```
t1n[1][1] = (-(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*S4*0.3420+
             +(-(+S1*C2+C1*S2)*S3+(-S1*S2+C1*C2)*C3)*C4*0.3420);
```

$C_{1234}C\alpha_4$

```
t1n[1][2] = (+(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*S4*0.9396+
             -(-(+S1*C2+C1*S2)*S3+(-S1*S2+C1*C2)*C3)*C4*0.9396);
```

$-C_{1234}S\alpha_4$

```
t1n[1][3] = (+(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*a4*C4+
             +(-(+S1*C2+C1*S2)*S3+(-S1*S2+C1*C2)*C3)*a4*S4+
             +(+(+S1*C2+C1*S2)*a3*C3+(-S1*S2+C1*C2)*a3*S3+
             +(+S1*a2*C2+C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = 0.9396;
```

$= S\alpha_4$

```
t1n[2][2] = 0.3420;
```

$= C\alpha_4$

```
t1n[2][3] = (+d4+(+d3+(+d2+d1)));
```

2) Caso em que $\alpha_1 = \pi$, $\alpha_2 = \alpha_3 = 0$ e $\alpha_4 = 70^o$

```
t1n[0][0] = (+(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*C4+
            +(-(+C1*C2+S1*S2)*S3+(-C1*S2+S1*C2)*C3)*S4);
                                                    = C1-2-3-4
t1n[0][1] = (-(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*S4*0.3420+
            +(-(+C1*C2+S1*S2)*S3+(-C1*S2+S1*C2)*C3)*C4*0.3420);
                                                    = S1-2-3-4Cα4
t1n[0][2] = (+(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*S4*0.9396+
            -(-(+C1*C2+S1*S2)*S3+(-C1*S2+S1*C2)*C3)*C4*0.9396);
                                                    = -S1-2-3-4Sα4
t1n[0][3] = (+(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*a4*C4+
            +(-(+C1*C2+S1*S2)*S3+(-C1*S2+S1*C2)*C3)*a4*S4+
            +(+(+C1*C2+S1*S2)*a3*C3+(-C1*S2+S1*C2)*a3*S3+
            +(+C1*a2*C2+S1*a2*S2+a1*C1)));
t1n[1][0] = (+(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*C4+
            +(-(+S1*C2-C1*S2)*S3+(-S1*S2-C1*C2)*C3)*S4);
                                                    = S1-2-3-4
t1n[1][1] = (-(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*S4*0.3420+
            +(-(+S1*C2-C1*S2)*S3+(-S1*S2-C1*C2)*C3)*C4*0.3420);
                                                    = -C1-2-3-4Cα4
t1n[1][2] = (+(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*S4*0.9396+
            -(-(+S1*C2-C1*S2)*S3+(-S1*S2-C1*C2)*C3)*C4*0.9396);
                                                    = C1-2-3-4Sα4
t1n[1][3] = (+(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*a4*C4++
             (-(+S1*C2-C1*S2)*S3+(-S1*S2-C1*C2)*C3)*a4*S4++
             (+(+S1*C2-C1*S2)*a3*C3+(-S1*S2-C1*C2)*a3*S3+
             +(+S1*a2*C2-C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
                                                    = 0
t1n[2][1] = -0.9396;
                                                    = -Sα4
t1n[2][2] = -0.3420;
                                                    = -Cα4
t1n[2][3] = (-d4+(-d3+(-d2+d1)));
```

3) Caso em que $\alpha_1 = 0$, $\alpha_2 = \pi$, $\alpha_3 = 0$ e $\alpha_4 = 70^{\circ}$

```
t1n[0][0] = (+(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*C4+
            +(-(+C1*C2-S1*S2)*S3+(+C1*S2+S1*C2)*C3)*S4);
```

$= C_{12-3-4}$

```
t1n[0][1] = (-(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*S4*0.3420+
            +(-(+C1*C2-S1*S2)*S3+(+C1*S2+S1*C2)*C3)*C4*0.3420);
```

$= S_{12-3-4}C_{\alpha_4}$

```
t1n[0][2] = (+(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*S4*0.9396+
            -(-(+C1*C2-S1*S2)*S3+(+C1*S2+S1*C2)*C3)*C4*0.9396);
```

$= -S_{12-3-4}S_{\alpha_4}$

```
t1n[0][3] = (+(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*a4*C4+
            +(-(+C1*C2-S1*S2)*S3+(+C1*S2+S1*C2)*C3)*a4*S4+
            +(+(+C1*C2-S1*S2)*a3*C3+(+C1*S2+S1*C2)*a3*S3+
            +(+C1*a2*C2-S1*a2*S2+a1*C1)));
t1n[1][0] = (+(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*C4+
            +(-(+S1*C2+C1*S2)*S3+(+S1*S2-C1*C2)*C3)*S4);
```

$= S_{12-3-4}$

```
t1n[1][1] = (-(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*S4*0.3420+
            +(-(+S1*C2+C1*S2)*S3+(+S1*S2-C1*C2)*C3)*C4*0.3420);
```

$= -C_{12-3-4}C_{\alpha_4}$

```
t1n[1][2] = (+(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*S4*0.9396+
            -(-(+S1*C2+C1*S2)*S3+(+S1*S2-C1*C2)*C3)*C4*0.9396);
```

$= C_{12-3-4}S_{\alpha_4}$

```
t1n[1][3] = (+(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*a4*C4+
            +(-(+S1*C2+C1*S2)*S3+(+S1*S2-C1*C2)*C3)*a4*S4+
            +(+(+S1*C2+C1*S2)*a3*C3+(+S1*S2-C1*C2)*a3*S3+
            +(+S1*a2*C2+C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = -0.9396;
```

$= -S_{\alpha_4}$

```
t1n[2][2] = -0.3420;
```

$= -C_{\alpha_4}$

```
t1n[2][3] = (-d4+(-d3+(+d2+d1)));
```

4) Caso em que $\alpha_1 = \alpha_2 = 0$, $\alpha_3 = \pi$ e $\alpha_4 = 70^\circ$

```
t1n[0][0] = (+(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*C4+
            +(+(+C1*C2-S1*S2)*S3-(-C1*S2-S1*C2)*C3)*S4);
```

$= C_{123-4}$

```
t1n[0][1] = (-(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*S4*0.3420+
            +(+(+C1*C2-S1*S2)*S3-(-C1*S2-S1*C2)*C3)*C4*0.3420);
```

$= S_{123-4}C_{\alpha_4}$

```
t1n[0][2] = (+(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*S4*0.9396+
            -(+(+C1*C2-S1*S2)*S3-(-C1*S2-S1*C2)*C3)*C4*0.9396);
```

$= -S_{123-4}S_{\alpha_4}$

```
t1n[0][3] = (+(+(+C1*C2-S1*S2)*C3+(-C1*S2-S1*C2)*S3)*a4*C4+
            +(+(+C1*C2-S1*S2)*S3-(-C1*S2-S1*C2)*C3)*a4*S4+
            +(+(+C1*C2-S1*S2)*a3*C3+(-C1*S2-S1*C2)*a3*S3+
            +(+C1*a2*C2-S1*a2*S2+a1*C1)));
```

```
t1n[1][0] = (+(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*C4+
            +(+(+S1*C2+C1*S2)*S3-(-S1*S2+C1*C2)*C3)*S4);
```

$= S_{123-4}$

```
t1n[1][1] = (-(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*S4*0.3420+
            +(+(+S1*C2+C1*S2)*S3-(-S1*S2+C1*C2)*C3)*C4*0.3420);
```

$= -C_{123-4}C_{\alpha_4}$

```
t1n[1][2] = (+(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*S4*0.9396+
            -(+(+S1*C2+C1*S2)*S3-(-S1*S2+C1*C2)*C3)*C4*0.9396);
```

$= C_{123-4}S_{\alpha_4}$

```
t1n[1][3] = (+(+(+S1*C2+C1*S2)*C3+(-S1*S2+C1*C2)*S3)*a4*C4+
            +(+(+S1*C2+C1*S2)*S3-(-S1*S2+C1*C2)*C3)*a4*S4+
            +(+(+S1*C2+C1*S2)*a3*C3+(-S1*S2+C1*C2)*a3*S3+
            +(+S1*a2*C2+C1*a2*S2+a1*S1)));
```

```
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = -0.9396;
```

$= -S_{\alpha_4}$

```
t1n[2][2] = -0.3420;
```

$= -C_{\alpha_4}$

```
t1n[2][3] = (-d4+(+d3+(+d2+d1)));
```

5) Caso em que $\alpha_1 = \alpha_2 = \alpha_3 = \pi$ e $\alpha_4 = 70^\circ$

```
t1n[0][0] = (+(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*C4+
            +(+(+C1*C2+S1*S2)*S3-(+C1*S2-S1*C2)*C3)*S4);
```

$= C_{1-23-4}$

```
t1n[0][1] = (-(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*S4*0.3420+
            +(+(+C1*C2+S1*S2)*S3-(+C1*S2-S1*C2)*C3)*C4*0.3420);
```

$= S_{1-23-4}C_{\alpha_4}$

```
t1n[0][2] = (+(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*S4*0.9396+
            -(+(+C1*C2+S1*S2)*S3-(+C1*S2-S1*C2)*C3)*C4*0.9396);
```

$= -S_{1-23-4}S_{\alpha_4}$

```
t1n[0][3] = (+(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*a4*C4+
            +(+(+C1*C2+S1*S2)*S3-(+C1*S2-S1*C2)*C3)*a4*S4+
            +(+(+C1*C2+S1*S2)*a3*C3+(+C1*S2-S1*C2)*a3*S3+
            +(+C1*a2*C2+S1*a2*S2+a1*C1)));
t1n[1][0] = (+(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*C4+
            +(+(+S1*C2-C1*S2)*S3-(+S1*S2+C1*C2)*C3)*S4);
```

$= S_{1-23-4}$

```
t1n[1][1] = (-(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*S4*0.3420+
            +(+(+S1*C2-C1*S2)*S3-(+S1*S2+C1*C2)*C3)*C4*0.3420);
```

$= -C_{1-23-4}C_{\alpha_4}$

```
t1n[1][2] = (+(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*S4*0.9396+
            -(+(+S1*C2-C1*S2)*S3-(+S1*S2+C1*C2)*C3)*C4*0.9396);
```

$= C_{1-23-4}S_{\alpha_4}$

```
t1n[1][3] = (+(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*a4*C4+
            +(+(+S1*C2-C1*S2)*S3-(+S1*S2+C1*C2)*C3)*a4*S4+
            +(+(+S1*C2-C1*S2)*a3*C3+(+S1*S2+C1*C2)*a3*S3+
            +(+S1*a2*C2-C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = -0.9396;
```

$= -S_{\alpha_4}$

```
t1n[2][2] = -0.3420;
```

$= -C_{\alpha_4}$

```
t1n[2][3] = (-d4+(+d3+(-d2+d1)));
```

6) Caso em que $\alpha_1 = \alpha_2 = \pi$, $\alpha_3 = 0$ e $\alpha_4 = 70^o$

```
t1n[0][0] = C+(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*C4+
            +(-(+C1*C2+S1*S2)*S3+(+C1*S2-S1*C2)*C3)*S4);
```

$= C_{1-234}$

```
t1n[0][1] = C-(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*S4*0.3420+
            +(-(+C1*C2+S1*S2)*S3+(+C1*S2-S1*C2)*C3)*C4*0.3420);
```

$= -S_{1-234}C_{\alpha 4}$

```
t1n[0][2] = C+(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*S4*0.9396+
            -(-(+C1*C2+S1*S2)*S3+(+C1*S2-S1*C2)*C3)*C4*0.9396);
```

$= S_{1-234}S_{\alpha 4}$

```
t1n[0][3] = C+(+(+C1*C2+S1*S2)*C3+(+C1*S2-S1*C2)*S3)*a4*C4+
            +(-(+C1*C2+S1*S2)*S3+(+C1*S2-S1*C2)*C3)*a4*S4+
            +(+(+C1*C2+S1*S2)*a3*C3+(+C1*S2-S1*C2)*a3*S3+
            +(+C1*a2*C2+S1*a2*S2+a1*C1)));
t1n[1][0] = C+(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*C4+
            +(-(+S1*C2-C1*S2)*S3+(+S1*S2+C1*C2)*C3)*S4);
```

$= S_{1-234}$

```
t1n[1][1] = C-(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*S4*0.3420+
            +(-(+S1*C2-C1*S2)*S3+(+S1*S2+C1*C2)*C3)*C4*0.3420);
```

$= C_{1-234}C_{\alpha 4}$

```
t1n[1][2] = C+(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*S4*0.9396+
            -(-(+S1*C2-C1*S2)*S3+(+S1*S2+C1*C2)*C3)*C4*0.9396);
```

$= -C_{1-234}S_{\alpha 4}$

```
t1n[1][3] = C+(+(+S1*C2-C1*S2)*C3+(+S1*S2+C1*C2)*S3)*a4*C4+
            +(-(+S1*C2-C1*S2)*S3+(+S1*S2+C1*C2)*C3)*a4*S4+
            +(+(+S1*C2-C1*S2)*a3*C3+(+S1*S2+C1*C2)*a3*S3+
            +(+S1*a2*C2-C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = 0.9396;
```

$= S_{\alpha 4}$

```
t1n[2][2] = 0.3420;
```

$= C_{\alpha 4}$

```
t1n[2][3] = C+d4+(+d3+(-d2+d1)));
```

7) Caso em que $\alpha_1 = \pi$, $\alpha_2 = 0$, $\alpha_3 = \pi$ e $\alpha_4 = 70^o$

```
t1n[0][0] = (+(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*C4+
            +(+(+C1*C2+S1*S2)*S3-(-C1*S2+S1*C2)*C3)*S4);
```

$= C_{1-2-3-4}$

```
t1n[0][1] = (-(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*S4*0.3420+
            +(+(+C1*C2+S1*S2)*S3-(-C1*S2+S1*C2)*C3)*C4*0.3420);
```

$= -S_{1-2-3-4}C_{\alpha_4}$

```
t1n[0][2] = (+(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*S4*0.9396+
            -(+(+C1*C2+S1*S2)*S3-(-C1*S2+S1*C2)*C3)*C4*0.9396);
```

$= S_{1-2-3-4}S_{\alpha_4}$

```
t1n[0][3] = (+(+(+C1*C2+S1*S2)*C3+(-C1*S2+S1*C2)*S3)*a4*C4+
            +(+(+C1*C2+S1*S2)*S3-(-C1*S2+S1*C2)*C3)*a4*S4+
            +(+(+C1*C2+S1*S2)*a3*C3+(-C1*S2+S1*C2)*a3*S3+
            +(+C1*a2*C2+S1*a2*S2+a1*C1)));
t1n[1][0] = (+(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*C4+
            +(+(+S1*C2-C1*S2)*S3-(-S1*S2-C1*C2)*C3)*S4);
```

$= S_{1-2-3-4}$

```
t1n[1][1] = (-(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*S4*0.3420+
            +(+(+S1*C2-C1*S2)*S3-(-S1*S2-C1*C2)*C3)*C4*0.3420);
```

$= C_{1-2-3-4}C_{\alpha_4}$

```
t1n[1][2] = (+(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*S4*0.9396+
            -(+(+S1*C2-C1*S2)*S3-(-S1*S2-C1*C2)*C3)*C4*0.9396);
```

$= -C_{1-2-3-4}S_{\alpha_4}$

```
t1n[1][3] = (+(+(+S1*C2-C1*S2)*C3+(-S1*S2-C1*C2)*S3)*a4*C4+
            +(+(+S1*C2-C1*S2)*S3-(-S1*S2-C1*C2)*C3)*a4*S4+
            +(+(+S1*C2-C1*S2)*a3*C3+(-S1*S2-C1*C2)*a3*S3+
            +(+S1*a2*C2-C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = 0.9396;
```

$= S_{\alpha_4}$

```
t1n[2][2] = 0.3420;
```

$= C_{\alpha_4}$

```
t1n[2][3] = (+d4+(-d3+(-d2+d1)));
```

8) Caso em que $\alpha_1 = 0$, $\alpha_2 = \alpha_3 = \pi$ e $\alpha_4 = 70^\circ$

```
t1n[0][0] = (+(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*C4+
            +(+(+C1*C2-S1*S2)*S3-(+C1*S2+S1*C2)*C3)*S4);
```

$= C_{12-34}$

```
t1n[0][1] = (-(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*S4*0.3420+
            +(+(+C1*C2-S1*S2)*S3-(+C1*S2+S1*C2)*C3)*C4*0.3420);
```

$= -S_{12-34}C_{\alpha 4}$

```
t1n[0][2] = (+(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*S4*0.9396+
            -(+(+C1*C2-S1*S2)*S3-(+C1*S2+S1*C2)*C3)*C4*0.9396);
```

$= S_{12-34}S_{\alpha 4}$

```
t1n[0][3] = (+(+(+C1*C2-S1*S2)*C3+(+C1*S2+S1*C2)*S3)*a4*C4+
            +(+(+C1*C2-S1*S2)*S3-(+C1*S2+S1*C2)*C3)*a4*S4+
            +(+(+C1*C2-S1*S2)*a3*C3+(+C1*S2+S1*C2)*a3*S3+
            +(+C1*a2*C2-S1*a2*S2+a1*C1)));
t1n[1][0] = (+(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*C4+
            +(+(+S1*C2+C1*S2)*S3-(+S1*S2-C1*C2)*C3)*S4);
```

$= S_{12-34}$

```
t1n[1][1] = (-(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*S4*0.3420+
            +(+(+S1*C2+C1*S2)*S3-(+S1*S2-C1*C2)*C3)*C4*0.3420);
```

$= C_{12-34}C_{\alpha 4}$

```
t1n[1][2] = (+(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*S4*0.9396+
            -(+(+S1*C2+C1*S2)*S3-(+S1*S2-C1*C2)*C3)*C4*0.9396);
```

$= -C_{12-34}S_{\alpha 4}$

```
t1n[1][3] = (+(+(+S1*C2+C1*S2)*C3+(+S1*S2-C1*C2)*S3)*a4*C4+
            +(+(+S1*C2+C1*S2)*S3-(+S1*S2-C1*C2)*C3)*a4*S4+
            +(+(+S1*C2+C1*S2)*a3*C3+(+S1*S2-C1*C2)*a3*S3+
            +(+S1*a2*C2+C1*a2*S2+a1*S1)));
t1n[2][0] = 0;
```

$= 0$

```
t1n[2][1] = 0.9396;
```

$= S_{\alpha 4}$

```
t1n[2][2] = 0.3420;
```

$= C_{\alpha 4}$

```
t1n[2][3] = (+d4+(-d3+(+d2+d1)));
```

ANEXO 3

PROGRAMAS DESENVOLVIDOS NAS IIIª E IVª EBAI

```
/*****************************************************************
 *****************************************************************
 **********                                             **********
 **********  VALIDACION DEL MODELO GEOMETRICO INVERSO   **********
 **********                                             **********
 **********            DEL ROBOT  'TH-8'                **********
 **********                                             **********
 *****************************************************************
 *****************************************************************
 **********                                             **********
 **********  AUTORES:                                   **********
 **********                                             **********
 **********         Paulo  Renato Battaglia             **********
 **********         Daniel Juan Franco                  **********
 **********         Domingos S. L. Simonetti            **********
 **********                                             **********
 **********         III EBAI - Curitiba,  02-02-1988    **********
 **********                                             **********
 *****************************************************************
 *****************************************************************/



#include        <math.h>

#define         LINT_ARGS       1
#define         BEEP            printf( "\7" )
#define         CLS             printf( "[2J" )
#define         PI              3.141592654

#define         Q1MINIMO        -160*PI/180         /* Minimo tita1 */
#define         Q1MAXIMO         160*PI/180         /* Maximo tita1 */
#define         Q2MINIMO        0.2                 /* Minimo d2    */
#define         Q2MAXIMO        0.7                 /* Maximo d2    */
#define         Q3MINIMO        -0.7                /* Minimo d3    */
#define         Q3MAXIMO        -0.2                /* Maximo d3    */
#define         Q4MINIMO        -160*PI/180         /* Minimo tita4 */
#define         Q4MAXIMO         160*PI/180         /* Maximo tita4 */
#define         Q5MINIMO        -160*PI/180         /* Minimo tita5 */
#define         Q5MAXIMO         160*PI/180         /* Maximo tita5 */
#define         Q6MINIMO        -160*PI/180         /* Minimo tita6 */
#define         Q6MAXIMO         160*PI/180         /* Maximo tita6 */
#define         A2              0.051               /* Valor de 'a2'*/
#define         D6              0.291               /* Valor de 'd6'*/
#define         INTENTOS        5



typedef struct operacionales { double nx, ox, ax, px;
```

```
                              double ny, oy, ay, py;
                              double nz, oz, az, pz;
                             };

typedef struct generalizadas { double teta1, teta4, teta5, teta6;
                              double d2, d3;
                             };


double a2 = A2, d6 = D6;       /* Distancias constantes del Robot 'TH-8' */


void   mod_directo( struct generalizadas *,
                    struct operacionales * );
int    mod_inverso( struct generalizadas *,
                    struct operacionales *,
                    struct generalizadas * );
int    copara_resultados( struct generalizadas *, struct generalizadas * );
double square( double );
void   clr_locate( int, int );
void   imprime_errores( struct generalizadas *,
                        struct generalizadas *,
                        int, float, float,
                        struct operacionales * );


/* ------------------------------------------------------------------ */


main()

/*
 * Validacion del Modelo Geometrico inverso del Robot 'TH-8'.
 *
 * Devuelve (0) si no hubo errores.
 * Devuelve (1) si hubo algun error.
 * Devuelve (2) si se selecciono la opcion de 'EXIT'.
 *
 */


{
   struct generalizadas lnew;    /* Nuevas posiciones de las articulaciones */
   struct generalizadas * new;   /* Nuevas posiciones de las articulaciones */
   struct generalizadas lold;    /* Viejas posiciones de las articulaciones */
   struct generalizadas * old;   /* Viejas posiciones de las articulaciones */
   struct operacionales lmatriz; /* Matriz de coordenadas operacionales     */
   struct operacionales * matriz;/* Matriz de coordenadas operacionales     */
   double qmin[7], qmax[7];      /* Valores limites de recorrido            */
   float  auxiliar;              /* Variable auxiliar para ingreso de datos */
   float  casos_totales = 1.0;   /* Total de casos para analizar            */
```

```
    float   casos_analizados = 0.0;/* Total de casos analizados al momento    */
    int     i, j, k, l, m, n;      /* Contadores de iteraciones               */
    int     c;                     /* Valor de una tecla presionada           */
    int     intentos = INTENTOS;
    int     item, errores = 0;

    new = &lnew;
    old = &lold;
    matriz = &lmatriz;

/*
 * Valores de Default para los limites de las articulaciones.
 */

    qmin[1] = Q1MINIMO;
    qmax[1] = Q1MAXIMO;
    qmin[2] = Q2MINIMO;
    qmax[2] = Q2MAXIMO;
    qmin[3] = Q3MINIMO;
    qmax[3] = Q3MAIMO;
    qmin[4] = Q4MINIMO;
    qmax[4] = Q4MAXIMO;
    qmin[5] = Q5MINIMO;
    qmax[5] = Q5MAXIMO;
    qmin[6] = Q6MINIMO;
    qmax[6] = Q6MAXIMO;


/*
 * Ingreso de datos para la corrida del programa.
 */

    CLS;
    clr_locate( 1, 5 );
    puts( "VALIDACION DEL MODELO GEOMETRICO INVERSO" );
    clr_locate( 2, 5 );
    puts( "========== === ====== ========== =======" );
    clr_locate( 4, 5 );
    puts( "         DEL ROBOT 'TH-8' " );
    clr_locate( 5, 5 );
    puts( "          === =====  ====  " );


/*
 * Impresion de los parametros de corrida del programa.
 */

    clr_locate( 8, 1 );
    printf( " 1 - Minimo valor de Thita 1: %f\n", qmin[1]*180.0/PI );
    printf( " 2 - Maximo valor de Thita 1:  %f\n", qmax[1]*180.0/PI );
    printf( " 3 - Minimo valor de d2     :  %f\n", qmin[2] );
```

```
    printf( " 4 - Maximo valor de d2      :  %f\n", qmax[2] );
    printf( " 5 - Minimo valor de d3      : %f\n", qmin[3] );
    printf( " 6 - Maximo valor de d3      : %f\n", qmax[3] );
    printf( " 7 - Minimo valor de Thita 4: %f\n", qmin[4]*180.0/PI );
    printf( " 8 - Maximo valor de Thita 4:  %f\n", qmax[4]*180.0/PI );
    printf( " 9 - Minimo valor de Thita 5: %f\n", qmin[5]*180.0/PI );
    printf( "10 - Maximo valor de Thita 5:  %f\n", qmax[5]*180.0/PI );
    printf( "11 - Minimo valor de Thita 6: %f\n", qmin[6]*180.0/PI );
    printf( "12 - Maximo valor de Thita 6:  %f\n", qmax[6]*180.0/PI );
    printf( "13 - Valor de 'a2'           :  %f\n", a2      );
    printf( "14 - Valor de 'd6'           :  %f\n", d6      );
    printf( "15 - Numero de muestras      :  %d\n", intentos);


/*
 * Ingreso de datos que se desean modificar.
 */

    clr_locate( 24, 1);
    printf( "Elija una opcion ( 0 = CONT , 16 = EXIT ) : " );

    do {
        do {
            clr_locate( 24, 45 );
            scanf( "%d", &item );
        } while ( item ) 16 || item ( 0 );

        if ( item == 16 )
            exit( 2 );

        clr_locate( 7 + item, 31 );
        clr_locate( 7 + item, 32 );
        if ( item )
            if ( item == 15 )  {
                scanf( "%d", &intentos );
                clr_locate( 7 + item, 31 );
                printf( " %d", intentos );
            } else
                if ( item == 14 ) {
                    scanf( "%f", &auxiliar );
                    d6 = auxiliar;
                    clr_locate( 7 + item, 31 );
                    printf( " %f", d6 );
                } else
                    if ( item == 13 )  {
                        scanf( "%f", &auxiliar ù;
                        a2 = auxiliar;
                        clr_locate( 7 + item, 31 );
                        printf( " %f", a2 );
                    } else
                        if ( item & 1 )  {         /* Es una opcion impar */
```

```
                        scanf( "%f", &auxiliar );
                        if ( item < 3 || item > 6 )
                            qmin[ (item + 1) / 2 ] = auxiliar * PI / 180.0;
                        clr_locate( 7 + item, 31 );
                        if ( auxiliar > 0 )
                            printf( " %f", auxiliar );
                        else
                            printf( "%f", auxiliar );
                    } else {                    /* Es una opcion par   */
                        scanf( "%f", &auxiliar );
                        if ( item < 3 || item > 6 )
                            qmax[ item / 2 ] = auxiliar * PI / 180.0;
                        clr_locate( 7 + item, 31 );
                        if ( auxiliar > 0 )
                            printf( " %f", axiliar );
                        else
                            printf( "%f", auxiliar );
                    }

    } while ( item );                           /* Termina cuando es cero */

    for ( i = 1; i <= 6; i++ )          /* casos_totales = (intentos+1)^6 */
        casos_totales *= (intentos + 1);

    clr_locate( 24, 1);
    printf( "Calculando . . .          Faltan : %.0f",
                                        casos_totales - casos_analizados );


/*
 *      Barrido del Volumen de Trabajo para la validacion
 *      del modelo geometrico inverso.
 *
 */

    for ( i = 0; i <= intentos; i++ ) {
       old->teta1 = qmin[1] + i * (qmax[1] - qmin[1]) / intentos;
       for ( j = 0; j <= intentos; j++ ) {
          old->d2 = qmin[2] + j * (qmax[2] - qmin[2]) / intentos;
          for ( k = 0; k <= intentos; k++ ) {
             old->d3 = qmin[3] + k * (qmax[3] - qmin[3]) / intentos;
             for ( l = 0; l <= intentos; l++ ) {
                old->teta4 = qmin[4] + l * (qmax[4] - qmin[4]) / intentos;
                for ( m = 0; m <= intentos; m++ ) {
                   old->teta5 = qmin[5] + m * (qmax[5] - qmin[5]) / intentos;
                   for ( n = 0; n <= intentos; n++ ) {
                      old->teta6 = qmin[6] + n * (qmax[6] - qmin[6]) / intentos;

                      casos_analizados++;
                      mod_directo( old, matriz );
                      if ( mod_inverso( old, matriz, new ) )
```

```
                    {
                        errores++;
                        imprime_errores( old, new, errores, casos_analizados,
                                        casos_totales, matriz );
                    }
                    if ( kbhit() )  {
                        getch();
                        imprime_errores( old, new, errores, casos_analizados,
                                        casos_totales, matriz );
                        clr_locate( 24, 1 );
                        printf( "Presione una tecla para continuar." );
                        getch();
                        clr_locate( 24, 1);
                        printf( "Calculando . . .          Faltan : %.0f",
                                    casos_totales - casos_analizados );
                    }
                }
            }
        }
    }
}

/*
 * Final del programa.
 */

    imprime_errores( old, new, errores, casos_nalizados,
                    casos_totales, matriz );
    clr_locate( 24, 1 );
    printf( "Presione alguna tecla para finalizar." );
    c = getch();
    CLS;

    if ( errores )
        exit( 1 );
    else
        exit(0);
}

/* ------------------------------------------------------------- */

void mod_directo( coordenadas_generalizadas, matriz )

/*
 * Calcula el modelo directo del Robot TH-8
 *
 */

struct generalizadas * coordenadas_generalizadas;
```

```
struct operacionales * matriz;

{
    double si, ci, d2, d3, s4, c4, s5, c5, s6, c6;
    double d[24];                                   /* Ecuaciones Auxiliares */
    double tin[3][4];


    si = sin( coordenadas_generalizadas->tetai );
    ci = cos( coordenadas_generalizadas->tetai );
    d2 = coordenadas_generalizadas->d2;
    d3 = coordenadas_generalizadas->d3;
    s4 = sin( coordenadas_generalizadas->teta4 );
    c4 = cos( coordenadas_generalizadas->teta4 );
    s5 = sin( coordenadas_generalizadas->teta5 );
    c5 = cos( coordenadas_generalizadas->teta5 );
    s6 = sin( coordenadas_generalizadas->teta6 );
    c6 = cos( coordenadas_generalizadas->teta6 );


/*
 * Modelo Geometrico Directo obtenido a traves de GMSIR.
 */

    d[1] = ci*a2;
    d[2] = si*a2;
    d[3] = si*d3+d[1];
    d[4] = -ci*d3+d[2];
    d[5] = ci*c4;
    d[6] = ci*s4;
    d[7] = si*c4;
    d[8] = si*s4;
    d[9] = d[5]*c5+si*s5;
    d[10] = d[5]*s5-si*c5;
    d[11] = d[7]*c5-ci*s5;
    d[12] = d[7]*s5+ci*c5;
    d[13] = s4*c5;
    d[14] = s4*s5;
    d[15] = d[9]*c6+d[6]*s6;
    d[16] = d[9]*s6-d[6]*c6;
    d[17] = d[10]*d6+d[3];
    d[18] = d[11]*c6+d[8]*s6;
    d[19] = d[11]*s6-d[8]*c6;
    d[20] = d[12]*d6+d[4];
    d[21] = d[13]*c6-c4*s6;
    d[22] = d[13]*s6+c4*c6;
    d[23] = d[14]*d6+d2;

    tin[0][0] = d[15];
    tin[0][1] = d[10];
    tin[0][2] = d[16];
```

```
    tin[0][3] = d[17];
    tin[1][0] = d[18];
    tin[1][1] = d[12];
    tin[1][2] = d[19];
    tin[1][3] = d[20];
    tin[2][0] = d[21];
    tin[2][1] = d[14];
    tin[2][2] = d[22];
    tin[2][3] = d[23];


/*
 * Carga de los datos obtenidos en la estructura.
 */

    matriz->nx = tin[0][0];
    matriz->ny = tin[1][0];
    matriz->nz = tin[2][0];
    matriz->ox = tin[0][1];
    matriz->oy = tin[1][1];
    matriz->oz = tin[2][1];
    matriz->ax = tin[0][2];
    matriz->ay = tin[1][2];
    matriz->az = tin[2][2];
    matriz->px = tin[0][3];
    matriz->py = tin[1][3];
    matriz->pz = tin[2][3];

}


/* ------------------------------------------------------------------ */


int mod_inverso( old, matriz, new )

/*
 * Calcula el modelo inverso del Robot TH-8.
 *
 * Devuelve (0) si se pudo calcular el modelo inverso.
 * Devuelve (1) si no se pudo calcular el modelo inverso.
 *
 */

struct generalizadas * old;    /* Posicion anterior de
                                        las articulaciones  */
struct operacionales * matriz; /* Coordenadas del espacio
                                        a ser alcanzadas    */
struct generalizadas * new;    /* Posicion de las articulaciones    */

{
```

```
double ri, fii;      /* Variables auxiliares      */
double s5, c5;       /* Seno y Coseno de 'teta5' */
double si, ci;       /* Seno y Coseno de 'tetai' */


                /* Calculo de 'd2' */

new->d2 = matriz->pz - matriz->oz * d6;


                /* Calculo de 'd3' */

ri = sqrt( square(matriz->px - matriz->ox * d6) +
           square(matriz->py - matriz->oy * d6) );

fii = atan2( matriz->px - matriz->ox * d6, matriz->py - matriz->oy * d6 );

new->d3 = -sqrt( ri * ri - a2 * a2 );


                /* Calculo de 'tetai' */

new->tetai = atan2( a2, -(new->d3) ) - fii;
si = sin( new->tetai );           /* Seno de 'tetai'   */
ci = cos( new->tetai );           /* Coseno de 'tetai' */


         /* Calculo del coseno de 'teta4' 'teta5' 'teta6'  */

c5 = ci * matriz->oy - si * matriz->ox;

/* Solucion cuando coseno(tetai) = +/- 1  */

if ( fabs(c5) ) 0.99999 )
{
    if ( c5 ) 0.0 )
        new->teta5 = 0;           /* Calcula 'teta5'      */
    else
        new->teta5 = PI;

    new->teta4 = old->teta4;      /* Calcula 'teta4'      */

                                  /* Calcula ' teta6'     */
    new->teta6 = new->teta4 - atan2( matriz->nz, matriz->az );
    if ( new->teta6 ) PI )
        new->teta6 -= 2*PI;       /* Corrige si teta6 ) PI */
    if ( new->teta6 ( -PI )
        new->teta6 += 2*PI;       /* Corrige si teta6 ( -PI */

    return( compara_resultados( old, new ) );
}
```

```
    /* Solucion cuando coseno(teta5) != +/- 1  */

    s5 = sqrt( 1 - c5 * c5 );

    new->teta5 = atan2( s5, c5 );       /* Calcula 'teta5' para s5 ) 0 */
    new->teta4 = atan2( matriz->oz , ci * matriz->ox + si * matriz->oy );
                                        /* Calcula 'teta4' */
    new->teta6 = atan2( si * matriz->ax - ci * matriz->ay,
                        si * matriz->nx - ci * matriz->ny );
                                        /* Calcula 'teta6' */
    if ( compara_resultados( old, new ) == 0 )
        return( 0 );

    new->teta5 = atan2( -s5, c5 );      /* Calcula 'teta5' para s5 ( 0 */
    new->teta4 = atan2( - matriz->oz ,
                        - ci * matriz->ox - si * matriz->oy );
                                        /* Calcula 'teta4' */
    new->teta6 = atan2( - si * matriz->ax + ci * matriz->ay,
                        - si * matriz->nx + ci * matriz->ny );
                                        /* Calcula 'teta6' */
    return( compara_resultados( old, new ) );

}


/* ------------------------------------------------------------------ */


int compara_resultados( old, new )

#define         TOLERANCIA_1    0.0001
#define         TOLERANCIA_2    0.0001
#define         TOLERANCIA_3    0.0001
#define         TOLERANCIA_4    0.0001
#define         TOLERANCIA_5    0.0001
#define         TOLERANCIA_6    0.0001

/*
 * Compara las coordenadas verdaderas con la obtenida con la inversion.
 *
 * Devuelve (0) si son iguales.
 * Devuelve (1) si algun parametro es distinto.
 *
 */

struct generalizadas * old;     /* Posicion anterior de
                                                las articulaciones  */
struct generalizadas * new;     /* Posicion de las articulaciones     */

{
```

```
    if ( fabs(old->teta1 - new->teta1) < TOLERANCIA_1  &&
         fabs(old->d2    - new->d2   ) < TOLERANCIA_2  &&
         fabs(old->d3    - new->d3   ) < TOLERANCIA_3  &&
         fabs(old->teta4 - new->teta4) < TOLERANCIA_4  &&
         fabs(old->teta5 - new->teta5) < TOLERANCIA_5  &&
         fabs(old->teta6 - new->teta6) < TOLERANCIA_6 )

         return( 0 );

    else
        return( 1 );

}


/* ------------------------------------------------------------------ */


double square( numero )

/*
 * Eleva un numero al cuadrado.
 */

double numero;

{
    return( numero * numero );
}


/* ------------------------------------------------------------------ */


void clr_locate( fila, columna )

/*
 * Posiciona el cursor en la pantalla y borra hasta el final de la linea.
 *   1 <= Fila <= 25.
 *   1 <= Columna <= 80.
 */

int fila;
int columna;

{
    if ( fila > 0 && fila < 25 && columna > 0 && columna <80 )
        printf( "[%d;%dfEK", fila, columna );
}
```

```
/*-------------------------------------------------------------*/


void imprime_errores( old, new, errores,
                      casos_analizados, casos_totales, matriz )

/*
 * Imprime las dos estructuras y la cantidad de errores ocurridos.
 */

struct generalizadas * old;    /* Posicion anterior de
                                                las articulaciones    */
struct generalizadas * new;    /* Posicion de las articulaciones       */
int    errores;                /* Cantidad de errores                  */
float  casos_analizados;       /* Numero de casos analizados al momento */
float  casos_totales;          /* Numero de casos totales a analizar   */
struct operacionales * matriz; /* Coordenadas alcanzadas del espacio   */

{
    int i;

    CLS;
    clr_locate( 1, 5 );
    puts( "VALIDACION DEL MODELO GEOMETRICO INVERSO DEL ROBOT 'TH-8'" );
    clr_locate( 2, 5 );
    puts( "========== === ====== ========== ======= === =====  ====" );

    clr_locate( 4, 1 );
    printf( " JUNTA          VALOR VERDADERO        VALOR CALCULADO\n" );
    printf( " ------------------------------------------------------\n" );
    printf( "teta1.............. %1.6f ", old->teta1 );
    clr_locate( 6, 31 );
    printf( "............. %1.6f\n", new->teta1 );
    clr_locate( 7, 1 );
    printf( "d2................. %1.6f ", old->d2 );
    clr_locate( 7, 31 );
    printf( "............. %1.6f\n", new->d2 );
    clr_locate( 8, 1 );
    printf( "d3................. %1.6f ", old->d3 );
    clr_locate( 8, 31 );
    printf( "............%1.6f\n", new->d3 );
    clr_locate( 9, 1 );
    printf( "teta4.............. %1.6f ", old->teta4 );
    clr_locate( 9, 31 );
    printf( "............. %1.6f\n", new->teta4 );
    clr_locate( 10, 1 );
    printf( "teta5.............. %1.6f ", old->teta5 );
    clr_locate( 10, 31 );
    printf( "............. %1.6f\n", new->teta5 );
    clr_locate( 11, 1 );
    printf( "teta6.............. %1.6f ", old->teta6 );
```

```
    clr_locate( 11, 31 );
    printf( "..... %1.6f\n", new->teta6 );


    clr_locate( 14, 1 );
    printf( "nx= %f", matriz->nx );
    clr_locate( 14, 17 );
    printf( "ox= %f", matriz->ox );
    clr_locate( 14, 33 );
    printf( "ax= %f", matriz->ax );
    clr_locate( 14, 49 );
    printf( "px= %f", matriz->px );
    clr_locate( 15, 1 );
    printf("Ç"ny= %f", matriz->ny );
    clr_locate( 15, 17Ç);
    printf( "oy= %f", matriz->oy );
    clr_locate( 15, 33 );
    printf( "ay= %f", matriz->ay );
    clr_locate( 15, 49 );
    printf( "py= %f", matriz->py );
    clr_locate( 16, 1 );
    printf( "nz= %f", matriz->nz );
    clr_locate( 16, 17 );
    printf( "oz= %f", matriz->oz );
    clr_locate( 16, 33 );
    printf( "az= %f", matriz->az );
    clr_locate( 16, 49 );
    printf( "pz= %f", matriz->pz );


    clr_locate( 19, 1 );
    printf( "Numero total de casos a analizar    : %.0f\n", casos_totales );
    clr_locate( 20, 1 );
    printf( "Numero de casos analizados          : %.0f\n", casos_analizados );
    clr_locate( 21, 1 );
    printf( "Numero de casos que faltan analizar : %.0f\n", casos_totales - casos_analizados );
    clr_locate( 22, 1 );
    printf( "Cantidad de errores ocurridos       : %d\n", errores );
}

/* ------------------------------------------------------------------ */
```

```
/*****************************************************************
 *****************************************************************
 **********                                             **********
 **********  GENERACION DE TRAYECTORIAS EN EL DOMINIO   **********
 **********  DE LAS COORDENADAS GENERALIZADAS A PARTIR  **********
 **********  DEL DOMINIO DE LAS OPERACIONALES Y CALCU-  **********
 **********  LO DE GASTOS ENERGETICOS                   **********
 *****************************************************************
 *****************************************************************
 **********                                             **********
 **********  AUTORES:                                   **********
 **********                                             **********
 **********         Daniel Juan Franco                  **********
 **********         Paulo Renato Battaglia              **********
 **********         Domingos S. L. Simonetti            **********
 **********                                             **********
 **Î*******         III EBAI - Curitiba,  02-02-1988    **********
 **********                                             **********
 *****************************************************************
 *****************************************************************/


#include      <stdio.h>
#include      <math.h>
#include      <IO.h>

#define       LINT_ARGS       1
#define       BEEP            printf( "\7" )
#define       CLS             printf( "[2J" )
#define       PI              3.141592654

#define       Q1MINIMO        -160*PI/180         /* Minimo tita1 */
#define       Q1MAXIMO         160*PI/180         /* Maximo tita1 */
#define       Q2MINIMO        0.2                 /* Minimo d2    */
#define       Q2MAXIMO        0.7                 /* Maximo d2    */
#define       Q3MINIMO        -0.7                /* Minimo d3    */
#define       Q3MAXIMO        -0.2                /* Maximo d3    */
#define       Q4MINIMO        -160*PI/180         /* Minimo tita4 */
#define       Q4MAXIMO         160*PI/180         /* Maximo tita4 */
#define       Q5MINIMO        -160*PI/180         /* Minimo tita5 */
#define       Q5MAXIMO         160*PI/180         /* Maximo tita5 */
#define       Q6MINIMO        -160*PI/180         /* Minimo tita6 */
#define       Q6MAXIMO         160*PI/180         /* Maximo tita6 */
#define       A2              0.051               /* Valor de 'a2'*/
#define       D6              0.291               /* Valor de 'd6'*/
```

```
typedef struct operacionales { double nx, ox, ax, px;
                               double ny, oy, ay, py;
                               double nz, oz, az, pz;
                             };

typedef struct generalizadas { double teta1, teta4, teta5, teta6;
                               double d2, d3;
                             };


double a2 = A2, d6 = D6;        /* Distancias constantes del Robot 'TH-8' */


void   mod_directo( struct eneralizadas *,
                    struct operacionales * );
int    mod_inverso( struct operacionales *,
                    double, double,
                    struct generalizadas *,
                    struct generalizadas * );
double square( double );
void   clr_locate( int, int );
void  carop (void);       /* Carga archivo de coordenadas operacionales */
int   solucion (void);    /* Calcula geom. inverso de trayectoria */
void  nada(void);         /* Opciones no implementadas */
int   imprime(void);      /* Imprime soluciones */



/* ------------------------------------------------------------------ */


main()

/*
 * Generacion del modelo geometrico inverso del Robot 'TH-8'.
 *
 *
 */


{
    int  opcion;

    do{
        CLS;
        clr_locate( 2, 1 );
        printf(" 1 - Carga de coordenadas operacionales \n\n" );
        printf(" 2 - Calcular soluciones de trayectorias \n\n" );
        printf(" 3 - Listar archivo de operacionales \n\n"    );
        printf(" 4 - Listar archivo de generalizadas \n\n"    );
        printf(" 5 - Imprimir soluciones de trayectorias \n\n" );
```

```
        printf(" 6 - Fin \n\n" );

        clr_locate ( 15, 1 );
        printf ( "Seleccione una opcion: " );
        do {
            scanf( "%d", &opcion );
        } while ( opcion ( 1 || opcion ) 6 );

        switch ( opcion )  {
            case  1:
                carop();
                break;
            case  2:
                solucion();
                break;
            case  3:
                nada();
                break;
            case  4:
                nada();
                break;
            case  5:
                imprime();
                break;
            case  6:
                exit( 0 );
        };
    } while ( opcion != 6 );
}

/*
 * Final del programa.
 */

/* ------------------------------------------------------------------ */

  void nada()
  /* opciones del menu no implementadas */
  {
   CLS;
   clr_locate(10,5);
   printf ("Esta opcion aun no ha sido implementada.\n");
   clr_locate(12,5);
   printf ("Si la necesita, hagala. Es muy facil. \n");
    clr_locate(14,5);
   printf ("Presione una tecla para continuar. chau \n");
   getch();
  }

/* ------------------------------------------------------------------ */
```

```
void carop()

/*
 * Carga y modificacion de juegos de datos
 *   del archivo  de coordenadas operacionales.
 */

{
    struct operacionales *p;      /* Coordenadas operacionales a cargar   */
    int     fd;                   /* Descriptor de archivo                */
    int     i,j,k,l,m,n;          /* Variables auxiliares                 */
    float   a1,a2,a3,a4;          /* Variables auxiliares                 */
    char    nombre[16];           /* Nombre del archivo                   */
    char    modif;                /* Variable para confirmar modificacion */


    p = malloc (sizeof(struct operacionales));
    CLS;
    clr_locate( 2, 1 );
    printf ( "Ingrese nombre del archivo : " );
    scanf  ( "%s",&nombre[0] );            /* Ingresar nombre del archivo */
    strcat ( &nombre[0], ".cop");          /* Agregar extension .cop      */

    fd = open ( &nombre[0], 2 );/* Intentar abrirlo y si no existe crearlo */

    if (fd == -1) fd = creat(&nombre[0],0777);

    do{
              /* Largo del archivo en registros */
        k = (int) filelength(fd) / sizeof(struct operacionales);

        CLS;
        clr_locate( 2, 1 );
        printf ( "Ingrese numero de coordenada  ( 0 = fin ) : " );
        scanf("%d",&i);           /* Numero de coordenada a cargar datos */

        if ( i ) 0 ) {
            if ( i <= k ) {              /* Si el registro existe leerlo */

                lseek(fd,(long)(i-1)*sizeof (struct operacionales),0);
                                         /* Posicionar para  leer       */

                j = read( fd, p, sizeof(struct operacionales) );
                clr_locate( 4, 1 );
                printf( "Los valores actuales son: \n" );
                printf( "nx %f ox %f ax %f px %f \n",p->nx,p->ox,p->ax,p->px );
                printf( "ny %f oy %f ay %f py %f \n",p->ny,p->oy,p->ay,p->py );
                printf( "nz %f oz %f az %f pz %f \n",p->nz,p->oz,p->az,p->pz );
            );
            clr_locate( 9, 1 );
```

```
            printf( "Desea ingresar nuevos valores (S/N) : " );
            modif = getch();
            if ( modif == 's' || modif == 'S' ) {

                putch( modif );          /* Imprime la opcion elegida     */

                clr_locate( ii, i );

                printf( "Ingrese valores de  nx  ox  ax  px \n" );
                scanf ( "%f %f %f %f", &a1,&a2,&a3,&a4 );
                p->nx = a1; p->ox = a2; p->ax = a3 ; p->px = a4;

                printf( "Ingrese valores de  ny  oy  ay  py \n" );
                scanf ( "%f %f %f %f", &a1,&a2,&a3,&a4 );
                p->ny = a1; p->oy = a2; p->ay = a3; p->py = a4;

                printf( "Ingrese valores de  nz  oz  az  pz \n" );
                scanf ( "%f %f %f %f", &a1,&a2,&a3,&a4 );
                p->nz = a1; p->oz = a2; p->az = a3; p->pz = a4;

                lseek( fd, 0L, 0 );
                lseek( fd, (long)(i-1)*sizeof(struct operacionales), 0 );
                                /* Posicionar para  escribir          */

                write ( fd, p, sizeof(struct operacionales) );
                                /* Escribir nuevos valores            */
            };
        };

    } while ( i != 0 );

    free (p);
    close ( fd );

}

/* ------------------------------------------------------------------ */

int solucion ()


/*
 * Obtiene para un juego de coordenadas operacionales dadas en un ar
 * chivo, las soluciones inversas cuando las tiene y el gasto ener-
 * getico necesario para alcanzar cada posicion, tomando como refe-
 * rencia la posicion del primer juego de coordenadas.
 * El archivo de entrada tiene la extension .cop , y el archivo de sa-
 * lida tiene el mismo nombre que el de entrada con la extension .sol
 * con el siguiente formato:
 * -Estructura operacionales para almacenar datos de entrada
 * -Estructura generalizadas para almacenar solucion a)
```

```
 * -Estructura generalizadas para almacenar solucion b)
 * -Double para almacenar gasto energetico desde referencia a) a solucion a)
 * -Double para almacenar gasto energetico desde referencia b) a solucion a)
 * -Double para almacenar gasto energetico desde referencia a) a solucion b)
 * -Double para almacenar gasto energetico desde referenci b) a solucion b)
 * Si el punto esta fuera del volumen de trabajo los 4 gastos son = -1.-
 */
{
struct registro { struct operacionales p;  /* Estructura de la salida para */
                  struct generalizadas qa;   /* cada juego de coordenadas    */
                  struct generalizadas qb;   /* operacionales                */
                  double giaa;
                  double giab;
                  double giba;
                  double gibb;
                  } * coord   ;
int i,j,k,l,m,n;                        /* Variables enteras auxiliares */
float a, b, c, d;                      /* Variables flotantes auxiliares */
double qra[7] , qrb[7];                /* Arreglos para almacenar las coor
                                          denadas generalizadas de referen-
                                          cia, correspondientes al primer
                                          registro del archivo de entrada */
int fdin , fdout;           /* Descriptores de archivos de entrada y salida */
char nombre [20];           /* Arreglo para recibir nombre del archivo */
char name [20];             /* Arreglo para armar nombre archivo salida */
double ti4aa,ti4ab;         /* Almacenar valores de tita4 de la posicion
                               anterior                                  */
float peso [7];             /* peso del movimiento para cada articulacion */

coord = malloc (sizeof (struct registro));

CLS;
printf(" Calculo de modelo geometrico inverso con las diferentes sol- \n");
printf(" ciones y el gasto energetico necesario para llegar a ellas des-\n");
printf(" de un punto de referencia. Este primer punto corresponde al \n");
printf(" juego de coordenadas que se encuentre en el primer registro del \n");
printf(" archivo de entrada \n\n\n");

/* Ingreso de pesos para movimiento de las articulaciones */
 for (n = 1; n (= 6; n++)
     {
     printf ("Ingrese el peso del movimiento de la articulacion %d \n",n);
     scanf ("%f", &peso [n]);
     };


printf("Ingrese nombre del archivo de entrada: ");
          /* Armar nombres de archivos de entrada y salida */
scanf ("%s",&nombre[0]);
strcpy (name, nombre);
strcat (&nombre[0],".cop");
```

```
strcat (&name[0],".sol");

          /* Intentar abrir el archivo de entrada y si da error
             salir con un mensaje */
fdin = open ( nombre, 2 );
if (fdin == -1)
   {
    printf (" El archivo %s no existe -  Presione una tecla para seguir \n",nombre);
    getch();
    free (coord);
    return (1);
   };
k = (int) filelength (fdin) / sizeof (struct operacionales);
printf(" En el archivo %s tiene %d registros para procesar\n",nombre,k);



          /* En caso de que el archivo exista, crear el de salida */
fdout = creat ( name, 0777);



/* Leer el primer registro del archivo de entrada  y obtener el primer
*  juego de inversas.
*  En caso de que estas no sean validas salir
*/
read (fdin , coord, sizeof(struct operacionales));
ti4aa = Q4MINIMO + 0.1;    /* Para calcular el inverso de la primer          */
ti4ab = Q4MINIMO + 0.1;    /* coordenada se adopta tita4 anterior = Q4MINIMO */


/* printf (" Voy a pasar los siguientes punteros p %d qa %d qb %d \n",
          coord, &(coord->qa.tetai),&(coord->qb.tetai));  */


m = mod_inverso (coord, ti4aa, ti4ab, &(coord->qa.tetai),&(coord->qb.tetai));
/*printf("valor devuelto por mod_inverso %d ",m); */
if ( m )
   {
    printf("La coordenada de referencia es irresoluble - Presione tecla \n");
    getch();
    close (fdin); close (fdout);
    return (1);
   };
/* Guardar los valores de las coordenadas generalizadas de referencia */
qra [1] = coord->qa.tetai ;      qrb [1] = coord->qb.tetai ;
qra [2] = coord->qa.d2 ;         qrb [2] = coord->qb.d2;
qra [3] = coord->qa.d3 ;         qrb [3] = coord->qb.d3;
qra [4] = coord->qa.teta4;       qrb [4] = coord->qb.teta4;
qra [5] = coord->qa.teta5;       qrb [5] = coord->qb.teta5;
qra [6] = coord->qa.teta6;       qrb [6] = coord->qb.teta6;
```

```
/* Asumir gasto energetico para llegar a posicion de referencia = 0 */
coord->giaa = 0.0; coord->giab = 0.0;
coord->giba = 0.0; coord->gibb = 0.0;

/* Grabar los parametros de la coordenada de referencia en el archivo
*  de salida.
*/

write (fdout,coord, sizeof (struct registro));

/* Loop de calculo de solucion para todo el archivo de entrada */

 ti4aa = coord->qa.teta4; /* Fijamos los valores de tita4 anterior igual al */
 ti4ab = coord->qb.teta4; /* tita4 de la posicion de referencia             */

for ( j = 2 ; j <= k ; j++)
 {
   /* Leer desde el archivo de entrada las coordenadas operacionales */
  read (fdin, coord, sizeof (struct operacionales));

   /* Obtener el modelo inverso */
  m = mod_inverso (coord, ti4aa, ti4ab, &(coord->qa.tetai), &(coord->qb.tetai));

   /* Si el modelo inverso no es valido hacer los gastos energeticos = -1
    * Caso contrario calcularlos.
    */
  if (m)
    { coord->giaa = -1; coord->giab = -1;
      coord->giba = -1; coord->gibb = -1;
    }
  else
    {

      /* Para llegar a alternativa a) desde referencia a) */
      coord->giaa =  fabs(coord->qa.tetai - qra[1]) * peso [1] +
                     fabs(coord->qa.d2    - qra[2]) * peso [2] +
                     fabs(coord->qa.d3    - qra[3]) * peso [3] +
                     fabs(coord->qa.teta4 - qra[4]) * peso [4] +
                     fabs(coord->qa.teta5 - qra[5]) * peso [5] +
                     fabs(coord->qa.teta6 - qra[6]) * peso [6];

      /* Para llegar a alternativa a) desde referencia b) */
      coord->giab =  fabs(coord->qa.tetai - qrb[1]) * peso [1] +
                     fabs(coord->qa.d2    - qrb[2]) * peso [2] +
                     fabs(coord->qa.d3    - qrb[3]) * peso [3] +
                     fabs(coord->qa.teta4 - qrb[4]) * peso [4] +
                     fabs(coord->qa.teta5 - qrb[5]) * peso [5] +
                     fabs(coord->qa.teta6 - qrb[6]) * peso [6];

      /* Para llegar a alternativa b) desde referencia a) */
```

```
        coord->giba =  fabs(coord->qb.tetai - qra[1]) * peso [1] +
                       fabs(coord->qb.d2    - qra[2]) * peso [2] +
                       fabs(coord->qb.d3    - qra[3]) * peso [3] +
                       fabs(coord->qb.teta4 - qra[4]) * peso [4] +
                       fabs(coord->qb.teta5 - qra[5]) * peso [5] +
                       fabs(coord->qb.teta6 - qra[6]) * peso [6];


        /* Para llegar a alternativa b) desde referencia b) */
        coord->gibb =  fabs(coord->qb.tetai - qrb[1]) * peso [1] +
                       fabs(coord->qb.d2    - qrb[2]) * peso [2] +
                       fabs(coord->qb.d3    - qrb[3]) * peso [3] +
                       fabs(coord->qb.teta4 - qrb[4]) * peso [4] +
                       fabs(coord->qb.teta5 - qrb[5]) * peso [5] +
                       fabs(coord->qb.teta6 - qrb[6]) * peso [6];


      );
    /* Grabar los valores calculados en el registro de salida */
    write (fdout, coord, sizeof (struct registro));
   );  /* Fin del lazo de calculo */

   close (fdin); close (fdout);
   free (coord);
   return (0);
}
/* ------------------------------------------------------------------ */

int imprime ()


/*
 * Imprime los resultados obtenidos por el programa solucion().
 * Estos datos son especificados entrando el nombre del archivo-
 * sin la extension .sol.
 * Los datos son leidos, formateados y escritos en el dispositivo
 * de salida que se especifique.
 * El archivo de entrada (.sol) tiene el siguiente formato:
 * con el siguiente formato:
 * -Estructura operacionales: Coordenadas operacionales dato.
 * -Estructura generalizadas: Coordenadas generalizadas solucion a)
 * -Estructura generalizadas: Coordenadas generalizadas solucion b)
 * -Double para almacenar gasto energetico desde referencia a) a solucion a)
 * -Double para almacenar gasto energetico desde referencia b) a solucion a)
 * -Double para almacenar gasto energetico desde referencia a) a solucion b)
 * -Double para almacenar gasto energetico desde referencia b) a solucion b)
 * Si el punto esta fuera del volumen de trabajo los 4 gastos son = -1.-
 */
{
struct registro { struct operacionales p;  /* Estructura de la entrada para */
                  struct generalizadas qa;   /* cada juego de coordenadas     */
```

```
                struct generalizadas qb;   /* operacionales              */
                double giaa;
                double giab;
                double giba;
                double gibb;
                ) * coord   ;
FILE *fdout,*fopen(),*fcreat();   /*Descriptor archivo de salida*/
int i,j,k,l,m,n;                        /* Variables enteras auxiliares */
float a, b, c, d;                      /* Variables flotantes auxiliares */
int fdin ;                  /* Descriptor de archivo de entrada  */
char nombre [20];           /* Arreglo para recibir nombre del archivo */
char name [20];             /* Arreglo para nombre dispositivo salida  */

coord = malloc (sizeof(struct registro));
CLS;
printf(" Impresion de resultados obtenidos en resolucion de trayectorias\n");
printf(" Debera ingresarse el nombre del archivo donde se encuentran los\n");
printf(" datos de las coordenadas. Previamente se debe haber obtenido\n");
printf(" la solucion usando la opcion corriente del menu previo         \n");
printf(" \n\n");

printf("Ingrese nombre del archivo de entrada: ");
          /* Armar nombres de archivos de entrada y salida */
scanf ("%s",&nombre[0]);
strcat (&nombre[0],".sol");
printf("\n Ingrese el nombre del dispositivo de salida: ");
scanf ("%s",&name[0]);
printf("Si continua escriba (s) ");
if ( (i = getch()) != 's' && i != 'S')
    return (1);
          /* Intentar abrir el archivo de entrada y si da error
             salir con un mensaje */
fdin = open ( nombre, 2 );
if (fdin == -1)
   {
    printf (" El archivo %s no existe -  Presione una tecla para seguir ",nombre);
    getch();
    return (1);
   };
k = (int) filelength (fdin) / sizeof (struct registro);
printf(" En el archivo %s tiene %d registros para imprimir\n",nombre,k);
```

```
        /* En caso de que el archivo exista, crear el de salida */
fdout = fopen  ( name, "w");
   if (fdout != NULL )
   printf ("El archivo de salida abierto es %s \n",name);
   else
   {
   printf ("El archivo de salida no aberto es %s \n",name);
   getch();
   return (1);
   };
fprintf (fdout,"Resultados de resolucion de trayectorias %s \n",nombre);

        /* Impresion del encabezamiento */

fprintf (fdout, "************************************************** \n");

        /* Impresion de todos los valores calculados */

for (i = 1; i <= k ; i++)    /* Lazo de todos los regitros del archivo */
  {
   read (fdin, coord, sizeof (struct registro));
   fprintf(fdout,"Coordenada nro.: %d \n",i);
   fprintf(fdout,"nx %f    ox %f   ax %f     px %f  \n",
               coord->p.nx, coord->p.ox, coord->p.ax, coord->p.px);

   fprintf(fdout,"ny %f    oy %f   ay %f     py %f  \n",
               coord->p.ny, coord->p.oy, coord->p.ay, coord->p.py);

   fprintf(fdout,"nz %f    oz %f   az %f     pz %f  \n",
               coord->p.nz, coord->p.oz, coord->p.az, coord->p.pz);

   if (coord->giaa )= 0.0)
   {
   fprintf(fdout,"Coordenadas generalizadas alternativa a: \n");
   fprintf(fdout,"tita1 %f  d2 %f  d3 %f \n tita4 %f  tita5 %f  tita6 %f \n",
           coord->qa.teta1 * 180 / PI, coord->qa.d2,
           coord->qa.d3, coord->qa.teta4 * 180 / PI,
           coord->qa.teta5 * 180 / PI, coord->qa.teta6 * 180 / PI);


   fprintf(fdout,"Coordenadas generalizadas alternativa b: \n");
   fprintf(fdout,"tita1 %f  d2 %f  d3 %f \n tita4 %f  tita5 %f  tita6 %f \n",
           coord->qb.teta1 * 180 / PI, coord->qb.d2,
           coord->qb.d3, coord->qb.teta4 * 180 / PI,
           coord->qb.teta5 * 180 / PI, coord->qb.teta6 * 180 / PI);
   };
   fprintf(fdout,"Gasto desde ref. a) a alternativa a) %f \n",coord->giaa);
   fprintf(fdout,"Gasto desde ref. b) a alternativa a) %f \n",coord->giab);
   fprintf(fdout,"Gasto desde ref. a) a alternativa b) %f \n",coord->giba);
   fprintf(fdout,"Gasto desde ref. b) a alternativa b) %f \n",coord->gibb);
```

```
   fprintf(fdout,"\n\n");

   ); /* fin del loop de impresion */
free (coord);
close (fdin); fclose (fdout);
}   /* Final del programa imprime */
/* ------------------------------------------------------------- */


void mod_directo( coordenadas_generalizadas, matriz )

/*
 * Calcula el modelo directo del Robot TH-8
 *
 */

struct generalizadas * coordenadas_generalizadas;
struct operacionales * matriz;

{
    double si, ci, d2, d3, s4, c4, s5, c5, s6, c6;
    double d[24];                                  /* Ecuaciones Auxiliares */
    double tin[3][4];


    si = sin( coordenadas_generalizadas->tetai );
    ci = cos( coordenadas_generalizadas->tetai );
    d2 = coordenadas_generalizadas->d2;
    d3 = coordenadas_generalizadas->d3;
    s4 = sin( coordenadas_generalizadas->teta4 );
    c4 = cos( coordenadas_generalizadas->teta4 );
    s5 = sin( coordenadas_generalizadas->teta5 );
    c5 = cos( coordenadas_generalizadas->teta5 );
    s6 = sin( coordenadas_generalizadas->teta6 );
    c6 = cos( coordenadas_generalizadas->teta6 );


/*
 * Modelo Geometrico Directo obtenido a traves de GMSIR.
 */

    d[1] = ci*a2;
    d[2] = si*a2;
    d[3] = si*d3+d[1];
    d[4] = -ci*d3+d[2];
    d[5] = ci*c4;
    d[6] = ci*s4;
    d[7] = si*c4;
    d[8] = si*s4;
    d[9] = d[5]*c5+si*s5;
    d[10] = d[5]*s5-si*c5;
```

```
    d[11] = d[7]*c5-c1*s5;
    d[12] = d[7]*s5+c1*c5;
    d[13] = s4*c5;
    d[14] = s4*s5;
    d[15] = d[9]*c6+d[6]*s6;
    d[16] = d[9]*s6-d[6]*c6;
    d[17] = d[10]*d6+d[3];
    d[18] = d[11]*c6+d[8]*s6;
    d[19] = d[11]*s6-d[8]*c6;
    d[20] = d[12]*d6+d[4];
    d[21] = d[13]*c6-c4*s6;
    d[22] = d[13]*s6+c4*c6;
    d[23] = d[14]*d6+d2;

    tin[0][0] = d[15];
    tin[0][1] = d[10];
    tin[0][2] = d[16];
    tin[0][3] = d[17];
    tin[1][0] = d[18];
    tin[1][1] = d[12];
    tin[1][2] = d[19];
    tin[1][3] = d[20];
    tin[2][0] = d[21];
    tin[2][1] = d[14];
    tin[2][2] = d[22];
    tin[2][3] = d[23];


/*
 * Carga de los datos obtenidos en la estructura.
 */

    matriz->nx = tin[0][0];
    matriz->ny = tin[1][0];
    matriz->nz = tin[2][0];
    matriz->ox = tin[0][1];
    matriz->oy = tin[1][1];
    matriz->oz = tin[2][1];
    matriz->ax = tin[0][2];
    matriz->ay = tin[1][2];
    matriz->az = tin[2][2];
    matriz->px = tin[0][3];
    matriz->py = tin[1][3];
    matriz->pz = tin[2][3];

}


/* ------------------------------------------------------------------ */
```

```
int mod_inverso( p, ti4aa, ti4ab, qa, qb)

/*
 * Calcula el modelo inverso del Robot TH-8.
 *
 * Devuelve (0) si se pudo calcular el modelo inverso.
 * Devuelve (1) si no se pudo calcular el modelo inverso.
 * Recibe como parametros un puntero a las operacionales,
 *        los valores de tita4 anterior para solucion a y b, y
 *        punteros a generalizadas para entregar soluciones a y b;
 */

struct operacionales *p ;      /* Coordenadas operacionales del robot */
double ti4aa, ti4ab;           /* Coordenadas thita4 de la posicion anterior
                                  correspondientes a las soluciones a y b*/
struct generalizadas *qa ;     /* Puntero para entregar solucion a)      */
struct generalizadas *qb ;     /* Puntero para entregar solucion b)      */
{
    double r1, fi1;      /* Variables auxiliares      */
    double s5, c5;       /* Seno y Coseno de 'teta5' */
    double si, ci;       /* Seno y Coseno de 'tetai' */
    int verqa, verqb ;   /* =0 si qa y qb no verifican las condiciones de
                          * topes de tita4, tita5 y tita6.
                          */


 /* printf(" Recibi los siguientes punteros p  %d   qa %d   qb %d  \n",
             p,qa,qb); */



                    /* Calculo de 'd2' */

    qa->d2 = p->pz - p->oz * d6;
    qb->d2 = p->pz - p->oz * d6;
    if (qa->d2 < Q2MINIMO || qa->d2 > Q2MAXIMO) return (1);

                    /* Calculo de 'd3' */

    r1 =  square(p->px - p->ox * d6) +
          square(p->py - p->oy * d6) ;
    if (r1 < 0.0 ) return (1);
    r1 = sqrt ( r1 );

    fi1 = atan2( p->px - p->ox * d6, p->py - p->oy * d6 );

    qa->d3 = ( r1 * r1 - a2 * a2 );
    if (qa->d3 ) (Q3MINIMO * Q3MINIMO) || qa->d3 < (Q3MAXIMO * Q3MAXIMO) )
       return (1);
    qa->d3 = -sqrt (qa->d3);
    qb->d3 = qa->d3;
```

```
                ii* Calculo de 'tetai' */

qa->tetai = atan2( a2, -(qa->d3) ) - fii;
qb->tetai = qa->tetai;
if (qa->tetai < Q1MINIMO || qa->tetai > Q1MAXIMO)
   return (1);
si = sin( qa->tetai );            /* Seno de 'tetai'   */
ci = cos( qa->tetai );            /* Coseno de 'tetai' */


         /* Calculo del coseno de 'teta4' 'teta5' 'teta6'  */

c5 = ci * p->oy - si * p->ox;

/* Solucion cuando coseno(tetai) = +/- 1  */

if ( fabs(c5) > 0.99999 )
{
    if ( c5 > 0.0 )
        {
        qa->teta5 = 0;            /* Calcula 'teta5'       */
        qb->teta5 = 0;
        }
    else
        {
        qa->teta5 = PI;
        qb->teta5 = PI;
        };
    if (qa->teta5 < Q5MINIMO || qa->teta5 > Q5MAXIMO)
    return (1);
    qa->teta4 = ti4aa;            /* Calcula 'teta4'       */
    qb->teta4 = ti4ab;

                                   /* Calcula ' teta6'      */
    qa->teta6 = ti4aa - atan2( p->nz, p->az );
    qb->teta6 = ti4ab - atan2( p->nz, p->az );
    if ( qa->teta6 > PI )
         qa->teta6 -= 2*PI;        /* Corrige si teta6 > PI */
    if ( qa->teta6 < -PI )
        qa->teta6 += 2*PI;         /* Corrige si teta6 < -PI */
    if ( qb->teta6 > PI )
         qb->teta6 -= 2*PI;
    if ( qb->teta6 < -PI )
         qb->teta6 += 2*PI;

    /* Llamada a rutina que verifica que qa y qb corresponden a p
     * obteniendo el modelo directo
     */
}
else
```

```
    {
    /* Solucion cuando coseno(teta5) != +/- 1  */

    s5 = sqrt( 1 - c5 * c5 );

    qa->teta5 = atan2( s5, c5 );       /* Calcula 'teta5' para s5 > 0 */
    qa->teta4 = atan2( p->oz , ci * p->ox + si * p->oy );
                                       /* Calcula 'teta4' */
    qa->teta6 = atan2( si * p->ax - ci * p->ay,
                       si * p->nx - ci * p->ny );
                                       /* Calcula 'teta6' */



    qb->teta5 = atan2( -s5, c5 );      /* Calcula 'teta5' para s5 < 0 */
    qb->teta4 = atan2( - p->oz ,
                       - ci * p->ox - si * p->oy );
                                       /* Calcula 'teta4' */
    qb->teta6 = atan2( - si * p->ax + ci * p->ay,
                       - si * p->nx + ci * p->ny );
                                       /* Calcula 'teta6' */
    };

  /* ojo retorno puesto para que no verifique tita4 /tita6
   * sacar despues de depurar
   */

/* return (0);  */


    /* Para las dos soluciones obtenidas verificar cual de ellas cumple
     * con los topes maximos y minimos de teta4, teta5 y teta6.
     * Si ninguna verifica la condicion retornar error.
     *ÇSi una sola verifica la condicion entregar dos soluciones guales
     * a esta.
     * Si las dos verifica la condicion entregar las dos soluciones dis-
     * tintas.
     * Antes de devolver el control verificar que las soluciones obtenidas
     * corresponden al punto dato en coordenadas operacionales a traves
     * de la funcion compara_resultados. (Esto tendria que ser siempre
     * cierto, ya que la generacion del modelo inverso fue validada
     * a traves de corridas del programa th8val
     */
    verqa = 0;    /*Si =0 qa no verifica la condicion */
    verqb = 0;    /*Si =0 qb no verifica la condicion */

    /* Verificar condiciones de topes para tita4, 5 y 6 de qa  */

    if (qa->teta4 >= Q4MINIMO && qa->teta4 <= Q4MAXIMO &&
        qa->teta5 >= Q5MINIMO && qa->teta5 <= Q5MAXIMO &è
        qa->teta6 >= Q6MINIMO && qa->teta6 <= Q6MAXIMO  )
```

```
        verqa = 1;



    /* Verificar condiciones de topes para tita4, 5 y 6 de qb  */

    if (qb->teta4 >= Q4MINIMO && qb->teta4 <= Q4MAXIMO &&
        qb->teta5 >= Q5MINIMO && qb->teta5 <= Q5MAXIMO &&
        qb->teta6 >= Q6MINIMO && qb->teta6 <= Q6MAXIMO  )
        verqb = 1;



    /* Si ninguna verifica retornar error*/
    if (verqa == 0 && verqb == 0 )
       {printf ("sali antes de comparar resultados \n");
       return (1);
       };
    /* Si verifica una de las dos igualar la otra a esta */

    if (verqa != verqb)
    {
      if (verqa )        /* la que verifica es qa */
      {
       qb->teta4 = qa->teta4;
       qb->teta5 = qa->teta5;
       qb->teta6 = qa->teta6;

      }

      else               /* la que verifica es qb */
      {
       qa->teta4 = qb->teta4;
       qa->teta5 = qb->teta5;
       qa->teta6 = qb->teta6;

      };
    };

    return (compara_resultados ( qa, qb, p ));

}

/* ------------------------------------------------------------------ */

int compara_resultados( qa, qb, p )
```

```
#define         TOLERANCIA_NX   0.1001
#define         TOLERANCIA_NY   0.1001
#define         TOLERANCIA_NZ   0.1001
#define         TOLERANCIA_OX   0.1001
#define         TOLERANCIA_OY   0.1001
#define         TOLERANCIA_OZ   0.1001
#define         TOLERANCIA_AX   0.1001
#define         TOLERANCIA_AY   0.1001
#define         TOLERANCIA_AZ   0.1001
#define         TOLERANCIA_PX   0.1001
#define         TOLERANCIA_PY   0.1001
#define         TOLERANCIA_PZ   0.1001

/*
 * Compara las dos soluciones obtenidas por la transformacion inversa
 * si la transformada directa de estas corresponde al punto dado.
 * Si cualquiera de ellas difiere retorna el codigo de error.
 * Devuelve (0) si las soluciones son correctas
 * Devuelve (1) si alguna de ellas es incorrecta.
 *
 */

struct generalizadas * qa ;    /* Alternativa a) de solucion encontra-
                                  da por la transformacion  inversa  */
struct generalizadas * qb ;    /* Alternativa b) de solucion encontra-
                                  da por la transformacion inversa   */
struct operacionales * p  ;    /* Posicion a verificar por la transfor-
                                  mada directa de a) y b) */
{
struct operacionales * pin ;   /* Coordenadas operacionales devueltas
                                  por la transformacion directa de las
                                  soluciones a y b para ser comparadas
                                  con p                              */

pin = malloc (sizeof (struct operacionales));

    /* Evaluar la veracidad de qa */


    ..........................................................................

    mod_directo ( qa , pin );


    if ( fabs(p->nx - pin->nx) > TOLERANCIA_NX    ||
         fabs(p->ny - pin->ny) > TOLERANCIA_NY    ||
         fabs(p->nz - pin->nz) > TOLERANCIA_NZ    ||
         fabs(p->ox - pin->ox) > TOLERANCIA_OX    ||
         fabs(p->oy - pin->oy) > TOLERANCIA_OY    ||
         fabs(p->oz - pin->oz) > TOLERANCIA_OZ    ||
         fabs(p->ax - pin->ax) > TOLERANCIA_AX    ||
```

```
        fabs(p->ay - pin->ay) ) TOLERANCIA_AY    ||
        fabs(p->az - pin->az) ) TOLERANCIA_AZ    ||
        fabs(p->px - pin->px) ) TOLERANCIA_PX    ||
        fabs(p->py - pin->py) ) TOLERANCIA_PY    ||
        fabs(p->pz - pin->pz) ) TOLERANCIA_PZ  )
        {

        free (pin);
        return( 1 );
        }
    else
    {
    /* Evaluar la veracidad de qb */

    mod_directo ( qb , pin ) ;


    if ( fabs(p->nx - pin->nx) ) TOLERANCIA_NX    ||
        fabs(p->ny - pin->ny) ) TOLERANCIA_NY    ||
        fabs(p->nz - pin->nz) ) TOLERANCIA_NZ    ||
        fabs(p->ox - pin->ox) ) TOLERANCIA_OX    ||
        fabs(p->oy - pin->oy) ) TOLERANCIA_OY    ||
        fabs(p->oz - pin->oz) ) TOLERANCIA_OZ    ||
        fabs(p->ax - pin->ax) ) TOLERANCIA_AX    ||
        fabs(p->ay - pin->ay) ) TOLERANCIA_AY    ||
        fabs(p->az - pin->az) ) TOLERANCIA_AZ    ||
        fabs(p->px - pin->px) ) TOLERANCIA_PX    ||
        fabs(p->py - pin->py) ) TOLERANCIA_PY    ||
        fabs(p->pz - pin->pz) ) TOLERANCIA_PZ  )
        {


        free (pin);
        return( 1 );
        }
        else
        free (pin);
        return (0);
     ); /*final if-else principal*/
}

/* ------------------------------------------------------------ */


double square( numero )

/*
 * Eleva un numero al cuadrado.
 */
```

```
double numero;

{
    return( numero * numero );
}


/* ------------------------------------------------------------------ */


void clr_locate( fila, columna )

/*
 * Posiciona el cursor en la pantalla y borra hasta el final de la linea.
 *   1 <= Fila <= 25.
 *   1 <= Columna <= 80.
 */

int fila;
int columna;

{
    if ( fila > 0 && fila < 25 && columna > 0 && columna <80 )
        printf( "[%d;%df[K", fila, columna );
}


/*------------------------------------------------------------------*/




/* ------------------------------------------------------------------ */
```

```
/*****************************************************************
 *****************************************************************
 **********                                             **********
 **********  GENERACION DEL MODELO GEOMETRICO DIRECTO   **********
 **********                                             **********
 **********             DEL ROBOT  'TH-8'               **********
 **********                                             **********
 *****************************************************************
 *****************************************************************
 **********                                             **********
 **********  AUTORES:                                   **********
 **********                                             **********
 **********          Paulo  Renato Battaglia            **********
 **********          Daniel Juan Franco                 **********
 **********          Domingos S. L. Simonetti           **********
 **********                                             **********
 **********          III EBAI - Curitiba,  02-02-1988   **********
 **********                                             **********
 *****************************************************************
 *****************************************************************/



#include        <math.h>

#define         LINT_ARGS       1
#define         BEEP            printf( "\7" )
#define         CLS             printf( "[2J" )
#define         PI              3.141592654

#define         A2              0.051               /* Valor de 'a2'*/
#define         D6              0.291               /* Valor de 'd6'*/



typedef struct operacionales { double nx, ox, ax, px;
                               double ny, oy, ay, py;
                               double nz, oz, az, pz;
                             };

typedef struct generalizadas { double teta1, teta4, teta5, teta6;
                               double d2, d3;
                             };


double a2 = A2, d6 = D6;        /* Distancias constantes del Robot 'TH-8' */


void   mod_directo( struct generalizadas *,
```

```
                    struct operacionales * );
void   clr_locate( int, int );


/* ------------------------------------------------------------------ */


main()

/*
 * Obtiene el Modelo Geometrico Directo del Robot 'TH-8'.
 */


{
    struct generalizadas lold;    /* posiciones de las articulaciones       */
    struct generalizadas * old;   /* posiciones de las articulaciones       */
    struct operacionales lmatriz; /* Matriz de coordenadas operacionales    */
    struct operacionales * matriz;/* Matriz de coordenadas operacionales    */
    float  auxiliar, auxiliar2;   /* Variable auxiliar para ingreso de datos */
    int    c;                     /* Valor de una tecla presionada          */
    int    i;                     /* Contador                               */

    old = &lold;
    matriz = &lmatriz;



/*
 * Ingreso de datos para la corrida del programa.
 */

    CLS;
    clr_locate( 1, 5 );
    puts( "OBTENCION DEL MODELO GEOMETRICO DIRECTO" );
    clr_locate( 2, 5 );
    puts( "========== === ====== ========== ========" );
    clr_locate( 4, 5 );
    puts( "          DEL ROBOT 'TH-8' " );
    clr_locate( 5, 5 );
    puts( "          === =====  ====   " );

    do {
        for ( i = 8; i <= 24; i++ )
            clr_locate( i, 1 );            /* Borra una zona de pantalla */

        clr_locate( 8, 1 );
        printf( "Ingrese Tetai, en grados : " );
        scanf( "%f", &auxiliar );
        old->tetai = auxiliar * PI / 180.0;
        printf( "Ingrese d2, en metros    : " );
```

```
    scanf( "%f", &auxiliar2 );
    old->d2 = auxiliar2;
    printf( "Ingrese d3, en metros    : " );
    scanf( "%f", &auxiliar );
    old->d3 = auxiliar;
    printf( "Ingrese Teta4, en grados : " );
    scanf( "%f", &auxiliar2 );
    old->teta4 = auxiliar2 * PI / 180.0;
    printf( "Ingrese Teta5, en grados : " );
    scanf( "%f", &auxiliar );
    old->teta5 = auxiliar * PI / 180.0;
    printf( "Ingrese Teta6, en grados : " );
    scanf( "%f", &auxiliar2 );
    old->teta6 = auxiliar2 * PI / 180;

    mod_directo( old, matriz );

    clr_locate( 16, 1 );
    printf( "nx= %f", matriz->nx );
    clr_locate( 16, 17 );
    printf( "ox= %f", matriz->ox );
    clr_locate( 16, 33 );
    printf( "ax= %f", matriz->ax );
    clr_locate( 16, 49 );
    printf( "px= %f", matriz->px );
    clr_locate( 17, 1 );
    printf( "ny= %f", matriz->ny );
    clr_locat( 17, 17 );
    printf( "oy= %f", matriz->oy );
    clr_locate( 17, 33 );
    printf( "ay= %f", matriz->ay );
    clr_locate( 17, 49 );
    printf( "py= %f", matriz->py );
    clr_locate( 18, 1 );
    printf( "nz= %f", matriz->nz );
    clr_locate( 18, 17 );
    printf( "oz= %f", matriz->oz );
    clr_locate( 18, 33 );
    printf( "az= %f", matriz->az );
    clr_locate( 18, 49 );
    printf( "pz= %f", matriz->pz );


    clr_locate( 24, 1 );
    printf( "Desea calcular otro valor ( S/N ) : " );
    do {
        c = getch();
    } while ( c!='s' && c!='S' && c!='n' && c!='N' );

} while ( c == 's' || c == 'S' );
```

```
}

/* ------------------------------------------------------------- */

void mod_directo( coordenadas_generalizadas, matriz )

/*
 * Calcula el modelo directo del Robot TH-8
 *
 */

struct generalizadas * coordenadas_generalizadas;
struct operacionales * matriz;

{
    double si, ci, d2, d3, s4, c4, s5, c5, s6, c6;
    double d[24];                              /* Ecuaciones Auxiliares */
    double tin[3][4];


    si = sin( coordenadas_generalizadas->tetai );
    ci = cos( coordenadas_generalizadas->tetai );
    d2 = coordenadas_generalizadas->d2;
    d3 = coordenadas_generalizadas->d3;
    s4 = sin( coordenadas_generalizadas->teta4 );
    c4 = cos( coordenadas_generalizadas->teta4 );
    s5 = sin( coordenadas_generalizadas->teta5 );
    c5 = cos( coordenadas_generalizadas->teta5 );
    s6 = sin( coordenadas_generalizadas->teta6 );
    c6 = cos( coordenadas_generalizadas->teta6 );


/*
 * Modelo Geometrico Directo obtenido a traves de GMSIR.
 */

    d[1] = ci*a2;
    d[2] = si*a2;
    d[3] = si*d3+d[1];
    d[4] = -ci*d3+d[2];
    d[5] = ci*c4;
    d[6] = ci*s4;
    d[7] = si*c4;
    d[8] = si*s4;
    d[9] = d[5]*c5+si*s5;
    d[10] = d[5]*s5-si*c5;
    d[11] = d[7]*c5-ci*s5;
    d[12] = d[7]*s5+ci*c5;
    d[13] = s4*c5;
```

```
    d[14] = s4*s5;
    d[15] = d[9]*c6+d[6]*s6;
    d[16] = d[9]*s6-d[6]*c6;
    d[17] = d[10]*d6+d[3];
    d[18] = d[11]*c6+d[8]*s6;
    d[19] = d[11]*s6-d[8]*c6;
    d[20] = d[12]*d6+d[4];
    d[21] = d[13]*c6-c4*s6;
    d[22] = d[13]*s6+c4*c6;
    d[23] = d[14]*d6+d2;

    tin[0][0] = d[15];
    tin[0][1] = d[10];
    tin[0][2] = d[16];
    tin[0][3] = d[17];
    tin[1][0] = d[18];
    tin[1][1] = d[12];
    tin[1][2] = d[19];
    tin[1][3] = d[20];
    tin[2][0] = d[21];
    tin[2][1] = d[14];
    tin[2][2] = d[22];
    tin[2][3] = d[23];


/*
 * Carga de los datos obtenidos en la estructura.
 */

    matriz->nx = tin[0][0];
    matriz->ny = tin[1][0];
    matriz->nz = tin[2][0];
    matriz->ox = tin[0][1];
    matriz->oy = tin[1][1];
    matriz->oz = tin[2][1];
    matriz->ax = tin[0][2];
    matriz->ay = tin[1][2];
    matriz->az = tin[2][2];
    matriz->px = tin[0][3];
    matriz->py = tin[1][3];
    matriz->pz = tin[2][3];

}


/* ------------------------------------------------------------------ */


void clr_locate( fila, columna )

/*
```

```
 * Posiciona el cursor en la pantalla y borra hasta el final de la linea.
 *   1 <= Fila <= 25.
 *   1 <= Columna <= 80.
 */

int fila;
int columna;

{
    if ( fila > 0 && fila < 25 && columna > 0 && columna <80 )
        printf( "\033[%d;%df\033[K", fila, columna );
}


/*--------------------------------------------------------------*/
```

```
/*****************************************************************
 *****************************************************************
 **********                                             **********
 **********       GRAFICACION  DE  MOVIMIENTOS          **********
 **********                                             **********
 **********            DEL ROBOT  'TH-8'                **********
 **********                                             **********
 *****************************************************************
 *****************************************************************
 **********                                             **********
 **********  AUTORES:                                   **********
 **********                                             **********
 **********         Paulo  Renato Battaglia             **********
 **********         Daniel Juan Franco                  **********
 **********         Domingos S. L. Simonetti            **********
 **********                                             **********
 **********         III EBAI - Curitiba,  05-02-1988    **********
 **********                                             **********
 *****************************************************************
 ****î*************************************************************/



#include        (math.h)
#include        "graficos.h"

#define         LINT_ARGS       1
#define         BEEP            printf( "\7" )
#define         CLS             printf( "[2J" )
#define         PI              3.141592654

#define         Q1MINIMO        -160*PI/180         /* Minimo tita1     */
#define         Q1MAXIMO         160*PI/180         /* Maximo tita1     */
#define         Q2MINIMO         0.2                /* Minimo d2        */
#define         Q2MAXIMO         0.7                /* Maximo d2        */
#define         Q3MINIMO        -0.7                /* Minimo d3        */
#define         Q3MAXIMO        -0.2                /* Maximo d3        */
#define         Q4MINIMO        -160*PI/180         /* Minimo tita4     */
#define         Q4MAXIMO         160*PI/180         /* Maximo tita4     */
#define         Q5MINIMO        -160*PI/180         /* Minimo tita5     */
#define         Q5MAXIMO         160*PI/180         /* Maximo tita5     */
#define         Q6MINIMO        -160*PI/180         /* Minimo tita6     */
#define         Q6MAXIMO         160*PI/180         /* Maximo tita6     */
#define         A2               0.051              /* Valor de 'a2'    */
#define         D6               0.291              /* Valor de 'd6'    */
#define         INTENTOS        5
#define         VELOCIDAD       1       /* Velocidad de movimiento      */
#define         INC_PX          .1      /* Incremento de px             */
#define         INC_PY          .1      /* Incremento de py             */
#define         INC_PZ          .1      /* Incremento de pz             */
```

```
#define         X_FOCO          10      /* Posicion 'x' del foco       */
#define         Y_FOCO          -10     /* Posicion 'y' del foco       */
#define         Z_FOCO          5       /* Posicion 'z' del foco       */
#define         LINEAS          4       /* Numero de lineas a graficar */
#define         MODO_GRAFICO    6       /* Modo de video 640 * 200     */



typedef struct operacionales { double nx, ox, ax, px;
                               double ny, oy, ay, py;
                               double nz, oz, az, pz;
                             };

typedef struct generalizadas { double teta1, teta4, teta5, teta6;
                               double d2, d3;
                             };


double a2 = A2, d6 = D6;        /* Distancias constantes del Robot 'TH-8' */


void    mod_directo( struct generalizadas *, float * );
int     mod_inverso( struct generalizadas *,
                     struct operacionales *,
                     struct generalizadas *,
                     double *, double * );
double square( double );
void    clr_locate( int, int );
int     espacio_al_plano( float, float, float, float,
                          float, float, float *, float * );
void    imprime_qi( struct generalizadas * );
void    matriz_default( struct operacionales * );
void    imprime_matriz( struct operacionales * );


/* ------------------------------------------------------------------ */


main()

/*
 * Validacion del Modelo Geometrico inverso del Robot 'TH-8'.
 *
 * Devuelve (0) si no hubo errores.
 * Devuelve (1) si hubo algun error.
 * Devuelve (2) si se selecciono la opcion de 'EXIT'.
 *
 */
```

```
{
    struct generalizadas lnew;    /* Nuevas posiciones de las articulaciones */
    struct generalizadas * new;   /* Nuevas posiciones de las articulaciones */
    struct generalizadas lold;    /* Viejas posiciones de las articulaciones */
    struct generalizadas * old;   /* Viejas posiciones de las articulaciones */
    struct operacionales lmatriz; /* Matriz de coordenadas operacionales     */
    struct operacionales * matriz;/* Matriz de coordenadas operacionales     */
    float  t[6][4][4];            /* Matrices intermedias del Modelo Directo */
    double qmin[7], qmax[7];      /* Valores limites de recorrido            */
    float  auxiliar;              /* Variable auxiliar para ingreso de datos */
    float  casos_totales = 1.0;   /* Total de casos para analizar            */
    float  casos_analizados = 0.0;/* Total de casos analizados al momento    */
    int    i, j, k, l, m, n;      /* Contadores de iteraciones               */
    int    c;                     /* Valor de una tecla presionada           */
    int    intentos = INTENTOS;
    int    item, errores = 0;
    int    velocidad = VELOCIDAD; /* Velocidad de movimiento                 */
    int    tecla_invalida;        /* Tecla valida o no valida                */
    float  x_foco = X_FOCO;       /* Posicion 'x' del foco                   */
    float  y_foco = Y_FOCO;       /* Posicion 'y' del foco                   */
    float  z_foco = Z_FOCO;       /* Posicion 'z' del foco                   */
    float  x[10]   , y[10];       /* Puntos actuales en el plano             */
    float  xant[10], yant[10];    /* Puntos anteriores en el plano           */
    int    modo_video;            /* Almacena el modo de video actual        */
    void   line();
    void   window();
    void   polyline();

    new    = &lnew;
    old    = &lold;
    matriz = &lmatriz;


/*
 * Valores de Default para los limites de las articulaciones.
 */

    qmin[1] = Q1MINIMO;
    qmax[1] = Q1MAXIMO;
    qmin[2] = Q2MINIMO;
    qmax[2] = Q2MAXIMO;
    qmin[3] = Q3MINIMO;
    qmax[3] = Q3MAXIMO;
    qmin[4] = Q4MINIMO;
    qmax[4] = Q4MAXIMO;
    qmin[5] = Q5MINIMO;
    qmax[5] = Q5MAXIMO;
    qmin[6] = Q6MINIMO;
    qmax[6] = Q6MAXIMO;
```

```
/*
 * Ingreso de datos para la corrida del programa.
 */

    CLS;
    clr_locate( 2, 5 );
    puts( "SIMULACION DE MOVIMIENTOS DEL ROBOT 'TH-8'" );
    clr_locate( 3, 5 );
    puts( "========== == ========== === =====  ====" );


/*
 * Impresion de los parametros de corrida del programa.
 */

    clr_locate( 7, 1 );
    printf( " 1 - Minimo valor de Thita 1: %f\n", qmin[1]*180.0/PI );
    printf( " 2 - Maximo valor de Thita 1:  %f\n", qmax[1]*180.0/PI );
    printf( " 3 - Minimo valor de d2      :  %f\n", qmin[2] );
    printf( " 4 - Maximo valor de d2      :  %f\n", qmax[2] );
    printf( " 5 - Minimo valor de d3      : %f\n", qmin[3] );
    printf( " 6 - Maximo valor de d3      : %f\n", qmax[3] );
    printf( " 7 - Minimo valor de Thita 4: %f\n", qmin[4]*180.0/PI );
    printf( " 8 - Maximo valor de Thita 4:  %f\n", qmax[4]*180.0/PI );
    printf( " 9 - Minimo valor de Thita 5: %f\n", qmin[5]*180.0/PI );
    printf( "10 - Maximo valor de Thita 5:  %f\n", qmax[5]*180.0/PI );
    printf( "11 - Minimo valor de Thita 6: %f\n", qmin[6]*180.0/PI );
    printf( "12 - Maximo valor de Thita 6:  %f\n", qmax[6]*180.0/PI );
    printf( "13 - Valor de 'a2'           :  %f\n", a2      );
    printf( "14 - Valor de 'd6'           :  %f\n", d6      );


/*
 * Ingreso de datos que se desean modificar.
 */

    clr_locate( 24, 1);
    printf( "Elija una opcion ( 0 = CONT , 15 = EXIT ) : " );

    do {
        do {
            clr_locate( 24, 45 );
            scanf( "%d", &item );
        } while ( item ) 15 || item ( 0 );

        if ( item == 15 )
            exit( 2 );

        clr_locate( 6 + item, 31 );
        clr_locate( 6 + item, 32 );
        if ( item )
```

```
            if ( item == 14 ) {
                sanf( "%f", &auxiliar );
                d6 = auxiliar;
                cr_locate( 6 + item, 31 );
                printf( " %f", d6 );
            } else
                if ( item == 13 )  {
                    scanf( "%f", &auxiliar );
                    a2 = auxiliar;
                    clr_locate( 6 + item, 31 );
                    printf( " %f", a2 );
                } else
                    if ( item & 1 )  {            /* Es una opcion impar */
                        scanf( "%f", &auxiliar );
                        if ( item < 3 || item > 6 )
                            qmin[ (item + 1) / 2 ] = auxiliar * PI / 180.0;
                        clr_locate( 6 + item, 31 );
                        if ( auxiliar > 0 )
                            printf( " %f", auxiliar );
                        else
                            printf( "%f", auxiliar );
                    } else  {                      /* Es una opcion par   */
                        scanf( "%f", &auxiliar );
                        if ( item < 3 || item > 6 )
                            qmax[ item / 2 ] = auxiliar * PI / 180.0;
                        clr_locate( 6 + item, 31 );
                        if ( auxiliar > 0 )
                            printf( " %f", auxiliar );
                        else
                            printf( "%f", auxiliar );
                    }

    } while ( item );                          /* Termina cuando es cero */


/*
 * Inicializacion de las articulaciones de Robot.
 */

    matriz_default( matriz );


/*
 * Comienzo del loop grafico.
 */

    modo_video = getmode();              /* Guarda el modo de video actual */
    init( MODO_GRAFICO );                /* Inicializa modo grafico        */

    x[0] = y[0] = 0;                     /* La base del robot esta en el
```

```
                                                origen de cooordenadas   */
    window( -.5, -0.1, 1.9, 1.2 );      /* Establece el formato de pantalla */


    c = mod_inverso( old, matriz, new, qmin, qmax );
    if ( c ) {
        clr_locate( 24, 62 );
        printf( "Fuera del Volumen" );
    } else {
        clr_locate( 24, 62 );
                                /* _                                        */
        old->teta1 = new->teta1; /*  |                                       */
        old->d2    = new->d2   ; /*  |                                       */
        old->d3    = new->d3   ; /*  |\ Obtiene la posicion anterior         */
        old->teta4 = new->teta4; /*  |/    para la proxima vuelta.           */
        old->teta5 = new->teta5; /*  |                                       */
        old->teta6 = new->teta6; /* _|                                       */

        mod_directo( new, t );


        /* Pasa del espacio al plano */

        espacio_al_plano( 0.0, 0.0, (float)new->d2,
                          x_foco, y_foco, z_foco,
                          &(x[1]), &(y[1])  );
        espacio_al_plano( t[1][0][3], t[1][1][3], t[1][2][3],
                          x_foco, y_foco, z_foco,
                          &(x[2]), &(y[2])  );
        espacio_al_plano( t[3][0][3], t[3][1][3], t[3][2][3],
                          x_foco, y_foco, z_foco,
                          &(x[3]), &(y[3])  );
        espacio_al_plano( t[5][0][3], t[5][1][3], t[5][2][3],
                          x_foco, y_foco, z_foco,
                          &(x[4]), &(y[4])  );


        for ( i = 0 ; i < 10; i++ ) {
            xant[i] = x[i];
            yant[i] = y[i];                 /* Guarda los puntos anterires */
        }

        polyline( x, y, LINEAS, 1 );            /* Dibuja el primer Robot   */

    }

    rline( -0.15, 0.0, 0.15, 0.0, 1 );          /* Dibuja la base del robot */

/*
```

```
* Imprime los parametros del Robot.
*/

   clr_locate( 1, 15 );
   printf( "Simulacion del Robot  'TH-8'" );

   line( 463, 200, 463, 0, 1 );
   line( 465, 200, 465, 0, 1 );
   line( 0, 180, 463, 180, 1 );
   line( 0, 182, 463, 182, 1 );
   line( 465, 43, 640, 43, 1);
   line( 465, 45, 640, 45, 1 );
   line( 465, 101, 640, 101, 1);
   line( 465, 99, 640, 99, 1 );
   line( 465, 30, 640, 30, 1);
   line( 465, 28, 640, 28, 1 );

   imprime_matriz( matriz ); /* Imprime los valores de la matriz operacional*/

   clr_locate( 21, 60 );
   printf( "Velocidad : %d", velocidad );

   imprime_qi( new );


   do {

      /* Espera a que se presione una tecla valida */

      do {
         tecla_invalida = 0;
         c = getch();
         if ( c == 0 )
            c = getch();          /* Ciertas teclas emiten '0' + 'Codigo */
         switch ( c )  {
            case 27 :             /* Se presiono 'ESC', fin de programa  */
               break;
            case 59 :
               if ( velocidad < 16 )      /* 'F1' aumenta la velocidad */
                  velocidad = velocidad * 2;
               clr_locate( 21, 60 );
               printf( "Velocidad : %d", velocidad );
               break;
            case 60 :
               if ( velocidad > 1 )       /* 'F2' baja la velocidad    */
                  velocidad = velocidad / 2;
               clr_locate( 21, 60 );
               printf( "Velocidad : %d", velocidad );
               break;
            case 'd':
            case 'D':
```

```
            matriz_default( matriz ); /* Pone los valores de default */
            imprime_matriz( matriz );
            break;
        case '7':                                   /* Aumenta 'px' */
        case 71 :
            matriz->px += velocidad * INC_PX;
            clr_locate( 1, 60 );
            printf( "Px  : %f", (float)matriz->px );
            break;
        case '8':                                   /* Aumenta 'py' */
        case 72 :
            matriz->py += velocidad * INC_PX;
            clr_locate( 2, 60 );
            printf( "Py  : %f", (float)matriz->py );
            break;
        case '9':                                   /* Aumenta 'pz' */
        case 73 :
            matriz->pz += velocidad * INC_PX;
            clr_locate( 3, 60 );
            printf( "Pz  : %f", (float)matriz->pz );
            break;
        case '1':                                /* Disminuye 'px' */
        case 79 :
            matriz->px -= velocidad * INC_PX;
            clr_locate( 1, 60 );
            printf( "Px  : %f", (float)matriz->px );
            break;
        case '2':                                /* Disminuye 'py' */
        case 80 :
            matriz->py -= velocidad * INC_PX;
            clr_locate( 2, 60 );
            printf( "Py  : %f", (float)matriz->py );
            break;
        case '3':                                /* Disminuye 'pz' */
        case 81 :
            matriz->pz -= velocidad * INC_PX;
            clr_locate( 3, 60 );
            printf( "Pz  : %f", (float)matriz->pz );
            break;


        default :                  /* Se presiono una tecla invalida */
            tecla_invalida = 1;
            BEEP;
    }
} while ( tecla_invalida );                /* Sigue esperando tecla  */


if ( c != 27 ) {
    c = mod_inverso( old, matriz, new, qmin, qmax );
    if ( c ) {
```

```
            clr_locate( 24, 60 );
            printf( "Fuera del Volumen" );
         } else {
            clr_locate( 24, 60 );

                                    /* _                                  */
            old->teta1 = new->teta1;  /*  |                                 */
            old->d2    = new->d2   ;  /*  |                                 */
            old->d3    = new->d3   ;  /*  |\ Obtiene la posicion anterior */
            old->teta4 = new->teta4;  /*  |/    para la proxima vuelta.    */
            old->teta5 = new->teta5;  /*  |                                 */
            old->teta6 = new->teta6;  /* _|                                 */

            imprime_qi( new );                    /* Imprime resultados */

            mod_directo( new, t );           /* Obtiene puntos a graficar */


            /* Pasa del espacio al plano */

            espacio_al_plano( 0.0, 0.0, (float)new->d2,
                              x_foco, y_foco, z_foco,
                              &(x[1]), &(y[1])  );
            espacio_al_plano( t[1][0][3], t[1][1][3], t[1][2][3],
                              x_foco, y_foco, z_foco,
                              &(x[2]), &(y[2])  );
            espacio_al_plano( t[3][0][3], t[3][1][3], t[3][2][3],
                              x_foco, y_foco, z_foco,
                              &(x[3]), &(y[3])  );
            espacio_al_plano( t[5][0][3], t[5][1][3], t[5][2][3],
                              x_foco, y_foco, z_foco,
                              &(x[4]), &(y[4])  );



            polyline( xant, yant, LINEAS, 0 );
                                          /* Borra el robot anterior  */

            for ( i = 0 ; i < 10; i++ ) {
               xant[i] = x[i];
               yant[i] = y[i];          /* Guarda los puntos anteriores */
            }

            polyline( x, y, LINEAS, 1 );     /* Dibuja nuevo robot       */

         }
      }

   } while ( c != 27 );              /* Termina cuando se presiona 'ESC'    */

   init( modo_video );                    /* Reestablece el modo anterior   */
```

```
    clr_locate( 24, 1 );
    printf( "Presione alguna tecla para finalizar." );
    c = getch();
    CLS;

    if ( errores )
        exit( 1 );
    else
        exit(0);
}


/* ------------------------------------------------------------- */

void mod_directo( coordenadas_generalizadas, t )

/*
 * Calcula el modelo directo del Robot TH-8
 *
 */

struct generalizadas * coordenadas_generalizadas;
float  t[][4][4];

{
    double si, ci, d2, d3, s4, c4, s5, c5, s6, c6;
    double d[24];                              /* Ecuaciones Auxiliares */


/*
 * Calculo de senos y cosenos.
 */

    si = sin( coordenadas_generalizadas->tetai );
    ci = cos( coordenadas_generalizadas->tetai );
    d2 = coordenadas_generalizadas->d2;
    d3 = coordenadas_generalizadas->d3;
    s4 = sin( coordenadas_generalizadas->teta4 );
    c4 = cos( coordenadas_generalizadas->teta4 );
    s5 = sin( coordenadas_generalizadas->teta5 );
    c5 = cos( coordenadas_generalizadas->teta5 );
    s6 = sin( coordenadas_generalizadas->teta6 );
    c6 = cos( coordenadas_generalizadas->teta6 );


/*
 * Modelo Geometrico Directo obtenido a traves de GMSIR.
 */

    d[1] = ci*a2;
```

```
    d[2] = si*a2;
    d[3] = si*d3+d[1];
    d[4] = -ci*d3+d[2];
    d[5] = ci*c4;
    d[6] = ci*s4;
    d[7] = si*c4;
    d[8] = si*s4;
    d[9] = d[5]*c5+si*s5;
    d[10] = d[5]*s5-si*c5;
    d[11] = d[7]*c5-ci*s5;
    d[12] = d[7]*s5+ci*c5;
    d[13] = s4*c5;
    d[14] = s4*s5;
    d[15] = d[9]*c6+d[6]*s6;
    d[16] = d[9]*s6-d[6]*c6;
    d[17] = d[10]*d6+d[3];
    d[18] = d[11]*6+d[8]*s6;
    d[19] = d[11]*s6-d[8]*c6;
    d[20] = d[12]*d6+d[4];
    d[21] = d[13]*c6-c4*s6;
    d[22] = d[13]*s6+c4*c6;
    d[23] = d[14]*d6+d2;


/*
 * Matrices intermedias del Modelo Directo.
 *
 * Solo se calculan los puntos de interes para graficar.
 * Estos puntos son 'px', 'py', 'pz'.
 */

   t[1][0][3] = d[1];
   t[1][1][3] = d[2];
   t[1][2][3] = d2;

   t[3][0][3] = d[3];
   t[3][1][3] = d[4];
   t[3][2][3] = d2;

   t[5][0][3] = d[17];
   t[5][1][3] = d[20];
   t[5][2][3] = d[23];

}


/* ------------------------------------------------------------------ */


int mod_inverso( old, matriz, new, qmin, qmax )
```

```
/*
 * Calcula el modelo inverso del Robot TH-8.
 *
 * Devuelve (0) si se pudo calcular el modelo inverso.
 * Devuelve (1) si no se pudo calcular el modelo inverso.
 *
 */

#define   GASTO_4     1.0     /* Constante de energia de la articulacion 1 */
#define   GASTO_5     1.0     /* Constante de energia de la articulacion 1 */
#define   GASTO_6     1.0     /* Constante de energia de la articulacion 1 */


struct generalizadas * old;    /* Posicion anterior de
                                          las articulaciones     */
struct operacionales * matriz; /* Coordenadas del espacio
                                          a ser alcanzadas       */
struct generalizadas * new;    /* Posicion de las articulaciones       */
double qmin[];                 /* Minimos valores de las articulaciones */
double qmax[];                 /* Maximos valores de las articulaciones */

{
    double r1, fi1;      /* Variables auxiliares                          */
    double s5, c5;       /* Seno y Coseno de 'teta5'                      */
    double si, ci;       /* Seno y Coseno de 'tetai'                      */
    double auxiliar;     /* Variable auxiliar                             */
    double auxiliar_4;   /* Almacena temporariamente 'teta4'              */
    double auxiliar_5;   /* Almacena temporariamente 'teta5'              */
    double auxiliar_6;   /* Almacena temporariamente 'teta6'              */
    double pondera_1;    /* Pondera la primera solucion                   */
    double pondera_2;    /* Pondera la segunda solucion                   */
    int    error_1 = 0;  /* Condicion de error de la primera solucion     */
    int    error_2 = 0;  /* Condicion de error de la segunda solucion     */



/*
 * Calculo de 'd2'.
 */

    auxiliar = matriz->pz - matriz->oz * d6;
    if ( auxiliar < qmin[2] || auxiliar > qmax[2] )
        return( 1 );
    else
        new->d2 = auxiliar;


/*
 * Calclo de 'd3'.
 */
```

```
    if ( ( auxiliar = square(matriz->px - matriz->ox * d6) +
           square(matriz->py - matriz->oy * d6) ) < 0.0 )
        return( 1 );
    else
        r1 = sqrt( auxiliar );

    fi1 = atan2( matriz->px - matriz->ox * d6, matriz->py - matriz->oy * d6 );

    auxiliar = r1 * r1 + a2 * a2;
    if ( (auxiliar > (qmin[3] * qmin[3])) || (auxiliar < (qmax[3] * qmax[3])) )
        return( 1 );      /* Tener en cuenta que ABS(qmin[3]) > ABS(qmax[3]) */
    else
        new->d3 = -sqrt( auxiliar );


/*
 * Calculo de 'teta1'.
 */

    new->teta1 = atan2( a2, -(new->d3) ) - fi1;
    if ( new->teta1 < qmin[1] || new->teta1 > qmax[1] )
        return( 1 );

    s1 = sin( new->teta1 );           /* Seno de 'teta1'   */
    c1 = cos( new->teta1 );           /* Coseno de 'teta1' */


/*
 * Calculo de 'teta4' 'teta5' 'tta6'.
 */

    auxiliar = c1 * matriz->oy - s1 * matriz->ox;
    if ( fabs( auxiliar ) > 1 )
        return( 1 );
    else
        c5 = auxiliar;

    /* Solucion cuando coseno(teta1) = +/- 1  */

    if ( fabs(c5) > 0.99999 )
    {
        if ( c5 > 0.0 )
            new->teta5 = 0;           /* Calcula 'teta5'      */
        else
            return( 1 );              /* tita5 no puede ser PI */

        new->teta4 = old->teta4;      /* Calcula 'teta4'      */

                                      /* Calcula ' teta6'     */
        new->teta6 = new->teta4 - atan2( matriz->nz, matriz->az );
        if ( new->teta6 > PI )
```

```
        new->teta6 -= 2*PI;        /* Corrige si teta6 > PI */
    if ( new->teta6 < -PI )
        new->teta6 += 2*PI;        /* Corrige si teta6 < -PI */
    if ( new->teta6 < qmin[6] || new->teta6 > qmax[6] )
        return( 1 );

    return( 0 );
}

/* Solucion cuando coseno(teta5) != +/- 1  */

s5 = sqrt( 1 - c5 * c5 );

auxiliar_5 = atan2( s5, c5 );          /* Calcula 'teta5' para s5 > 0 */
if ( auxiliar_5 < qmin[5] || auxiliar_5 > qmax[5] )
    error_1 = 1;
else {
    auxiliar_4 = atan2( matriz->oz , ci * matriz->ox + si * matriz->oy );
                                                    /* Calcula 'teta4' */
    if ( auxiliar_4 < qmin[4] || auxiliar_4 > qmax[4] )
        error_1 = 1;
    else {
        auxiliar_6 = atan2( si * matriz->ax - ci * matriz->ay,
                            si * matriz->nx - ci * matriz->ny );
                                                    /* Calcula 'teta6' */
        if ( auxiliar_6 < qmin[6] || auxiliar_6 > qmax[6] )
            error_1 = 1;
    }
}

if ( error_1 == 0 )                    /* Pondera la primera solucion */
    pondera_1 = (auxiliar_4 - old->teta4) * GASTO_4 +
                (auxiliar_5 - old->teta5) * GASTO_5 +
                (auxiliar_6 - old->teta6) * GASTO_6 ;




new->teta5 = atan2( -s5, c5 );         /* Calcula 'teta5' para s5 < 0 */
if ( new->teta5 < qmin[5] || new->teta5 > qmax[5] )
    error_2 = 1;
else {
    new->teta4 = atn2( - matriz->oz ,
                       - ci * matriz->ox - si * matriz->oy );
                                                    /* Calcula 'teta4' */
    if ( new->teta4 < qmin[4] || new->teta4 > qmax[4] )
        error_2 = 1;
    else {
        new->teta6 = atan2( - si * matriz->ax + ci * matriz->ay,
                    - si * matriz->nx + ci * matriz->ny );
```

```
                                                    /* Calcula 'teta6' */
            if ( new->teta6 ( qmin[6] || new->teta6 ) qmax[6] )
                error_2 = 1;
        }
    }



    if ( error_1 == 0 )                     /* Pondera la segunda solucion */
        pondera_2 = (new->teta4 - old->teta4) * GASTO_4 +
                    (new->teta5 - old->teta5) * GASTO_5 +
                    (new->teta6 - old->teta6) * GASTO_6 ;

/*
 * Elige la mejor solucion.
 */

    if ( error_1 & error_2 )
        return( 1 );                        /* No hay solucion             */
    if ( error_1 )
        return( 0 );                        /* Se elige la opcion '2'      */
    if ( error_2) {
        new->teta4 = auxiliar_4;
        new->teta5 = auxiliar_5;
        new->teta6 = auxiliar_6;
        return( 0 );                        /* Se elije la opcion '1'      */
    }
    if ( pondera_2 (= pondera_1 )
        return( 0 );                        /* La opcion '2' es la mejor   */

    new->teta4 = auxiliar_4;
    new->teta5 = auxiliar_5;
    new->teta6 = auxiliar_6;

    return( 0 );                            /* La opcion '1' es la mejor   */

}


/* ------------------------------------------------------------------- */


double square( numero )

/*
 * Eleva un numero al cuadrado.
 */

double numero;
```

```
{
    return( numero * numero );
}


/* ------------------------------------------------------------------ */


void clr_locate( fila, columna )

/*
 * Posiciona el cursor en la pantalla y borra hasta el final de la linea.
 *   1 (= Fila (= 25.
 *   1 (= Columna (= 80.
 */

int fila;
int columna;

{
    if ( fila ) 0 && fila ( 25 && columna ) 0 && columna (80 )
        printf( "[%d;%dfCK", fila, columna );
}


/*------------------------------------------------------------------*/


int espacio_al_plano( xp, yp, zp, xf, yf, zf, x, y )

/*
 * Transforma un punto en el espacio a un
 *     punto en el plano mediante perspectiva
 *     con dos puntos de fuga.
 *
 * Las coordenadas 'xp - yp' corresponden a la
 *     vista desde arriba del punto.
 * La coordenada 'zp' es la altura del punto.
 *
 * Devuelve (0) si se pudo calcular el punto.
 * Devuelve (1) si no se puede calcular el punto.
 *
 */

float  xp;              /* Coordenada 'x' del punto en el espacio */
float  yp;              /* Coordenada 'y' del punto en el espacio */
float  zp;              /* Coordenada 'z' del punto en el espacio */
float  xf;              /* Coordenada 'x' del foco en el espacio  */
float  yf;              /* Coordenada 'y' del foco en el espacio  */
float  zf;              /* Coordenada 'z' del foco en el espacio  */
float  *x;              /* Coordenada 'x' en el plano             */
```

```
float  *y;              /* Coordenada 'y' en el plano            */

{
    if ( (yp - yf) < .0001 )
       return( 1 );            /* No se puede calcular el punto */

    *x = xf - yf * ( xp - xf ) / ( yp - yf );
    *y = zf - yf * ( zp - zf ) / ( yp - yf );

    return( 0 );
}


/* ------------------------------------------------------------------ */


void imprime_qi( new )

struct generalizadas * new;

{
    clr_locate( 14, 60 );
    printf( "Teta1 : %f", (float)new->teta1 );
    clr_locate( 15, 60 );
    printf( "d2    : %f", (float)new->teta1 );
    clr_locate( 16, 60 );
    printf( "d3    : %f", (float)new->teta1 );
    clr_locate( 17, 60 );
    printf( "Teta4 : %f", (float)new->teta4 );
    clr_locate( 18, 60 );
    printf( "Teta5 : %f", (float)new->teta5 );
    clr_locate( 19, 60 );
    printf( "Teta6 : %f", (float)new->teta6 );
}


/* ------------------------------------------------------------------ */


void matriz_default( matriz )

/*
 * Obtiene los parametros de default del Robot.
 */

struct operacionales *matriz;

{

    matriz->nx = 1.0;
    matriz->ny = 0.0;
```

```
    matriz->nz = 0.0;
    matriz->ox = 0.0;
    matriz->oy = 1.0;
    matriz->oz = 0.0;
    matriz->ax = 0.0;
    matriz->ay = 0.0;
    matriz->az = 1.0;
    matriz->px = 0.051;
    matriz->py = 0.541;
    matriz->pz = 0.5;

}


/* ------------------------------------------------------------ */


void imprime_matriz( matriz )

/*
 * Imprime los valores de la matriz operacional.
 */

struct operacionales *matriz;

{
    clr_locate( 1, 60 );
    printf( "Px  : %f", (float)matriz->px );
    clr_locate( 2, 60 );
    printf( "Py  : %f", (float)matriz->py );
    clr_locate( 3, 60 );
    printf( "Pz  : %f", (float)matriz->pz );
    clr_locate( 4, 60 );
    printf( "Nx  : %f", (float)matriz->nx );
    clr_locate( 5, 60 );
    printf( "Ny  : %f", (float)matriz->ny );
    clr_locate( 6, 60 );
    printf( "Nz  : %f", (float)matriz->nz );
    clr_locate( 7, 60 );
    printf( "Ox  : %f", (float)matriz->ox );
    clr_locate( 8, 60 );
    printf( "Oy  : %f", (float)matriz->oy );
    clr_locate( 9, 60 );
    printf( "Oz  : %f", (float)matriz->oz );
    clr_locate( 10, 60 );
    printf( "Ax  : %f", (float)matriz->ax );
    clr_locate( 11, 60 );
    printf( "Ay  : %f", (float)matriz->ay );
    clr_locate( 12, 60 );
    printf( "Az  : %f", (float)matriz->az );
}
```