ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL
DA TESE DEFENDIDA POR Benedito Luis
Fayan E APROVADA PELA
COMISSÃO JULGADORA EM 16/12/92

João Maurício Rosário
ORIENTADOR

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

Estudo e Especificação de um Supervisor de Controle para um Robô Industrial

Benedito Luis Fayan

Orientador : Prof. Dr. João Maurício Rosário OK

Dissertação apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de mestre em engenharia mecânica.

Campinas - SP
dezembro 1992

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Tese de: Mestrado

Título da Tese: Estudo e especificação de um Supervisor de Controle para um Robô Industrial

Autor: Benedito Luis Fayan
Orientador: Prof. Doutor João Maurício Rosário

Aprovado por:

Prof. Dr. João Maurício Rosário, Presidente

Prof. Dr. Edson Moschim

Prof. Dr. Paulo Roberto Gardel Kurka

Campinas, 16 de dezembro de 1992.

AGRADECIMENTOS

Ao término de mais uma etapa de minha vida acadêmica, tenho aqui a oportunidade de agradecer e homenagear algumas pessoas que me ajudaram a completar a tarefa árdua que é a conclusão do Mestrado.

Gostaria primeiramente de prestar uma homenagem especial ao professor e amigo Doutor João Maurício Rosário, pelo incentivo e acompanhamento sempre presente durante todo o transcorrer do trabalho, e pela enorme capacidade de trabalho e entusiasmo com que enfrenta os seus desafios, sendo um verdadeiro exemplo de dedicação profissional.

Ao meu amigo Carlos Henrique Dias, pelo companheirismo e ajuda em todas as fases do projeto.

A Magali Akemi Shibana pela leitura dos capítulos, pelas sugestões apresentadas e pela paciência demonstrada quando do transcorrer das intermináveis noites e fins de semana de estudo.

A FAPESP pelo financiamento do projeto.

A Dárcio e Elza,
meus pais

RESUMO

O objetivo principal deste trabalho é a especificação de um supervisor de controle para um robô industrial. Primeiramente são introduzidos conceitos de robôs manipuladores e apresentados as diversas formas de modelagem. Dentre os vários modelos, ênfase especial é dada à modelagem cinemática inversa, visto ser essencial para uma das aplicações fundamentais que o supervisor de controle especificado deve possuir, que é controlar a posição do efetuador do robô no espaço cartesiano.

Diversos algoritmos para a modelagem cinemática inversa foram desenvolvidos, sendo detalhadas as suas características e então sendo feitas as comparações para a escolha do algoritmo que melhor se adequa à função de controle no espaço no espaço cartesiano.

As arquiteturas de hardware e de software propostas para o supervisor são abordadas, sendo feita a descrição funcional das diferentes unidades processadoras que compõem o sistema e a descrição das interfaces dos vários módulos de software especificados.

ÍNDICE

CAPÍTULO 4 : ARQUITETURAS DE CONTROLE

CAPÍTULO 5 : ARQUITETURA DE SOFTWARE

CAPÍTULO 6 : ANÁLISE DE DESEMPENHO E CONCLUSÕES

CAPITULO 1 : INTRODUÇÃO

1.1 CONTEXTO DO TRABALHO DE TESE

O alto custo e a dificuldade de utilização de controladores industriais em robôs, sempre mal adaptados à operações que requerem flexibilidade de utilização, tais como intervenções em meio hostis, exigem o desenvolvimento de estruturas de controle mais simples e economicamente viáveis e adaptadas às operações requeridas.

Deve-se ressaltar que a falta de conhecimento dos elementos do sistema e a dificuldade de acesso à estrutura interna dos supervisores de controle utilizados em robôs industriais obriga o usuário a adquirir pacotes computacionais e interfaces especiais para cada nova aplicação desejada, pois é difícil se encontrar robôs e supervisores de controle com características ideais para uma determinada aplicação, sendo geralmente necessária uma adequação que é muitas vezes complexa e onerosa.

Este trabalho tem como objetivo a estruturação de um supervisor de controle para um robô manipulador, a partir da definição de uma arquitetura modular, utilizando microprocessadores em forma hierarquizada e interfaces padronizadas. Esta estrutura permite a divisão das várias tarefas realizadas, bem como sua simples repetição em alguns processadores, tais como os algoritmos de controle de cada junta, onde cada processador trabalha de forma totalmente desacoplada em relação aos demais processadores que controlam os outros graus de liberdade de um robô. Ênfase especial é dada ao desenvolvimento de algoritmos para modelagem cinemática de robôs, sendo feita uma comparação dos resultados obtidos na simulação dos diversos algoritmos estudados. Um dos objetivos também é o de explorar e compreender todas as funcionalidades necessárias ao desenvolvimento de tal tipo de equipamento.

O desenvolvimento de um supervisor de controle se insere no contexto do projeto de cooperação científica entre a UNICAMP, o Instituto Tecnológico de Geesthacht - GKSS da Alemanha e a Petrobrás, que tem por objetivo o desenvolvimento de estruturas de controle apropriadas para a utilização em manipuladores que operem em condições adversas, tais como o fundo do mar. Para esta finalidade , uma maquete de plataforma submarina em escala real foi construída no Centro de Tecnologia da UNICAMP, onde está sendo desenvolvido um sistema de trilhos que permitirá o deslocamento de um robô industrial, e ainda serão simuladas as diversas tarefas automatizadas a serem realizadas no fundo do mar.

1.2 ESTRUTURA GERAL DE UM SUPERVISOR DE CONTROLE

A estrutura completa do supervisor de controle em desenvolvimento é mostrado na figura 1.1. A partir de uma interface homem-máquina amigável, um software de visualização do robô e de seu ambiente de trabalho possibilita a geração de arquivos de trajetória, os quais posteriormente são transmitidos à estrutura de controle propriamente dita.

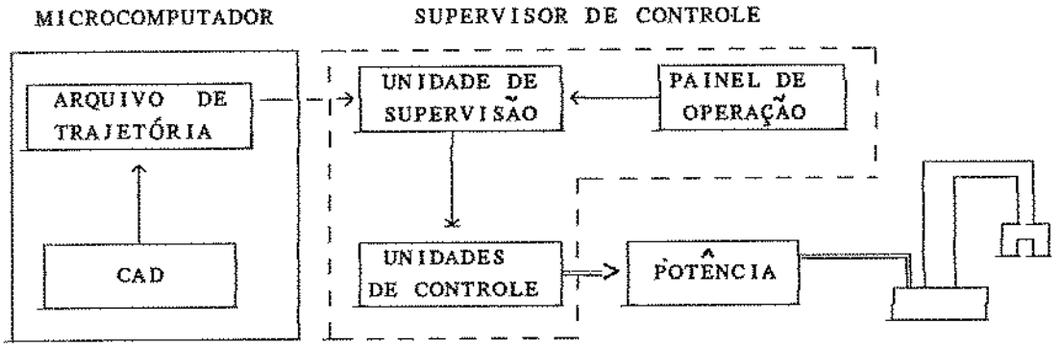


Figura 1.1 : Estrutura do Supervisor de Controle

Para a elaboração desta estrutura, são abordados no presente trabalho os seguintes aspectos, que são essenciais para a execução das aplicações especificadas para este supervisor :

- definição da arquitetura de controle
- geração de trajetórias
- modelagem de manipuladores
- algoritmos para modelagem cinemática inversa

1.3 DESCRIÇÃO DOS CAPÍTULOS POSTERIORES

O desenvolvimento dos vários aspectos acima relacionados compõe alguns dos capítulos do presente trabalho. Descreve-se a seguir, suscintamente, a ordem e o conteúdo dos capítulos posteriores.

No capítulo 2 são apresentadas noções gerais sobre os robôs industriais, seus principais componentes e formas de acionamento. É feita uma introdução do conceito de modelagem direta e inversa de manipuladores pois o uso dos mesmos é fundamental no controle de um robô. Através desses modelos é feita a transformação das coordenadas articulares dos vários graus de liberdade do robô em coordenadas cartesianas, assim como também a transformação inversa.

No capítulo 3 introduz-se o modelo do robô industrial Manutec-r3, que foi o modelo usado nas simulações dos algoritmos de geração de trajetória utilizando o modelo cinemático inverso. Em seguida, apresenta-se alguns tipos de algoritmos de modelagem cinemática inversa que foram desenvolvidos durante o transcorrer do trabalho e os resultados das simulações efetuadas.

No capítulo 4 são abordados vários tipos de arquitetura de controle propostas na literatura, sendo mostrada uma visão geral das mesmas. Também é apresentada a estrutura de controle proposta neste trabalho, com a descrição da diversas características do sistema. A arquitetura descrita baseia-se numa abordagem hierárquica para os níveis de controle, com uma estrutura modular com interfaces padronizadas, tanto a nível de hardware como de software.

É feita também a descrição das características do sistema abordando as diversas possibilidades referentes à geração de trajetórias, mostrando as alternativas possíveis para a realização desta função no sistema. Descreve-se também a arquitetura hardware e as interfaces existentes entre as unidades de controle.

No capítulo 5 descreve-se a arquitetura de software do sistema, e a especificação dos módulos de software necessários, detalhando-se as mensagens presentes nas interfaces entre estes, bem como os parâmetros por eles controlados. Esta especificação busca tornar o sistema modular e flexível, para que no futuro, se houver necessidade, seja possível a substituição e inclusão de outros módulos de maneira simples.

Finalmente, no capítulo 6 é realizada uma análise do desempenho esperado do sistema, explorando as situações mais críticas de processamento. Este tipo de análise nos leva *a priori* à detecção dos pontos de estrangulamento do sistema. De posse desses dados são analisadas algumas perspectivas futuras de desenvolvimento para uma melhoria de desempenho do mesmo, permitindo a concentração de esforços nos pontos onde há uma melhor relação entre o custo de desenvolvimento e melhoria de performance.

O desenvolvimento e elucidação de alguns aspectos informativos que não foram considerados essenciais a este trabalho são mostrados separadamente em forma de Anexos.

CAPÍTULO 2: CARACTERÍSTICAS GERAIS DE ROBÔS MANIPULADORES

Neste capítulo descreve-se inicialmente os principais componentes que compõem a estrutura de um robô, assim como as tecnologias empregadas em sua fabricação. Em seguida é feita uma introdução à modelagem de robôs, com a descrição dos modelos geométrico, cinemático e dinâmico, com suas características associadas. Por último aborda-se as aplicações para o modelamento cinemático inverso e o uso de algoritmos de inversão adequados para cada aplicação.

2.1 PRINCIPAIS COMPONENTES

Um robô é composto basicamente de quatro componentes principais [1], que são:

1) Manipulador
2) Sensores
3) Controlador
4) Elementos de potência

A figura 2.1 ilustra estes componentes interligados em um sistema.

2.1.1 Manipulador

A estrutura mecânica ilustrada na figura 2.1 consiste de um manipulador com elementos mecânicos interligados por meios de juntas, formando uma cadeia cinemática aberta.

Para cada junta do manipulador podemos definir um eixo, ao longo do qual o elemento ligado a ela se movimenta. Este movimento pode ser do tipo rotacional (junta rotacional) e do tipo translacional (junta prismática).

O número de graus de liberdade dos manipuladores é igual ao número total de juntas que o compõem. Geralmente manipuladores industriais apresentam seis graus de liberdade, sendo três para posicionamento do robô no espaço de operação e três para orientação do elemento terminal. Um outro grau de liberdade pode ainda ser definido para preensão de objetos.

Além dos componentes mecânicos, os manipuladores contém dispositivos para a realização do controle dos movimentos de seus vários elementos e juntas. Estes dispositivos são os atuadores ou acionadores, que são de origem pneumática, hidráulica ou elétrica; os sensores de posição, rotação e torque; as chaves de fim de curso; os relés e outros.

O uso de sensores tem por finalidade informar as condições do manipulador, obter informações sobre a aceleração, velocidade e posição das juntas. Através da realimentação destas variáveis para a unidade de controle, pode-se controlar adequadamente a estrutura mecânica associada.

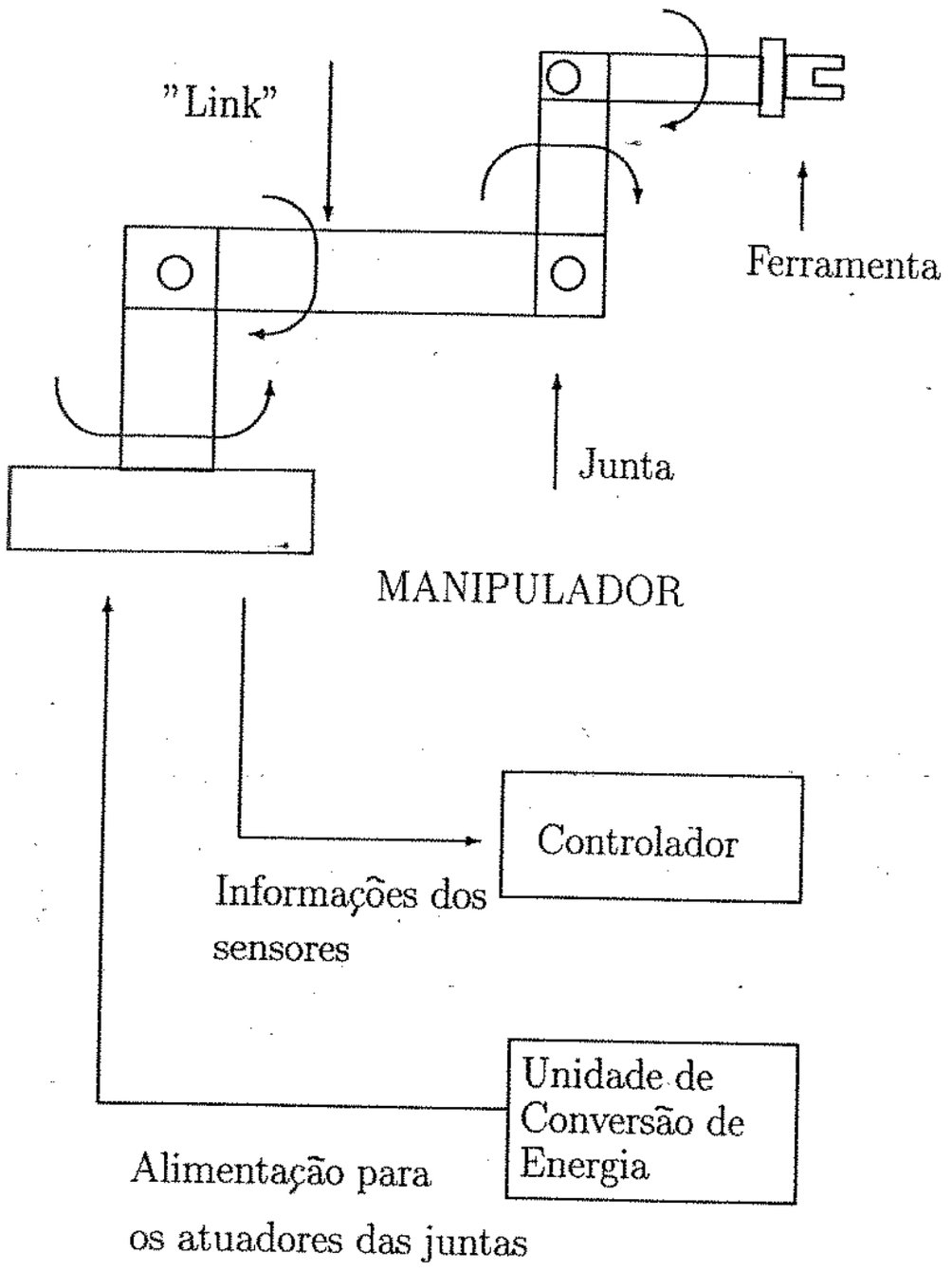


Figura 2.1 : Estrutura de um manipulador

Os sensores utilizados nos manipuladores podem ser divididos em dois grupos, chamados de sensores visuais e sensores não visuais.

Os sensores não visuais incluem chaves limitadoras (sensor de proximidade, fotoelétrico ou mecânico), sensores de posição (encoders, potenciômetros e resolvers), sensores de velocidade (tacômetros), sensores de força (células de carga).

Os sensores visuais consistem de câmeras CCD integradas ao hardware e software de detecção de imagem. Estes sensores podem ser utilizados para a realimentação da posição do efetuador, para reconhecimento e preensão de objetos.

Um dos sensores de posição mais utilizados é o encoder. Este é um dispositivo incremental que, acoplado a uma junta, fornece em sua saída uma posição angular relativa no espaço e através de tratamento lógico fornece também o sentido de rotação. Além destas informações, o encoder indica quando a junta atinge um ponto de referência pré-determinado.

Estas informações são fornecidas pelo encoder através de três sinais. Um desses sinais indica que o encoder passou pelo ponto de referência, enquanto que os outros sinais, defasados entre si de noventa graus, fornecem o número de impulsões.

Com base nestes dois sinais, a frequência de rotação da junta é determinada pela contagem do número de impulsões existentes num determinado intervalo de tempo e o sentido pode ser obtido através da análise de qual sinal está adiantado em relação ao outro.

2.1.2 Controlador

Pode-se considerar um manipulador como sendo um dispositivo mecânico de posicionamento. Para que o manipulador, dentro do espaço de trabalho, execute tarefas de posicionamento do seu efetuador com rapidez e precisão, é necessário que este apresente um sistema de controle capaz de assegurar um desempenho rápido e preciso em qualquer condição de operação.

Alguns dos aspectos do controle de manipuladores a serem considerados quando da estruturação de uma arquitetura para um supervisor de controle são:

i) O movimento da estrutura mecânica se realiza através de movimentos de rotação e translação de suas juntas, que devem ser controladas simultaneamente e cujo acoplamento dinâmico dificulta o controle independente das mesmas;

ii) O comportamento dinâmico da estrutura articulada, fortemente não linear e dependente das condições operativas, devem ser levadas em consideração na estratégia de controle escolhida, sendo desejável o acesso a elementos do sistema tais como parâmetros de controle, modelos, interface com meio exterior, calibração, etc, levando assim a uma perfeita adequação do robô às suas aplicações;

iii) A trajetória desejada é definida pela posição, velocidade, aceleração e orientação do efetuador, tornando-se necessário então, efetuar transformações de coordenadas com tempos bem definidos e grande complexidade de cálculos.

Levando em conta os diversos problemas acima apresentados e tendo em vista o desenvolvimento dos microprocessadores, cada vez mais versáteis, rápidos e precisos, tornou-se possível a utilização de técnicas avançadas de controle de movimentos de manipuladores através da utilização de estruturas descentralizadas implementadas com microprocessadores. Muitas dessas estruturas atuam de forma hierarquizada, permitindo um processamento independente de cada junta, viabilizando a implementação de estratégias de controle que necessitem de um grande volume de cálculo e/ou grande número de iterações.

2.1.3 Unidade de conversão de energia

A função básica deste elemento é fornecer energia para os acionamentos do manipulador. Pode ter a forma de um amplificador de potência para o caso de utilização de motores elétricos, compressores para acionamentos pneumáticos e reguladores de pressão para acionamentos hidráulicos. Os acionamentos utilizados são:

i) Elétrico

Utiliza-se de elementos que convertem energia elétrica em movimento rotacional ou translacional. São exemplos os motores de corrente contínua e os motores de passo.

ii) Pneumático

Utiliza ar comprimido para a realização de movimentos. São usados quase que exclusivamente em manipuladores pequenos para movimentação de pequenas peças entre lugares fixos, ou como sistema de preensão de objetos.

iii) Hidráulico

Utiliza óleo em alta pressão para realizar movimentos, sendo capaz de fornecer maior potência para um determinado volume. Tem o inconveniente de poder apresentar vazamentos, exige bombas, reguladores de pressão e outros acessórios, embora estes sejam componentes industriais comuns.

Todas as formas de acionamento descritas não são mutuamente exclusivas, mas é conveniente que pelo menos o acionamento básico do manipulador seja uniforme, reservando-se como opção a utilização de uma outra forma de acionamento para finalidades especiais. Por exemplo, pode-se utilizar acionamento pneumático no efetuador independentemente da forma de acionamento utilizada para os demais movimentos.

2.2 MODELAGEM DE ROBÔS

Robôs industriais são manipuladores mecânicos controlados por computadores usados em aplicações industriais. Tipicamente possuem seis juntas, com uma ferramenta acoplada em sua extremidade final, que é normalmente referenciada como efetuador [19].

Quando da necessidade de execução de uma determinada tarefa do robô, que consiste na realização de uma sequência de movimentos, esta é especificada em termos das coordenadas cartesianas de seu efetuador. Porém, tanto o acionamento como o controle das juntas, pressupõe a utilização de coordenadas angulares, pois as variáveis mensuráveis utilizadas nesta operação se relacionam ao controle de deslocamentos e velocidades dessas juntas.

Portanto, a especificação de onde a ferramenta deve atuar se dá em um sistema de coordenadas enquanto que o acionamento e controle das juntas para que este objetivo seja atingido está em outro. Este fato acarreta a necessidade de obtenção de modelos, onde o controle da posição e orientação do efetuador possa se realizar através das variáveis utilizadas nos acionadores das juntas. Estes modelos são, na realidade, um conjunto de transformações de coordenadas entre o efetuador e as diversas juntas, sendo baseados fundamentalmente em duas abordagens distintas, que são a análise geométrica e a análise cinemática.

Assim, como descrito acima, pode-se determinar a posição/orientação do efetuador de um manipulador no interior de seu volume de trabalho através de dois sistemas de coordenadas: coordenadas de juntas e coordenadas de *link* (cartesianas).

Conceitualmente, o modelo geométrico apresenta uma relação matemática estática entre as coordenadas angulares e as coordenadas de *link,* enquanto que o modelo cinemático busca encontrar relações lineares que representem o modelo do robô entre dois pontos determinados. Esta linearização resulta num conjunto de relações, válidas apenas para pequenos deslocamentos, que podem ser representadas matricialmemte e que origina numa matriz denominada Jacobiano, que será posteriormente definida.

2.2.1 Modelo geométrico

O principal objetivo de um manipulador é trabalhar com seu efetuador (ferramenta), ou seja o efetuador é a parte do manipulador que fisicamente se interfaceia com o ambiente. Para executar uma tarefa, é necessário o conhecimento exato da localização do objeto a ser manipulado e qual a orientação que o efetuador deve ter em relação a este objeto.

O modelo geométrico direto consiste numa relação onde se obtém , a partir do conhecimento das variáveis articulares das diversas juntas, a posição/orientação de cada *link*. Consequentemente a descrição da posição/orientação do efetuador em relação ao referencial de base será:

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

onde:

$\underline{\theta} = (\theta_1, \theta_2, ..., \theta_n)$: n x 1 vetor posição angular

$\underline{X} = (x, y, z, \psi, \theta, \phi)$: vetor posição/orientação (efetuador)

Como já descrito anteriormente, as trajetórias realizadas por um robô de seis graus de liberdade podem ser planejadas em relação à cada junta (coordenadas articulares), ou em termos da posição/orientação do efetuador (coordenadas cartesianas). Como os acionadores requerem como sinal de referência uma coordenada angular, quando uma trajetória é planejada em termos de posição e orientação da ferramenta, a mesma necessariamente deverá ser transformada em coordenadas articulares. Para isto, a partir da equação (2.1), pode-se obter uma transformação inversa do tipo:

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

A resolução desta equação, quando encontrada, pode não ser única, originando a necessidade da utilização de algoritmos que escolham uma única solução. Esta escolha ocorre geralmente em função dos limites físicos do robô e configurações geométricas mais estáveis.

A transformação de um sistema de coordenadas para outro, pode ser realizado a partir de matrizes de transformação homogêneas. Uma matriz de transformação homogênea é uma matriz 4 x 4 da forma:

$$T_{i-1,i} = \left[\begin{array}{ccc:c} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c:c} O_{(t)} & p_{(t)} \\ \hdashline 0 & 1 \end{array}\right] \qquad (2.3)$$

onde:

$T_{i-1,i}$ é a matriz de transformação do sistema de coordenadas *i - 1* com relação a *i*.

O é a matriz orientação 3 x 3 dos cossenos diretores do sistema de coordenada i expresso em coordenadas *i - 1*.

Na matriz acima, as componentes que expressam a posição do referencial terminal são descritas pelo vetor posição p(t), enquanto que a orientação é descrita pelos três vetores ortonormais n(t), s(t) e a(t), chamados vetores "normal", "slide" e "approach", respectivamente. Todos estes vetores são definidos em relação ao sistema de coordenadas da base e são mostrados na figura 2.2.

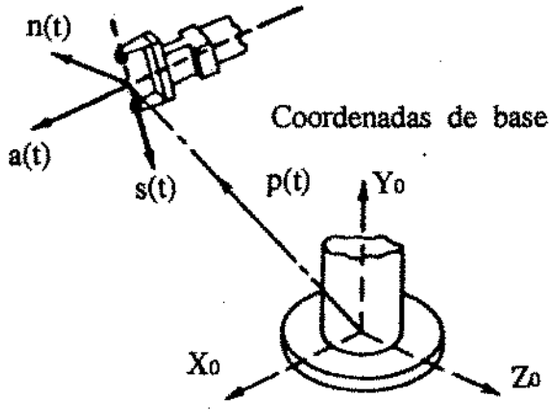


Figura 2.2 Vetor Posição e Orientação do punho

2.2.1.1 Descrição da matriz de orientação

Como a maioria das tarefas realizadas por um manipulador são realizadas pelo seu elemento terminal, torna-se necessário o conhecimento da expressão da posição do mesmo num instante de tempo t em coordenadas cartesianas, expressas em relação à base do manipulador.

Para comodidade dos cálculos, a matriz de orientação poderá ser expressa em termos de três angulos, denominados A, B e C. Esta formulação implica que a posição/orientação do elemento terminal do robô será definida por um vetor de dimensão 6 . x 1 representado por : [p_x, p_y, p_z, A, B, C].

Uma das formas de definição desses ângulos de orientação do elemento terminal do manipulador em relação ao punho são os chamados ângulos de Euler. Estes ângulos podem ser obtidos através de três rotações elementares ψ, θ e ϕ, que são definidas abaixo:

Inicialmente em t=t_0, temos n(t_0), s(t_0) e a(t_0) alinhados com X_0, Y_0 e Z_0, respectivamente, como mostrado na figura 2.2(b). Qualquer orientação [n(t), s(t), a(t)] pode ser obtida por uma rotação de ϕ radianos sobre o eixo Z_0, até s(t_0) alinhar-se com s(t_1). Então, efetua-se uma rotação de θ radianos sobre s(t_1) até que a(t_0), que é igual a a(t_1), alinhe-se com a(t). Finalmente efetua-se uma rotação de ψ radianos sobre a(t) para obter o requerido n(t) e s(t). Isto é equivalente a rodar as coordenadas [n(t_0), s(t_0), a(t_0)], que se alinham com [X_0, Y_0, Z_0]. Originalmente, ψ radianos sobre Z_0, então θ radianos sobre Y_0, e finalmente ϕ radianos sobre Z_0 novamente. A relação é portanto:

$$\text{EULER } (\psi,\theta,\phi) = \text{ROT}(z,\psi,) \; . \; \text{ROT}(y,\theta) \; . \; \text{ROT}(z,\phi) \qquad (2.4)$$

$$\text{Euler } (\psi,\theta,\phi) = \begin{bmatrix} c\psi\, c\phi - s\psi\, c\theta\, s\phi & -c\psi\, s\phi - s\psi\, c\theta\, c\phi & s\psi\, s\theta \\ s\psi\, c\phi + c\psi\, c\theta\, s\phi & -s\psi\, s\phi + c\psi\, c\theta\, c\phi & -c\psi\, s\theta \\ s\theta\, s\phi & s\theta\, c\phi & c\theta \end{bmatrix} \quad (2.5)$$

Consequentemente o estado do elemento terminal do manipulador num determinado tempo *t* no espaço de coordenadas cartesianas com respeito a um referencial solidário à base, pode ser também representado por um vetor de seis dimensões $[p_x, p_y, p_z, \psi, \theta, \phi]$.

É possível também a solução do problema inverso, que é a obtenção dos três ângulos de Euler a partir de uma dada matriz de orientação.

O modelo geométrico direto de um robô fornece um vetor posição e uma matriz de orientação de seu elemento terminal expresso em relação ao sistema de coordenadas de sua base.

Para a obtenção dos três ângulos de Euler (ψ,θ,ϕ) a partir da matriz de orientação, define-se a função ATAN2, que tem por finalidade garantir a unicidade de solução, visto serem muitas as soluções de obtenção de três ângulos que satisfazem uma dada orientação. Através da comparação entre a matriz de Euler (2.5) e a matriz de orientação final (2.3) pode-se obter os três ângulos de Euler, que são expressos pelas seguintes equações:

$$\psi = \text{ATAN2}\left[\frac{a_x}{-a_y}\right] \qquad \phi = \text{ATAN2}\left[\frac{-C\psi.a_x - S\psi.s_y}{C\psi.n_x + S\psi.n_y}\right] \qquad \theta = \text{ATAN2}\left[\frac{S\psi.a_x - C\psi.a_y}{a_z}\right]$$

Equações 2.6 : Ângulos de Euler

A função ATAN2 define os quadrantes em que se situarão os ângulos de Euler, dependendo da relação dos sinais dos componentes da matriz de orientação. Esta função é definida por:

$$\theta = \text{ATAN2}\left[\frac{X}{Y}\right] = \begin{cases} 0 \le \theta \le 90, & \text{COM } +X, +Y \\ 90 \le \theta \le 180, & \text{COM } -X, +Y \\ -180 \le \theta \le -90, & \text{COM } -X, -Y \\ -90 \le \theta \le 0, & \text{COM } +X, -Y \end{cases}$$

Tabela 2.1 : Definição da função ATAN2

2.2.1.2 Parâmetros de Denavit-Hartenberg (D.H.)

A partir do modelo geométrico direto podemos determinar as coordenadas de posição e orientação de cada *link* do robô, e consequentemente a posição/orientação do elemento terminal do mesmo em relação ao sistema de coordenadas de base.

A sistemática estabelecida por Denavit-Hartenberg [5] permite descrever a posição sucessiva das coordenadas de cada *link* através da definição de quatro parâmetros (a, α, d e θ). Estes parâmetros estão definidos na figura 2.3 [22].

Para cada *link* do robô é fixado um sistema de coordenadas composto de três versores ortogonais. Estas coordenadas são chamadas de coordenadas de *link*, e sua posição e orientação são definidas por matrizes de transformadas homogêneas.

a_i = a menor distancia entre Z_i e Z_{i-1}

α_i = angulo entre Z_i e Z_{i-1}

d_i = a menor distancia entre X_i e X_{i-1}

θ_i = angulo entre X_i e X_{i-1}

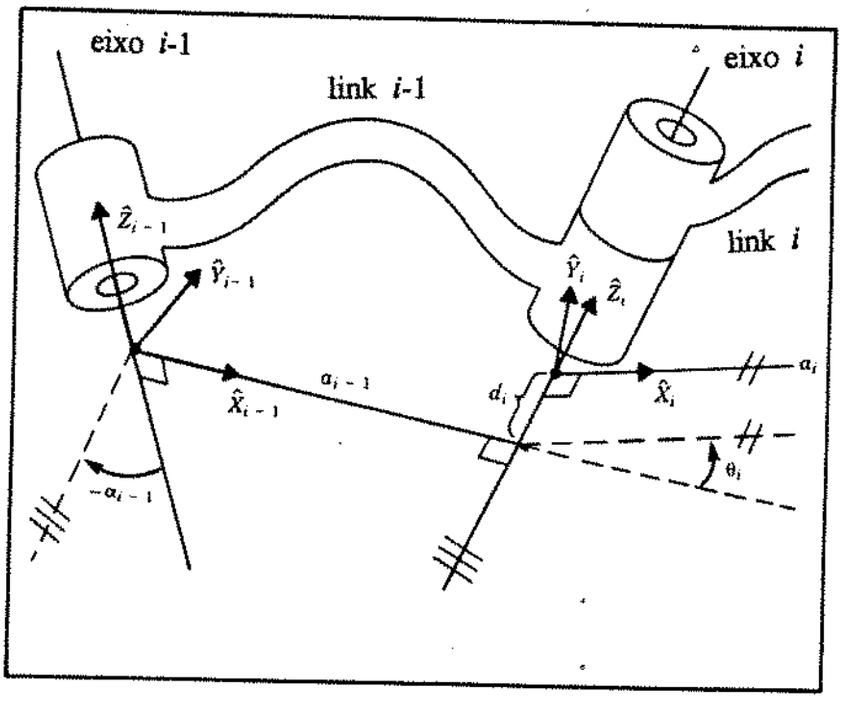


Figura 2.3 Definição dos parâmetros de Denavit-Hartemberg

A definição das variáveis é dependente do tipo de junta empregada na construção do robô. Se a junta *i* é rotacional, então θ_i é a variável de junta e d_i, a_i e α_i são constantes. Se a junta *i* é translacional, então d_i é a variável de junta e θ_i, a_i e α_i são constantes.

As coordenadas de posição e orientação de uma dado *link i* em relação as coordenadas do *link i-1* podem ser descritas a partir da seguinte relação matricial :

$$A_{i-1,i} = \begin{bmatrix} c\theta_i & -s\theta_i\, c\alpha_i & s\theta_i\, s\alpha_i & a_i\, c\theta_i \\ s\theta_i & c\theta_i\, c\alpha_i & -c\theta_i\, s\alpha_i & a_i\, s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.7)$$

Onde os termos "s" e "c" significam seno e cosseno respectivamente. As coordenadas de posição e orientação de um *link* *m* em relação ao sistema de coordenadas fixo à base do robôô são definidas pela equação (2.8) :

$$A_{0,m} = A_{0,1} \cdot A_{1,2} \cdots A_{m-2,m-1} \cdot A_{m-1,m} \qquad (2.8)$$

onde m : m-ésimo grau de liberdade do manipulador.

Portanto, a partir de uma multiplicação de uma sequência de matrizes de transformação de coordenadas, obtém-se as coordenadas de um *link* genérico em relação ao sistema de coordenadas definido na base do robô.

Finalmente, o elemento terminal de manipulador usualmente ainda tem uma ferramenta acoplada em sua extremidade final, fazendo parte de seu efetuador. Esta ferramenta também tem um sistema de coordenadas em referência às coordenadas de *link* e são notadas por FERR_L. A posição e orientação tendo como referência as coordenadas da base do manipulador são notadas por FERR_B. A matriz FERR_L é constante enquanto que FERR_B é uma função de variáveis de junta. A relação entre as duas matrizes é dada pela equação (2.9).

$$FERR\_B = A_{0,m} \cdot FERR\_L \qquad (2.9)$$

2.2.2 Modelo cinemático

A modelagem geométrica modela o robô estaticamente, não levando em consideração fatores como tempo e velocidade. Para aplicações de controle, são necessários modelos onde a relação destes fatores com o comportamento do robô sejam levados em conta.

A modelagem cinemática trata do movimento das juntas independentemente das forças que o causam, referindo-se a todas as propriedades baseadas na geometria e tempo.

Quando se deseja, dado a posição espacial da ferramenta de um robô, se conhecer os valores das suas variáveis de juntas, deve-se estabelecer o chamado modelo inverso do mesmo. A definição do modelo geométrico inverso, leva ao estabelecimento de equações algébricas que acarretam no surgimento de várias possibilidades de solução do sistema, levando a dificuldades para a solução do mesmo.

No modelo cinemático se estabelece uma relação numérica entre as coordenadas cartesianas e articulares, válida somente para pequenos deslocamentos. Esta restrição pode no entanto, ser contornada através do estabelecimento de um processo iterativo, onde um deslocamento total é particionado em um número adequado de deslocamentos "infinitesimais", tornando válida a utilização do mesmo.

Do modelo cinemático, tem-se para pequenos deslocamentos a equação:

$$\Delta x_{nx1} = [J]_{nxm} \quad \Delta\theta_{mx1} \qquad (2.10)$$

onde:
Δx = incremento de posicionamento espacial
$\Delta\theta$ = incremento angular nas diversas juntas
[J] = matriz Jacobiana
n = número de variáveis cartesianas
m = número de graus de liberdade do robô

Dado valores incrementais para os ângulos de cada junta, consegue-se obter o deslocamento espacial de seu efetuador. Este deslocamento é função direta do modelo cinemático do manipulador, que é dado pela matriz Jacobiana. O Anexo A mostra os procedimentos para obtenção desta matriz.

No problema inverso, sendo conhecida a posição/orientação final do efetuador, deseja-se conhecer o incremento angular necessário para controlar cada junta do manipulador, permitindo assim o posicionamento do efetuador num determinado ponto do volume de trabalho do mesmo. Isto pode ser obtido através da relação: .

$$\Delta\theta_{mx1} = [J]^{-1}_{mxn} \quad \Delta x_{nx1} \qquad (2.11)$$

A solução numérica deste problema consiste na aplicação da equação (2.11) iterativamente. Na primeira iteração, é conhecido o vetor Δx, que representa o deslocamento total especificado. Isto gera um vetor $\Delta\theta$, que não é o valor dos deslocamentos angulares finais desejados, pois como explicado anteriormente, o modelo só é válido para pequenos deslocamentos. Este vetor $\Delta\theta$ é usado na equação (2.10) dando origem a um ponto intermediário em coordenadas cartesianas, que gera um novo Δx e assim sucessivamente. Os deslocamentos encontrados vão se tornando menores até a convergência do processo.

Portanto, nos algoritmos para modelagem inversa, trabalha-se com inversão de matrizes e processos iterativos, onde as condições de parada são a posição/orientação desejada ou o número de iterações.

2.2.3 Modelo dinâmico

Até o presente momento, foi suposto que as massas pertencentes a cada junta do manipulador são concentradas no centro de gravidade, de maneira que o robô estará sempre em equilíbrio dinâmico durante o movimento. Consequentemente os efeitos dinâmicos poderão ser desprezados nos algoritmos de controle.

Porém, à medida que a velocidade do movimento aumenta, o modelo geométrico se distancia do modelo real. As forças inerciais, centrífuga e de deslocamento vão aparecer, e de outro lado, os jogos, atritos e elasticidade não poderão ser desprezados. Assim, para grandes velocidades, é necessário levar em consideração os fenômenos dinâmicos.

O modelo dinâmico pode ser obtido através de diferentes procedimentos. O procedimento mais utilizado devido à facilidade de implementação computacional é a partir da descrição do sistema pelas equações de Euler-Lagrange [15]

Quando as equações de Newton-Euler são calculadas para qualquer manipulador, encontra-se a equação dinâmica que pode ser escrita na seguinte forma:

$$\tau = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) \qquad (2.12)$$

onde:
$M(\theta)$ é a matriz de inércia do manipulador;
$V(\theta,\dot{\theta})$ é o vetor das forças centrífugas e de Coriolis;
$G(\theta)$ é o vetor das forças gravitacionais

Portanto, esta equação fornece a expressão do torque em um manipulador em função das posições, velocidades e acelerações das juntas.

Este trabalho não é dedicado à implementação de um modelo dinâmico do robô na estrutura do supervisor de controle proposto. Salientamos que a implementação do mesmo é exaustiva, devido à complexidade das equações e do grande número de não linearidades. Nada impede entretanto que, no futuro o modelo dinâmico venha a ser implementado, embora condições como capacidade de processamento devam antes ser cuidadosamente analisadas antes desta tarefa.

2.3 ALGORITMOS NUMÉRICOS PARA MODELAGEM CINEMÁTICA INVERSA

Na escolha de um determinado algoritmo numérico para a implementação do modelo cinemático inverso, deve-se conhecer *a priori* o tipo de aplicação para o qual o mesmo será utilizado. Podemos então encontrar algoritmos apropriados para algumas aplicações, que não sejam apropriados para outras e vice-versa. A seguir descreve-se algumas das aplicações em que tais algoritmos poderão ser utilizados e as características desejadas do algoritmo.

Para o problema de identificação de parâmetros de um manipulador, é necessário que o elemento terminal atinja a posição desejada, não importando os pontos intermediários obtidos. Para isto pode-se usar algoritmos de rápida convergência e que não levem em consideração a evolução dos movimentos das juntas até essa posição.

Para a realização de um controlador de posição no espaço cartesiano, utilizado frequentemente em operações de soldagem, usinagem e outros, é interessante um algoritmo que gere uma trajetória *bem comportada*, contendo um determinado número de pontos intermediários e que otimize os deslocamentos das juntas de maneira a diminuir o erro de posição do elemento terminal do manipulador.

Este tipo de aplicação pode ser melhor visualizado na figura 2.4, onde mostra-se uma segunda malha de controle atuando no sistema. Através de um sensor de força/torque localizado no elemento terminal do manipulador pode-se obter sinais que são proporcionais ao posicionamento espacial do mesmo. Estes sinais são então digitalizados através de um conversor A/D e com a utilização do modelo cinemático inverso são obtidos valores de correção dos ângulos que deverão ser enviados para as unidades de controle de juntas.

O supervisor de controle em desenvolvimento prevê o suporte do uso dos algoritmos para modelagem cinemática inversa para objetivos de controle de posição no espaço carteziano, uma vez que foram projetadas interfaces para esta função.

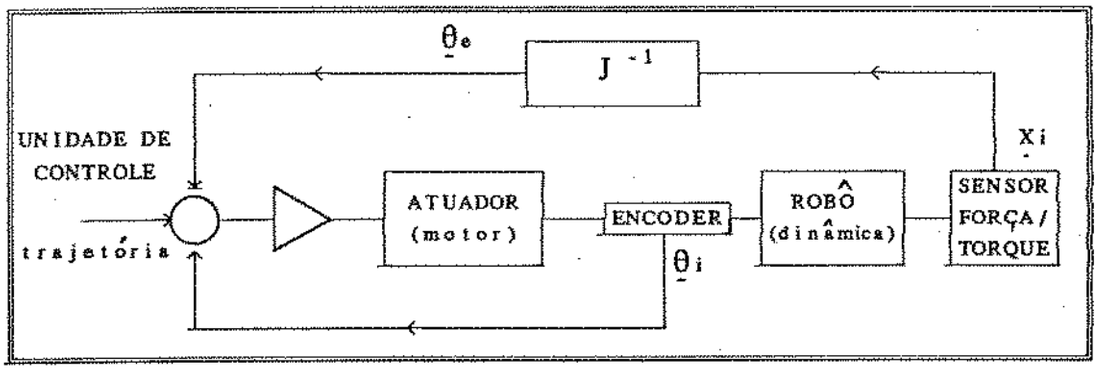


Figura 2.4: Controle de esforços - Diagrama de blocos

2.4 CONCLUSÕES

Neste capítulo descreveu-se os principais componentes de um robô manipulador. Foram introduzidos os conceitos de modelagem geométrica, modelagem dinâmica e modelagem cinemática, sendo abordados alguns aspectos da modelagem cinemática inversa e discutidas as aplicações onde alguns tipos de algoritmos numéricos podem ser úteis.

No próximo capítulo é apresentado o modelo do robô industrial utilizado para a implementação e testes dos algoritmos para a modelagem cinemática inversa.

CAPÍTULO 3 : MODELAGEM E CONTROLE DE UM MANIPULADOR INDUSTRIAL

No capítulo 2 abordou-se a modelagem de manipuladores de uma maneira geral, com os diversos tipos de modelos utilizados e suas aplicações.

Uma vez especificado as ferramentas básicas usadas na modelagem de manipuladores, este capítulo descreve especificamente os algoritmos para modelagem geométrica e cinemática inversa de um robô industrial. Não é objetivo deste trabalho a abordagem dos algoritmos de controle de junta, visto ser este assunto objeto de interesse de outra tese já defendida na Faculdade de Engenharia Mecânica da UNICAMP [6], sendo que os algoritmos estudados e testados no citado trabalho são aproveitados integralmente para uso no sistema em desenvolvimento. A ênfase é dada para os algoritmos de modelagem cinemática inversa, onde analisaremos os melhores métodos a serem utilizados nas diversas aplicações, tendo em vista especialmente o controle no espaço cartesiano.

Para o desenvolvimento dos algoritmos apresentados neste capítulo, foi utilizado um modelo do robô industrial MANUTEC r-3, de seis graus de liberdade. A seguir são apresentados as características deste robô e posteriormente descreve-se os resultados obtidos dos algoritmos implementados.

3.1 MODELO DO ROBÔ UTILIZADO

O modelo de robô utilizado é o MANUTEC r-3, que possui seis juntas rotacionais. Os parâmetros de Denavit e Hartenberg desse robô são apresentados na tabela 3.1.

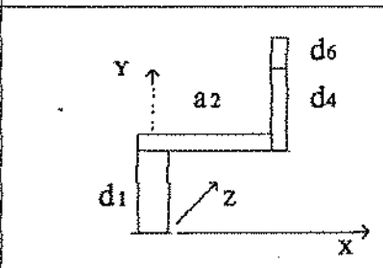
JUNTA	θ (deg)	d (mm)	α (deg)	a (mm)	REPRESENTAÇÃO
1	θ_1	665.0	-90.0	0.0	
2	θ_2	0.0	0.0	500.0	
3	θ_3	0.0	90.0	0.0	
4	θ_4	830.0	-90.0	0.0	
5	θ_5	0.0	90.0	0.0	
6	θ_6	99.5	0.0	0.0	

Tabela 3.1 : Parâmetros de D. H. para o robô MANUTEC r-3

Utilizando os valores acima apresentados na relação 2.7 , obtém-se as matrizes $A_{i-1,i}$ descritas na tabela 3.2. Esta tabela apresenta um conjunto de matrizes, onde cada uma representa uma transformação de coordenadas de um *link i* em relação ao *link i*-1 a que este está ligado.

$$A_{0,1}\begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_{1,2}\begin{bmatrix} C_2 & -S_2 & 0 & a_2.C_2 \\ S_2 & C_2 & 0 & a_2.S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{2,3}\begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_{3,4}\begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{4,5}\begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_{5,6}\begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

onde C_i e S_i significam cosseno(θ_i) e seno (θ_i), respectivamente.

Tabela 3.2 : Matrizes $A_{i-1,\ i}$ para o robô MANUTEC r-3

A transformação de coordenadas entre o referencial do último *link* (efetuador) e o sistema de referência fixo do robô é dada pela seguinte relação:

$T_6\ (\theta_1,\theta_2,\ldots,\theta_6) =$

$$A_{0,6} = A_{0,1}.\ A_{1,2}.\ A_{2,3}.\ A_{3,4}.\ A_{4,5}.\ A_{5,6} \qquad (3.1)$$

Esta relação pode ser expressa na forma matricial mostrada abaixo:

$$= \left[\begin{array}{c:c} \underline{n}\ \underline{s}\ \underline{a} & \underline{p} \\ \hdashline 0 & 1 \end{array}\right] \qquad (3.2)$$

onde:

a matriz [$\underline{n}$ $\underline{s}$ $\underline{a}$] representa a orientação e o vetor $\underline{p}$ a posição do efetuador no interior do volume de trabalho.

A operação de multiplicação dessas matrizes fornece as seguintes expressões para os vetores de posição e orientação do efetuador do robô em relação às coordenadas do referencial da base:

• Expressões do vetor Posição:

$$\begin{aligned} p_x &= \{(C_4S_5d_6).C_{23} + (C_5d_6 + d_4).S_{23} + a_2C_2\}.C_1 - \{S_4S_5d_6\}.S_1 \\ p_y &= \{(C_4S_5d_6).C_{23} + (C_5d_6 + d_4).S_{23} + a_2C_2\}.S_1 + \{S_4S_5d_6\}.C_1 \\ p_z &= -(C_4S_5d_6).S_{23} + (C_5d_6 + d_4).C_{23} - a_2S_2 + d_1 \end{aligned} \qquad (3.3)$$

• Expressões do vetor Orientação :

$$\begin{aligned} n_x &= \{(C_1C_{23}C_4 - S_1S_4).C_5 - C_1S_{23}S_5\}.C_6 + \{-C_1C_{23}S_4 - S_1C_4\}.S_6 \\ n_y &= \{(S_1C_{23}C_4 + C_1S_4).C_5 - S_1S_{23}S_5\}.C_6 + \{-S_1C_{23}S_4 + C_1C_4\}.S_6 \\ n_z &= \{-S_{23}C_4C_5 - C_{23}S_5\}.C_6 + S_{23}S_4S_6 \end{aligned}$$

$$\begin{aligned} s_x &= -\{(C_1C_{23}C_4 - S_1S_4).C_5 - C_1S_{23}S_5\}.S_6 + \{-C_1C_{23}S_4 - S_1C_4\}.C_6 \\ s_y &= -\{(S_1C_{23}C_4 + C_1S_4).C_5 - S_1S_{23}S_5\}.S_6 + \{-S_1C_{23}S_4 + C_1C_4\}.C_6 \\ s_z &= \{S_{23}C_4C_5 + C_{23}S_5\}.S_6 + S_{23}S_4C_6 \end{aligned} \qquad (3.4)$$

$$\begin{aligned} a_x &= (C_1C_{23}C_4 - S_1S_4).S_5 + C_1S_{23}C_5 \\ a_y &= (S_1C_{23}C_4 + C_1S_4).S_5 + S_1S_{23}C_5 \\ a_z &= -S_{23}C_4S_5 + C_{23}C_5 \end{aligned}$$

A partir dessas expressões pode-se também escrever a orientação em função dos ângulos de Euler, cujas relações com os vetores $\hat{n}$, $\hat{s}$ e $\hat{a}$ são descritas nas equações 2.6. Portanto uma outra possível forma de representação seria:

$$T_6\ (\theta_1,\theta_2,\ldots,\theta_6) = (p_x,p_y,p_z,\psi,\theta,\phi)$$

onde:
ψ, θ e ϕ são ângulos de Euler;
p_x, p_y e p_z são componentes cartesianas da posição do efetuador.

3.2 ALGORITMOS DE CONTROLE DE JUNTAS

Conforme exposto anteriormente, o nível 1 da hierarquia proposta trata as malhas clássicas de controle através da implementação de algoritmos de controle de juntas. A estrutura apresentada multiplica-se pelo número de juntas, bastando portanto a caracterização de apenas uma para que a análise seja estendida às demais.

No modelo clássico de controle de um robô temos para cada grau de liberdade a seguinte situação:

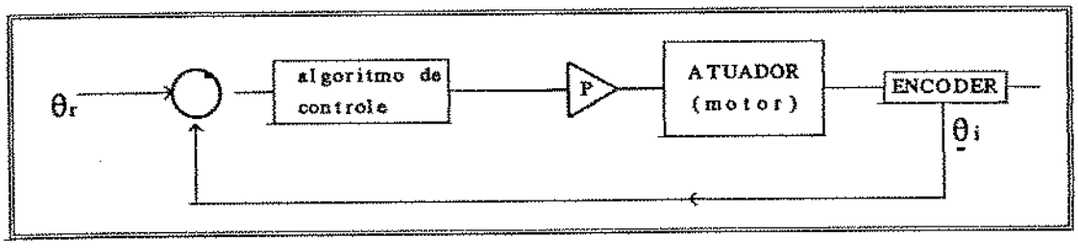


Figura 3.1 : Estrutura de Controle de nível 1

onde:
θ_r = ângulo de referência para posição desejada;
θ_i = ângulo efetivamente obtido;
P = amplificador de potência.

Os algoritmos apropriados para a realização do controle ficam residentes no sistema em memória não volátil (EPROM). Isto torna viável a substituição de um tipo de algoritmo por outro através da troca de componentes e, quando necessário, da lógica de excitação do amplificador de potência.

3.3 ALGORITMOS PARA MODELAGEM CINEMÁTICA INVERSA

Como já exposto no capítulo 2, a modelagem cinemática se preocupa com fatores como velocidade e aceleração das juntas, permitindo assim a obtenção de trajetórias efetuadas por cada uma num determinado movimento. Este capítulo aborda os algoritmos utilizados para a modelagem cinemática inversa.

Estes algoritmos têm por finalidade a geração de uma trajetória a ser executada pelo efetuador do robô, e a obtenção das coordenadas articulares do ponto final desejado, dado que se conhece as coordenadas cartesianas e articulares do ponto inicial, e as coordenadas cartesianas do ponto final.

Estes métodos se baseiam fundamentalmente em técnicas de inversão da matriz Jacobiana, que é a matriz que relaciona deslocamentos angulares com os deslocamentos em coordenadas cartesianas. O *corpo* dos algoritmos praticamente não tem diferenças, mantendo as características já apresentadas no capítulo 2.

No tratamento desta matriz para sua inversão, uma série de fatores devem ser levados em conta por esses métodos. Estes fatores são tanto de ordem matemática como de ordem física, tais como a necessidade de se evitar matrizes singulares (não inversíveis), a otimização dos movimentos das juntas, a normalização (balanceamento) dos elementos das matrizes para a eliminação dos problemas de precisão numérica, etc.

Para a escolha do algoritmo que mais se adapte à aplicação de controle no espaço cartesiano, que é a aplicação a ser utilizada no Supervisor de Controle, foram estudados e implementados três métodos de inversão, que são abordados no presente trabalho. O primeiro é o algoritmo de Moore-Penrose [7], o segundo é o algoritmo de Miss [8] e por fim o algoritmo de Gauss [9].

3.3.1 Método de Moore-Penrose

3.3.1.1 Princípio

Este algoritmo fornece um método iterativo para obtenção de uma matriz inversa generalizada. Apresenta rápida convergência, de modo que poucos pontos são obtidos intermediariamente no movimento desejado. Uma das aplicações desse tipo de algoritmo seria no momento que se está interessado somente na determinação da posição final do efetuador, por exemplo durante uma fase de aprendizagem, onde o comportamento da trajetória não é importante.

3.3.1.2 Descrição do algoritmo

A inversão iterativa se processa do seguinte modo:

• PASSO 1 : Define-se A, que no caso é uma matriz Jacobiana, como sendo uma matriz m × n com elementos reais:

$A = [a_1\ a_2\ a_3\ \ldots\ a_n]$

onde:

a_i denota suas n colunas, cada uma de tamanho m x 1.

• PASSO 2 : Define-se $A_i = [a_1 \; a_2 \; \ldots \; a_i]$ como a matriz onde a coluna i foi adicionada, começando com $A_1 = a_1$. Então

$A_i = [A_{i-1} \; a_i]$, onde $i = 2,3, \ldots ,n$

• PASSO 3 : O seguinte teorema se aplica então:

Seja A_{i-1} uma matriz m x (i-1) e seja a_i um vetor m x 1. Então a matriz de inversão generalizada de Moore-Penrose é definida como:

$$A_i^+ = \frac{A_{i-1}^+ - d_i b_i}{b_i^T} \qquad (3.5)$$

onde:

$$d_i = A_{i-1}^+ a_i \quad \text{e} \quad b_i = \begin{cases} \dfrac{1}{c_i^T \cdot c_i} \cdot c_i & \text{p/ } c_i \neq 0 \\ \dfrac{1}{1 + d_i^T d_i} \cdot d_i^T \cdot A_{i-1}^+ & \text{p/ } c_i = 0 \end{cases} \qquad (3.6)$$

onde $c_i = a_i - A_{i-1} \cdot d_i$

e "T" e "+" denotam a transposta e a inversa generalizada de Moore-Penrose respectivamente.

Para começar o algoritmo recursivo, conforme fluxograma apresentado na figura 3.2, utiliza-se o seguinte critério:

$$A_1^+ = \begin{cases} a_1^T & p/a_1 \neq 0 \\ \dfrac{1}{a_1^T \cdot a_1} \cdot a_1 & p/\ a_1 = 0 \end{cases} \qquad (3.7)$$

Um exemplo de aplicação deste método é mostrado no Anexo D, onde encontra-se a inversa generalizada de uma matriz que não possui a sua inversa convencional.

Este método foi utilizado para a inversão da matriz Jacobiana do modelo cinemático do robô. Os resultados obtidos mostram uma rápida convergência, com o objetivo final sendo encontrado em poucas iterações. Sua maior deficiência reside no fato desta inversão não levar em conta as características mecânicas do sistema, ocasionando movimentos bruscos das juntas. Porém, em aplicações onde não seja importante a trajetória realizada, este pode vir a ser um método útil e eficiente.

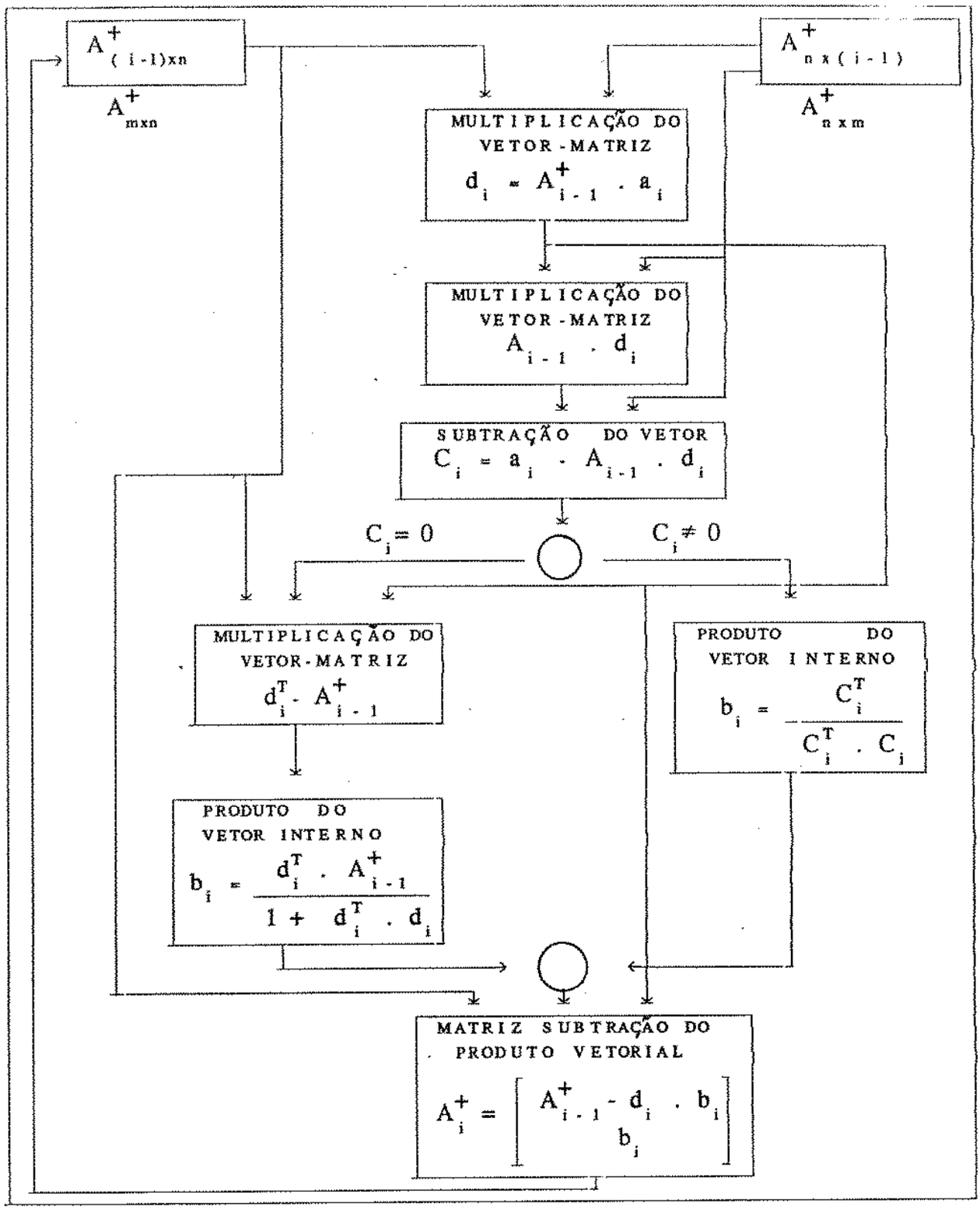


Figura 3.2 : Fluxograma do método de Moore-Penrose

3.3.2 Método de MISS

3.3.2.1 Princípio

Este método é baseado no princípio de se modificar uma variável de junta em cada passo de cálculo. O cálculo do valor da modificação é encontrado aplicando-se critérios de minimização na somatória do erro de posição das diversas juntas do manipulador [8].

3.3.2.2 Descrição do algoritmo

Como já descrito anteriormente, a expressão que relaciona os deslocamentos em coordenadas cartesianas com os deslocamentos em coordenadas articulares é dada abaixo.

$$\Delta x = J \cdot \Delta\theta \qquad (3.8)$$

O valor do erro oriundo da suposição que a equação acima é válida para qualquer deslocamento de uma determinada junta pode ser expresso por:

$$\varepsilon_i \equiv J_{xi} \cdot \Delta\theta - \Delta x_i \qquad (3.9)$$

onde:
i representa a junta;
J_{xi} são os elementos da linha i da matriz Jacobiana.

Podemos então definir uma função S que representa o erro total das juntas num determinado movimento, que é a somatória dos erros das diversas juntas ao quadrado (critério dos mínimos quadrados):

$$S \equiv \sum_{i} \varepsilon_i^2 = \sum_{i} \left[w_{ij} \cdot \Delta\theta_j - \Delta x_i \right]^2 \qquad (3.10)$$

onde w = elemento da matriz Jacobiana J

O objetivo é minimizar o erro de cada junta a partir desta equação. O quadrado da equação 3.10 é uma expressão polinomial que deve ser diferenciada para se obter o incremento ótimo das variáveis de juntas.

$$\frac{\delta S}{\delta d\Delta\theta_j} = \sum_{i=1,m} (2\omega_{ij}^2 \cdot d\Delta\theta_j - 2\Delta x_i \cdot \omega_{ij}) = 0 \qquad (3.11)$$

Isolando o termo do deslocamento angular temos então:

$$d\Delta\theta_j = \frac{\sum_{i=1,m} \Delta x_i \cdot \omega_{ij}}{\sum_{i=1,m} \omega_{ij}^2} \qquad (3.12)$$

O valor encontrado de $d\Delta\theta_j$ é o incremento da junta *j* numa determinada iteração do algoritmo.

Nesta versão do algorítmo de Miss, a qualidade dos resultados são um pouco comprometidos pelo fato de que o valor do incremento das juntas não representa o mínimo absoluto indicado pelo princípio dos mínimos quadrados.

Os resultados de simulações efetuadas com o método de Miss são mostradas no ítem de comparação entre os diversos métodos.

Uma otimização do algorítmo poderá ser realizada a partir de uma modificação apropriada nas condições de contorno, que permitirá que a condição de mínimo absolúto seja satisfeita.

Estas condições advém do fato de que uma função V de n variáveis ρ define uma superfície no espaço dimensional (n + 1) [8]. A distância mais curta entre dois pontos de uma superfície é singularmente definida pelo seu gradiente, o qual é igual à soma de suas derivadas parciais $\delta V/\delta\rho_j$ multiplicadas pelos vetores unidades k_j , isto é:

$$\text{grad } V = \sum_{j=1,n} k_j \cdot \delta V/d\rho_j \qquad (3.13)$$

No caso das equações cinemáticas usadas para definir o novo algoritmo, a função S do vetor ponderação Δx é uma função de n variáveis de juntas Δq. O ponto inicial e o final requeridos, correspondem respectivamente ao valor do vetor Δx e do vetor nulo final. Se o quadrado do vetor requerido de variáveis de juntas deve assumir seu mínimo valor, o vetor Δq deve ser ajustado em cada iteração por um incremento, que é proporcional ao gradiente da função S.

As derivadas parciais são multiplicadas por uma constante proporcional τ, que representa um múltiplo do correspondente vetor unidade. O ajuste do vetor ponderação Δx e do vetor de variáveis $\Delta\theta$, pela quantidade definida pelo seu gradiente e uma constante proporcional, acaba cada passo de cálculo. A iteração é continuada até o vetor ponderação ser minimizado a zero.

$$d\Delta\theta_j = \tau \cdot \frac{\delta S}{\delta d\Delta\theta_j} = \tau \cdot \sum_{i=1,m} (-2 \cdot \Delta x_i \cdot \omega_{ij}) \qquad (3.14)$$

Um aspecto a ser considerado é que o algoritmo definido por este procedimento somente desenvolve a convergência requerida, se a constante proporcional τ é ajustada em cada iteração por um valor particular, que corresponde a um valor ótimo para redução do vetor de ponderação.

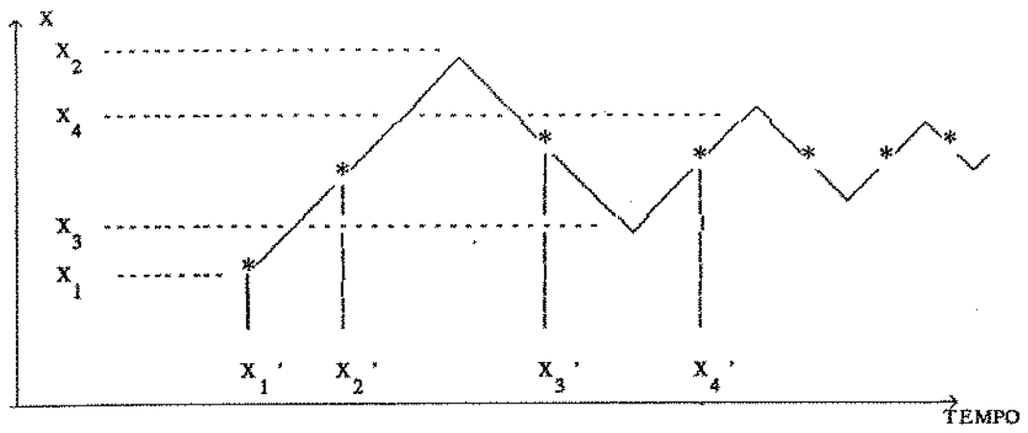
A equação utilizada para o cálculo de τ é :

$$\tau = \frac{\sum_{i=1,m} \left(\Delta x_i \cdot \sum_{j=1,n} (\omega_{ij} \cdot \delta S / \delta d\Delta\theta_j) \right)}{\sum_{i=1,m} \left[\left(\sum_{j=1,n} (\omega_{ij} \cdot \delta S / \delta d\Delta\theta_j^2) \right) \right]} \qquad (3.15)$$

3.3.2.3 Suavização de trajetórias

As trajetórias obtidas pelos métodos descritos anteriormente podem apresentar oscilações dos valores angulares para cada junta. Com o intuito de minimizar este problema, tornando a trajetória a ser enviada para cada junta a mais suave possível, implementou-se o seguinte artifício:

Suponha que a trajetória a ser enviada para a junta apresente uma determinada função (figura 3.3). Obtido o perfil da trajetória, o método consiste em manter o primeiro ponto da mesma, e para os demais pontos calcular-se o valor médio da variação com o anterior, de modo a se evitar uma oscilação desnecessária de juntas.



$$X_1' = X_1$$

$$X_n' = X_n - \frac{X_n - X_{n-1}}{2}$$

Figura 3.3 : Cálculo de pontos médios de uma trajetória

3.3.3 Método de Gauss

3.3.3.1 Princípio

Um outro método analisado foi o método de Gauss. Este método se baseia na construção de uma matriz inversa à matriz Jacobiano, de tal forma a transformá-la na matriz identidade e isolar o termo $\Delta\theta$ como na equação 3.8.

Para sermos precisos, este algoritmo não fornece diretamente a matriz inversa da Jacobiana, embora a inversa possa ser obtida pela sua aplicação.

3.3.3.2 Descrição do algoritmo

Este método irá resolver a equação matricial mostrada abaixo para x, dado qualquer y compatível [10].

$$y_{mx1} = A_{mxn} \cdot x_{nx1} \tag{3.16}$$

Genericamente, temos o seguinte conjunto de equações sendo representado na equação 3.16:

$$\begin{aligned} y_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ y_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ &\vdots \qquad\qquad \vdots \\ y_m &= a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{aligned} \tag{3.17}$$

Considere o conjunto de equações 3.17. Se a_{11} é igual a zero, então deve haver alguma equação no qual o coeficiente de x_1 é diferente de zero. Pode-se então deslocar a equação para a primeira posição e podemos assumir que a_{11} é diferente de zero sem perda de generalidade.

Usamos então a primeira equação para eliminar x_1 de todas as outras equações, de modo a obtermos o seguinte conjunto de equações:

$$\begin{aligned} y_1 &= a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n \\ y_2' &= 0 \quad b_{22}x_2 + b_{23}x_3 + \ldots + b_{2n}x_n \\ &\vdots \qquad \vdots \qquad\qquad \vdots \\ y_m' &= 0 \quad b_{m2}x_2 + b_{m3}x_3 + \ldots + b_{mn}x_n \end{aligned} \tag{3.18}$$

Onde b_{xx} são os novos coeficientes encontrados e y_i' são dados por:

$$y_i' = y_i - y_1 / a_{11} \tag{3.19}$$

Podemos repetir o processo para todas as colunas, até obtermos, em termos matriciais, a seguinte equação:

$$\begin{bmatrix} y_1 \\ y_2' \\ \vdots \\ y_{m(n-1)} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots a_{1n} \\ 0 & b_{22} & b_{23} & \ldots b_{2n} \\ & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & f_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad (3.20)$$

Na expressão 3.20 os coeficientes da matriz e os coeficientes do vetor **y** são conhecidos. Além disso, os coeficientes da diagonal principal são diferentes de zero.

A matriz desta forma, é chamada de matriz triangular superior e toda matriz pode ser colocada nesta forma genérica. A expressão 3.20 nos permite resolver, a partir da última linha, o sistema para **x**, dado qualquer **y**.

Além do processo de triangularização da matriz Jacobiana, é realizado o pivoteamento dessa matriz, que consiste em escolher elementos em ordem decrescente de grandeza para a diagonal principal, de modo a encontrar a melhor relação entre os vetores **x** e **y**.

É importante destacar que se a matriz Jacobiana não for singular, poderão ocorrer problemas na obtenção da solução desejada. Ao mesmo tempo esta matriz deve ser normalizada, pois pequenos erros nos coeficientes da matriz Jacobiana resultarão em grandes erros na solução final.

O método de Gauss foi implementado e foram simulados movimentos em várias situações, sendo verificado uma convergência com um número de iterações que se situa num patamar intermediário entre os algoritmos de Moore-Penrose e de Miss. Os resultados obtidos são apresentados no ítem de comparação de resultados.

3.3.4 Comparações entre os diversos métodos

Uma diagrama mostrando os métodos implementados neste trabalho é apresentado na figura 3.4.

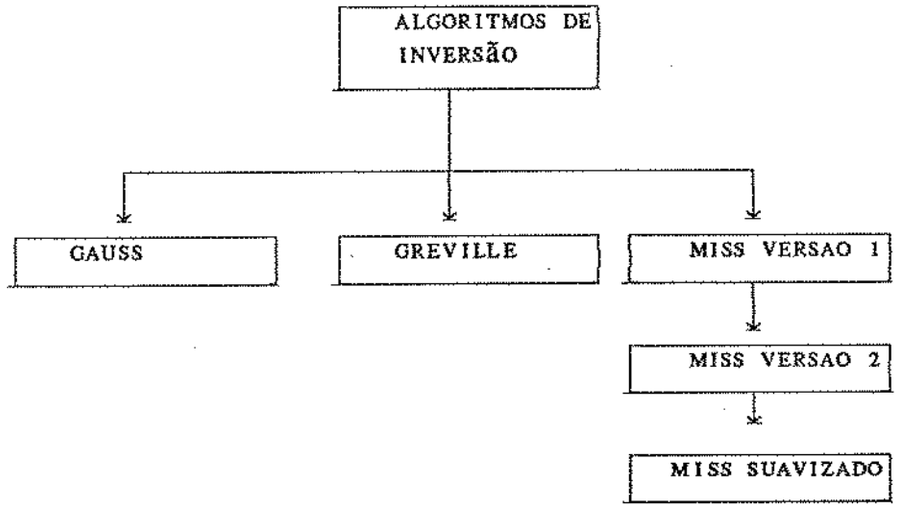


Figura 3.4 : Algoritmos de inversão estudados

No início de nosso trabalho, para detectarmos de maneira rápida as características de cada algoritmo, estes foram implementados e testados num modelo de robô de apenas dois graus de liberdade. Neste exercício, pudemos constatar que o método de Moore- Penrose diferia dos demais, por apresentar um número de interações para convergência extremamente pequeno (no máximo 5).

Como nossos propósitos estão voltados para controle no espaço cartesiano, estas características fizeram com que este método não fosse implementado no robô MANUTEC r-3, pois este certamente não seria o método escolhido para implementação no supervisor de controle em desenvolvimento. Portanto, após esta primeira fase do trabalho, partiu-se para a implementação dos algoritmos de Miss e Gauss no robô de seis graus de liberdade.

Como resultado destes trabalhos, pode-se verificar duas grandes linhas nestes algoritmos de inversão:

• Para aplicações onde o importante é a convergência rápida e não a trajetória, isto é, em aplicações de calibração do robô ou semelhantes, o algoritmo de Moore-Penrose e de certa forma o algoritmo de Gauss mostram-se adequados, pois com poucas iterações a convergência é obtida, apresentando poucos pontos intermediarios, porém com grandes oscilações na posição dos mesmos.

• No caso de geração de trajetórias, onde é importante a obtenção de vários pontos intermediarios e que a suavidade da variação dos mesmos é requerida, o algorítmo de Miss é o que apresenta maior suavidade nas trajetórias, se mostrando portanto mais adequado. As três variações dos algoritmos de Miss implementados mostram uma evolução na suavidade das curvas das trajetórias obtidas.

A seguir mostra-se o trabalho efetuado na implementação e comparação dos métodos de Gauss e Miss no modelo do robô MANUTEC r-3.

3.3.4.1 Comparação dos métodos de Gauss e Miss

Para a comparação dos dois métodos, foi utilizado um pacote computacional desenvolvido em linguagem ADA que implementa o modelo do robô MANUTEC r-3. Utilizando-se do algoritmo de Gauss e da primeira versão do algoritmo de Miss, diversas trajetórias para a ferramenta foram determinadas. Obtinha-se então um arquivo de pontos para cada grau de liberdade do robô. Com estes arquivos, foi possível traçar a trajetória de cada junta e posteriormente se fazer a comparação entre os dois métodos.

O seguinte procedimento foi seguido para a obtenção destes arquivos de pontos para os dois algoritmos:

1) Posição inicial do robô.

São fornecidos os ângulos em graus para as seis juntas do robô MANUTEC r-3. Estes ângulos correspondem à posição que o robô se encontra em relação ao volume de trabalho. No nosso caso utilizou-se a configuração apresentada na tabela 3.3.

CONFIGURAÇÃO INICIAL											
POSICAO (mm)			ORIENT. (graus)			coordenadas articulares					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
500	0	1495	0	0	0	0	90	-90	0	0	0

Tabela 3.3 : Posição/orientação inicial do robô MANUTEC r-3

Onde θ_1, θ_2 e θ_3 correspondem aos graus de liberdade para posicionamento do robô num volume de trabalho, e θ_4, θ_5 e θ_6 fornecem a orientação da ferramenta.

De acordo com o modelo geométrico, as coordenadas cartesianas apresentadas na tabela 3.3 são obtidos a partir das coordenadas articulares, utilizando-se as equações apresentadas na seção 3.1 deste capítulo.

2) Posição desejada

Após a posição inicial, é fornecida a posição/orientação final desejada. A posição é dada em coordenadas cartesianas e a orientação em ângulos de Euler (ψ, θ, ϕ).

3) Número de iterações

O último parâmetro fornecido é o número de iterações. Isto garante que o programa não entre em *loop* caso a posição final desejada seja impossível de se atingir (como por exemplo quando o ponto final especificado está fora da área de trabalho do robô).

3.3.4.2 Resultados obtidos

Dois tipos de informações foram obtidos quando da realização dos exemplos. Estas informações foram *plotadas* em gráficos que mostram duas características importantes que são:

• Gráfico de posição angular

Nestes gráficos são representados a posição angular da junta a cada iteração, fornecendo o comportamento da trajetória durante o movimento. A análise desta trajetória é um fator importante na escolha de um algoritmo de controle de posição, pois é desejável um comportamento suave da mesma, apresentando pouca oscilação e ausência de movimentos bruscos.

• Gráfico de erro de coordenada cartesiana

O valor do erro em uma coordenada a cada iteração é a diferença entre o valor absoluto naquele ponto e o seu valor final especificado. Neste tipo de gráfico busca-se visualizar não o comportamento da trajetória, mas a convergência do processo, isto é, a rapidez com que o algoritmo consegue encontrar a posição final especificada.

Juntamente com os gráficos, é apresentada uma tabela relacionando a configuração final do robô, tanto em coordenadas cartesianas como em coordenadas articulares. A configuração inicial para cada exemplo é única, sendo apresentada na tabela 3.3. Na tabela abaixo relaciona-se os exemplos realizados com os números dos gráficos apresentados. Outra informação mostrada é o número de iterações necessário para a convergência do processo.

Exemplo	Método	N. de iterações	Gráfico
1	Gauss	14	1, 4, 5, 6
1	Miss	45*	1, 4, 5, 6
2	Gauss	20	2
2	Miss	45*	2
3	Gauss	23	3
3	Miss	45*	3

Tabela 3.4: Relacionamento dos gráficos com exemplos

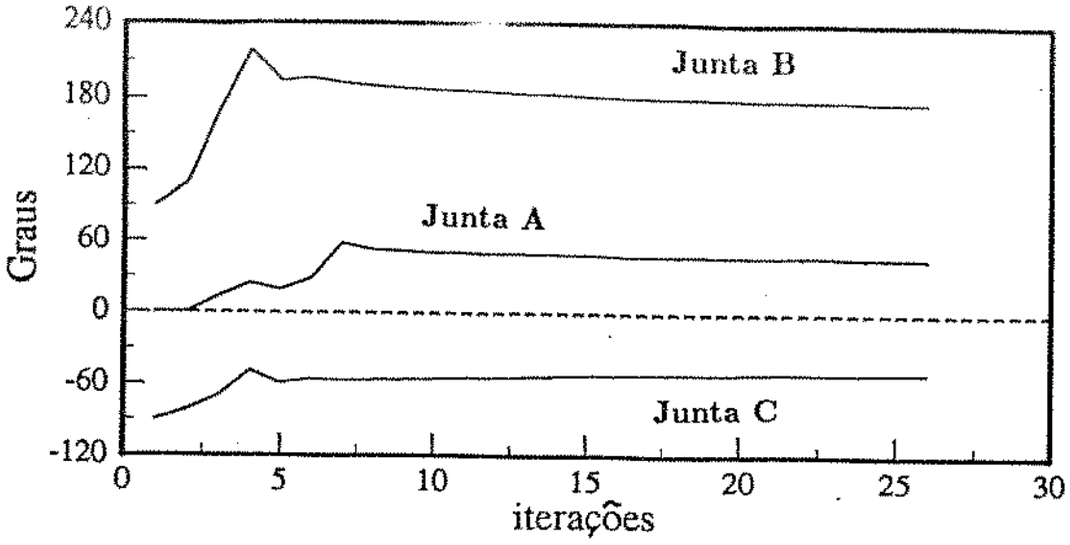
* No caso do método de Miss, alguns problemas foram encontrados no que concerne à orientação da ferramenta, de modo que o número de iterações para a convergência total não pode ser precisamente determinado. Para os graus de liberdade relacionados à posição, o número de iterações para a convergência da trajetória é menor que 45, porém devido aos problemas com a orientação do efetuador, foram feitas mais iterações na busca do resultado esperado.

3.3.4.3 Análise dos resultados obtidos

A análise dos dois métodos testados foi efetuada através da comparação gráfica das trajetórias obtidas, examinando-se as características descritas anteriormente.

A seguir apresenta-se os gráficos obtidos quando da realização dos exemplos.

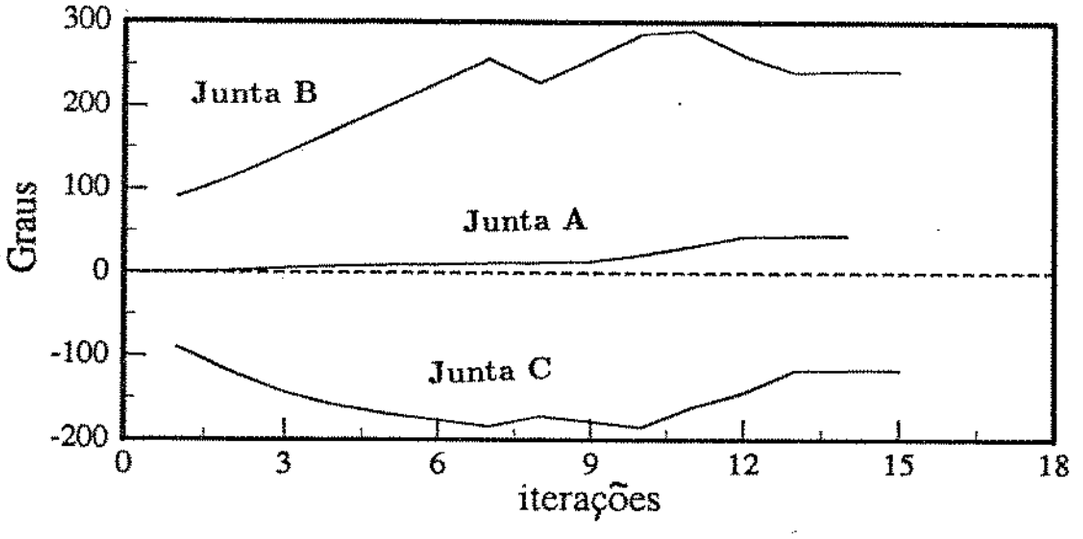
Os gráficos 1, 2 e 3 representam as trajetórias obtidas para os exemplos de mesmo número indicados na tabela 3.4. Nestes gráficos são traçadas as curvas obtidas com o método de Miss (a) e as obtidas com o método de Gauss (b). Nos gráficos 4, 5 e 6 são traçados os erros para cada coordenada cartesiana durante a realização do exemplo de número 1, sendo apresentado no mesmo gráfico os erros dos métodos de Miss e Gauss.



CONFIGURAÇÃO FINAL											
POSICAO (mm)			ORIENT. (graus)			coord. articulares (graus)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
100	100	100	0	0	0	47	175	-48	-9	379	-98

Exemplo 1 : Posição - Miss

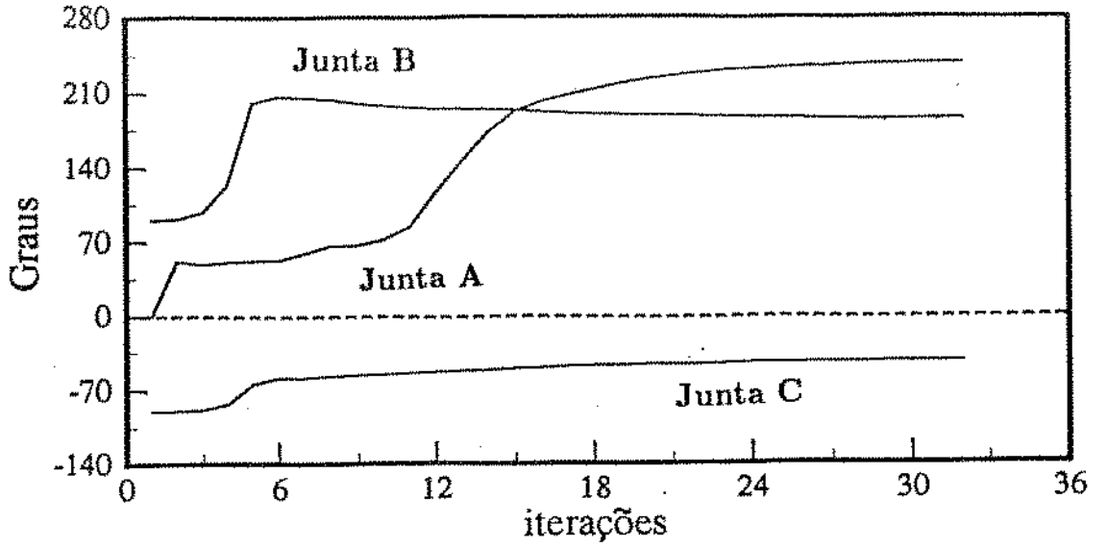
Gráfico 1.a



CONFIGURAÇÃO FINAL											
POSICAO (mm)			ORIENT. (graus)			coord. articulares (graus)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
100	100	100	0	0	0	45	243	-116	0	-127	-45

Exemplo 1 : Posição - Gauss

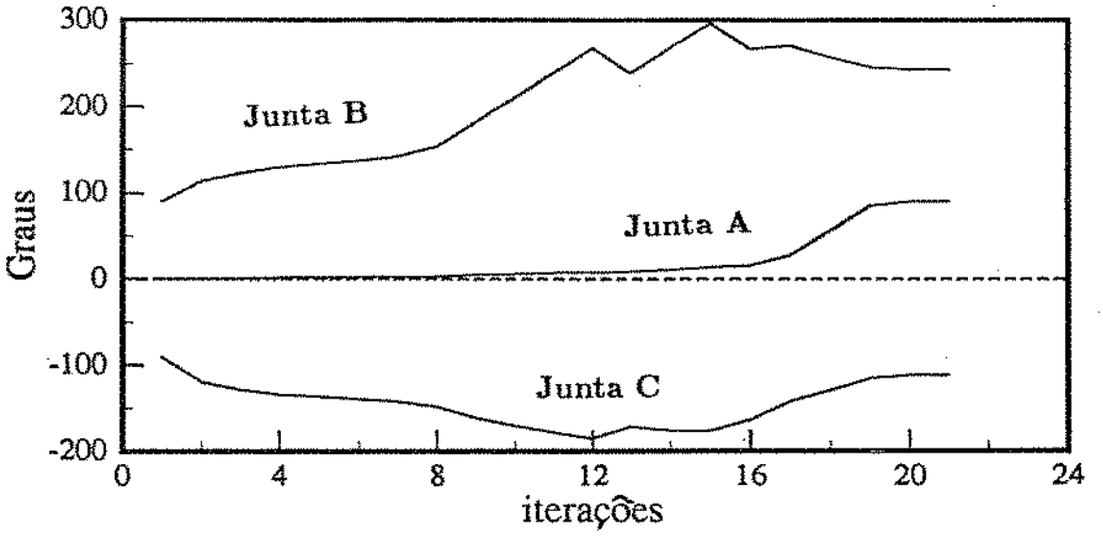
Gráfico 1.b



CONFIGURAÇÃO FINAL											
POSICAO (mm)			ORIENT. (graus)			coord. articulares (graus)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
10	20	30	10	40	40	240	184	-42	-2	22	94

Exemplo 2 : Posição - Miss

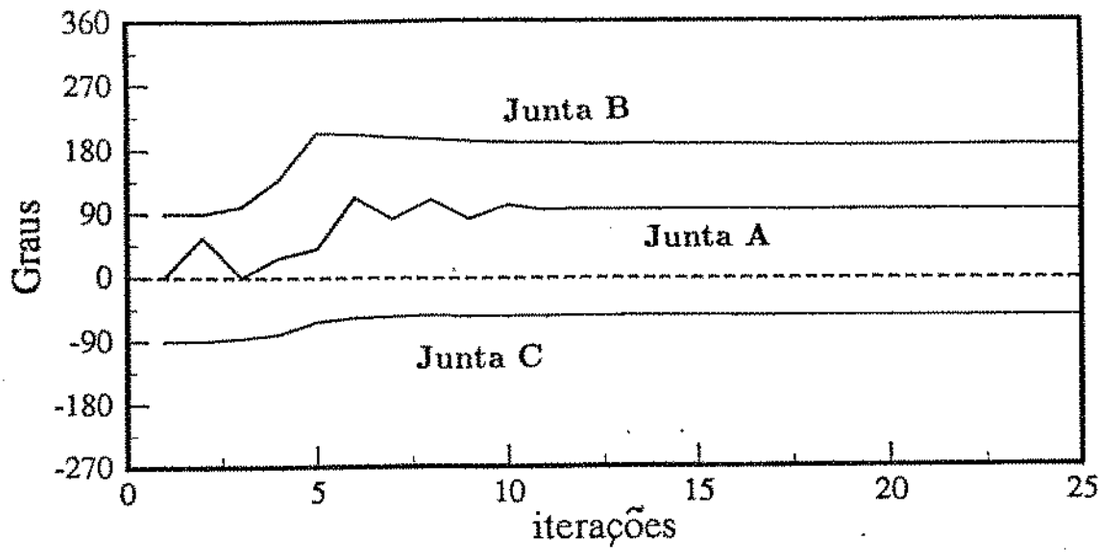
Gráfico 2.a



CONFIGURAÇÃO FINAL											
POSICAO (mm)			ORIENT. (graus)			coord. articulares (graus)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
10	20	30	10	40	40	91	245	-111	-141	170	175

Exemplo 2 : Posição - Gauss

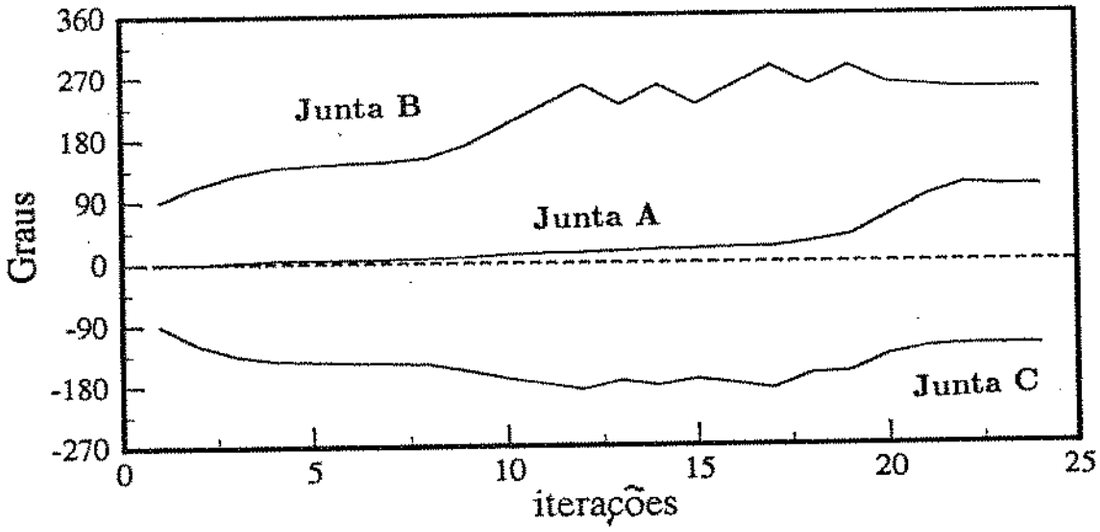
Gráfico 2.b



CONFIGURAÇÃO FINAL											
POSICAO (mm)			ORIENT. (graus)			coord. articulares (graus)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
0	100	150	90	20	0	96	187	-53	305	7	-145

Exemplo 3 : Posição - Miss

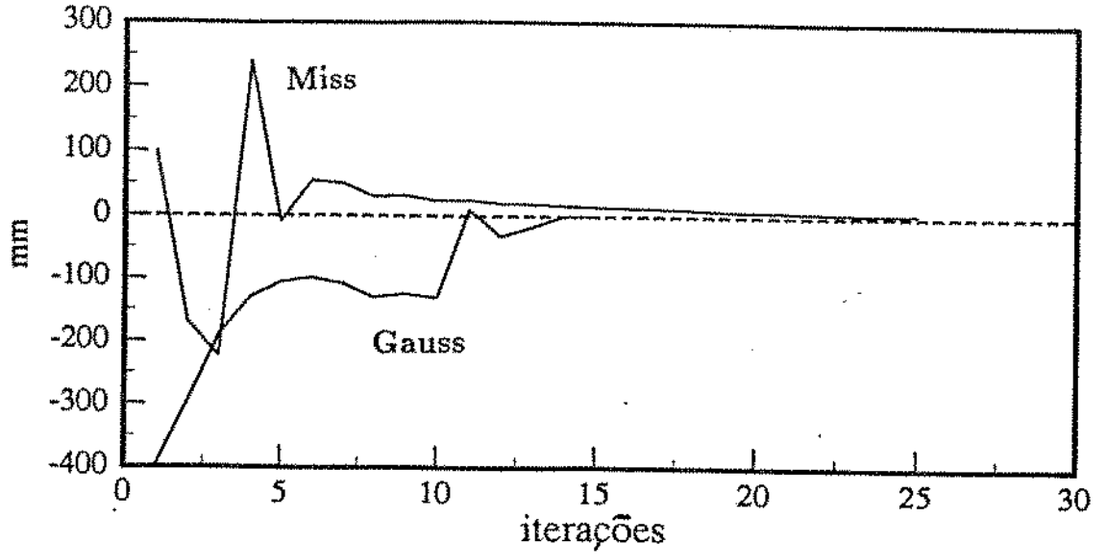
Gráfico 3.a



CONFIGURAÇÃO FINAL											
POSICAO (mm)			ORIENT. (graus)			coord. articulares (graus)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
0	100	150	90	20	0	109	251	-123	-154	131	177

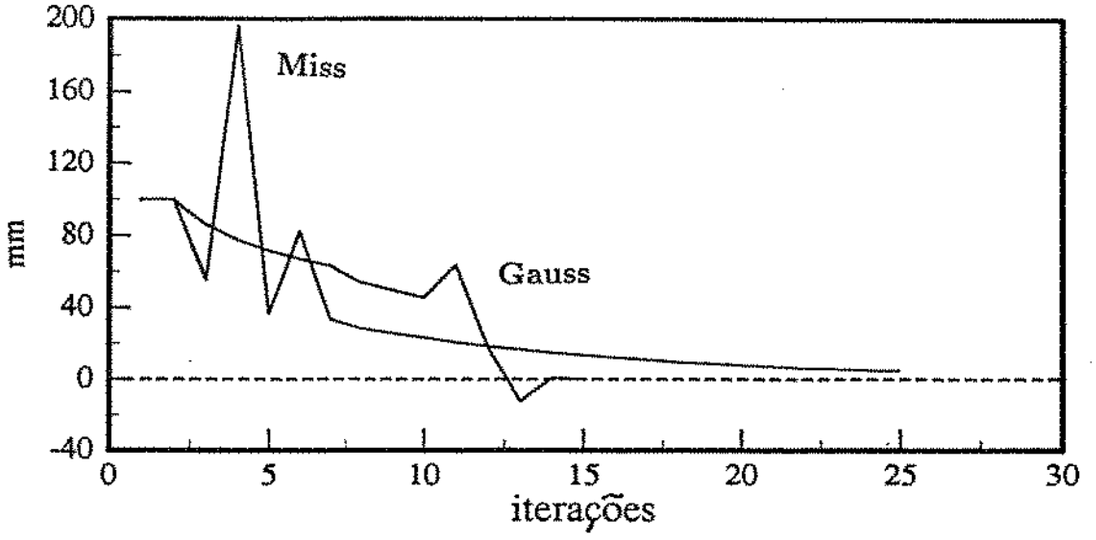
Exemplo 3 : Posição - Gauss

Gráfico 3.b



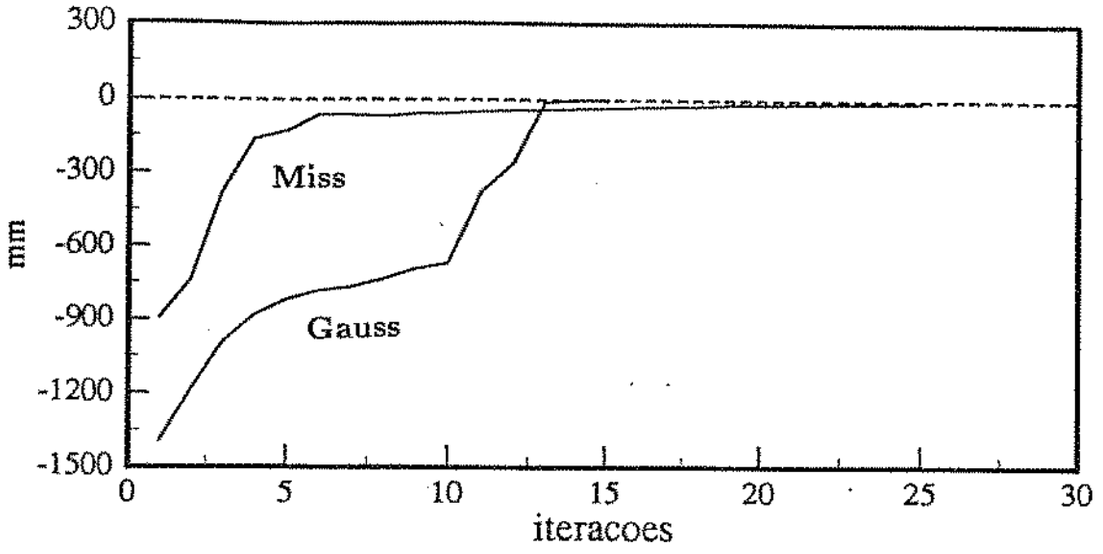
Exemplo 1 : Erro de posição da coordenada X - Miss x Gauss

Gráfico 4



Exemplo 1 : Erro de posição da coordenada Y - Miss x Gauss

Gráfico 5



Exemplo 1 : Erro de posição da coordenada Z - Miss x Gauss

Gráfico 6

Através da análise dos gráficos de posição angular (1, 2 e 3) nota-se que geralmente o método de Miss apresenta um comportamento mais brusco no início da trajetória, mas estabiliza-se rapidamente passando a ter um comportamento mais suave que o método de Gauss.

Os gráficos 4, 5 e 6 indicam que o método de Gauss possui melhor convergência que o de Miss, apresentando portanto um menor número de iterações. Porém, isto não significa que este seja mais rápido, pois apresenta uma maior complexidade de implementação quando comparado ao método de Miss.

Nesta comparação entre os dois métodos, podemos verificar que os diversos gráficos apresentados demonstram que, no cômputo geral, o método de Miss apresenta uma convergência mais suave que a apresentada pelo método de Gauss, tornando-o mais atraente para uso em aplicações de controle, onde é necessária a geração de uma trajetória suave e com grande número de pontos intermediários.

Deve-se ressaltar que as variações do método de Miss descritas anteriormente apresentam uma convergência mais rápida que a do método aqui testado, melhorando ainda mais a sua performance.

O método de Gauss mostrou ser um método eficiente na convergência e portanto adequado para aplicações onde a posição final é mais importante que os pontos intermediários gerados.

3.4 CONCLUSÕES

Neste capítulo descreveu-se o modelo do robô MANUTEC r-3, que foi utilizado nos algoritmos para a modelagem cinemática inversa. Abordou-se então a implementação desses algoritmos e posteriormente foi feita a comparação entre os métodos de Miss e Gauss. Esta comparação auxilia na identificação das aplicações que são mais adequadas aos dois métodos.

Existe uma quantidade enorme de situações onde a modelagem cinemética inversa pode ser aplicada. Situações tais como detecção de colisões em tempo real, que não foram objeto de estudo no presente trabalho, serão abordados em outra tese de mestrado a ser defendida na FEM UNICAMP. Como já mencionado anteriormente, o objetivo de nosso trabalho era caracterizar os algoritmos, para que pudéssemos escolher o melhor para nossa aplicação em controle cartesiano. Embora o método de Miss apresente problemas quanto à orientação que ainda não foram solucionados, para a evolução deste trabalho, este foi o método escolhido para ser implementado na Unidade Central de Supervisão (UCS), por apresentar melhores características de geração de trajetórias.

No próximo capítulo serão apresentadas as arquiteturas de controle usualmente utilizadas para controle de robôs e discutida a arquitetura proposta para o supervisor de controle a ser implementado.

CAPITULO 4 : ARQUITETURAS DE CONTROLE

Neste capítulo aborda-se inicialmente as várias propostas de arquiteturas de controle encontradas na literatura, com ênfase numa arquitetura de processamento que possibilite uma distribuição das tarefas entre os processadores que compõe o sistema [1].

Posteriormente, descreve-se a arquitetura de controle e a arquitetura de hardware desenvolvida neste trabalho, onde são detalhadas as funções de cada unidade de processamento e suas interfaces.

4.1 TENDÊNCIAS NA DEFINIÇÃO DE ARQUITETURAS

A utilização de controladores com processamento distribuído possui várias vantagens [2], tais como:

i) minimização do custo e complexidade, obtendo-se um ótimo desempenho devido à utilização de processadores adequados a cada aplicação.

ii) independência para o projeto, codificação e teste dos softwares de cada processador.

iii) modularização do projeto, podendo ser reduzida a complexidade das funções desenvolvidas quando da utilização de menos recursos para uma aplicação específica.

iv) distribuição de tarefas, de modo a facilitar a detecção dos defeitos ocorridos, pois somente os módulos responsáveis pelas tarefas "defeituosas" serão verificados.

Por outro lado, uma desvantagem do uso de vários microprocessadores é a dificuldade de comunicação entre os mesmos, principalmente em sistemas em que o atraso devido à comunicação é prejudicial ao desempenho do sistema. Necessita-se também de ferramentas voltadas a testes de integração entre as diversas unidades do sistema, tais como analisadores lógicos e analisadores de protocolo, que são de muita utilidade para sistemas com esse grau de complexidade.

4.1.1 Arquitetura proposta por Klafter

A figura 4.1 apresenta a estrutura do controlador. Esta arquitetura é citada como um exemplo onde assume-se que exista algum tipo de interface de comunicação de alta velocidade entre o controle central e os outros módulos de hardware.

Neste sistema são utilizados processadores separados para controlar cada junta associada ao manipulador. As únicas informações que estes módulos necessitam do sistema mestre são os dados de referência e sincronização, os quais são transferidos através do barramento comum. Para sincronizar os processadores das juntas, uma mensagem é enviada a todos os processadores simultaneamente, informando que os algoritmos devem utilizar novas referências.

O controle de cada junta é realizado por meio de um microprocessador, possibilitando a divisão de tarefas de controle e liberando o micro mestre para o controle de trajetórias, interface homem-máquina e outras atividades de sua responsabilidade.

Outra característica do sistema é a sua modularidade, pois o barramento permite o acoplamento de placas diversas, para a realização de tarefas específicas, tal como controle de esforços, controle a partir de tratamento de imagem, etc.

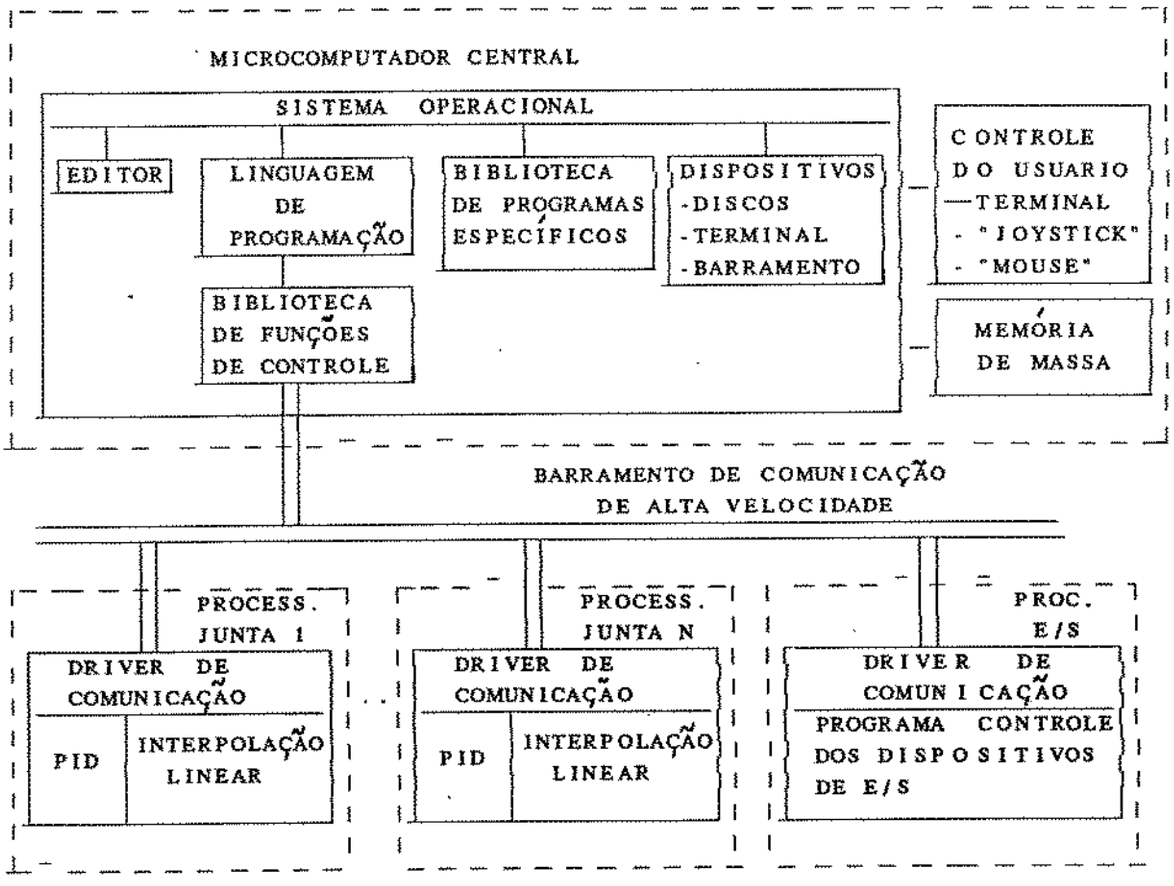


Figura 4.1 : Arquitetura proposta por Klafter

4.1.2 Arquitetura proposta por Zheng

A arquitetura proposta por Zheng [3] é apresentada na figura 4.2. O autor apresenta uma proposta de cálculo de torques aplicados em tempo real para a realização de um controlador que leve em consideração a dinâmica do manipulador.

A proposta é utilizar um sistema a multiprocessadores composto de uma CPU central e um grupo de CPUs satélites para dividir o cálculo da modelagem dinâmica feita através da formulação de *Newton-Euler*. A tarefa de cada CPU satélite é controlar uma junta, processando seus dados relativos. A tarefa da CPU central é calcular os torques a serem aplicados.

O software é o mesmo para todas as CPUs satélites. O esquema a multiprocessadores resultante é flexivel e modular, porém o esquema de comunicação com linhas entre todos os processadores pode ser muito difícil de ser implementado, ocasionando um *overhead* de comunicação oneroso ao sistema.

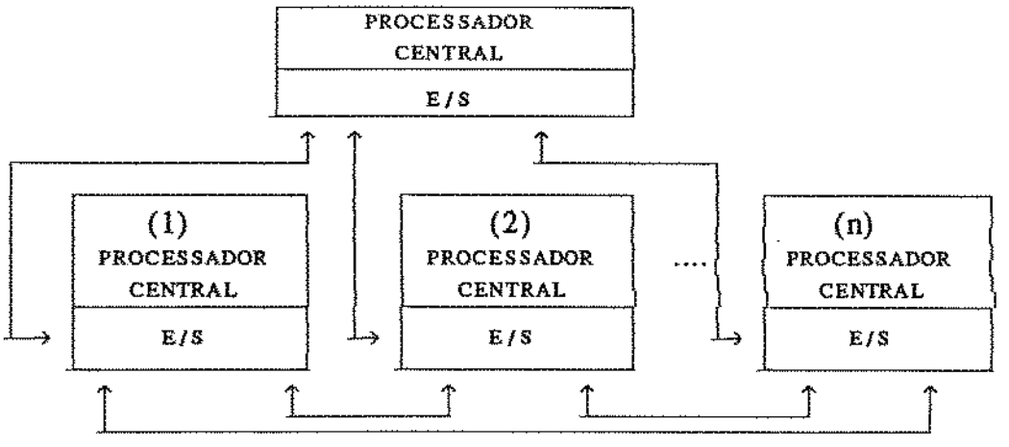
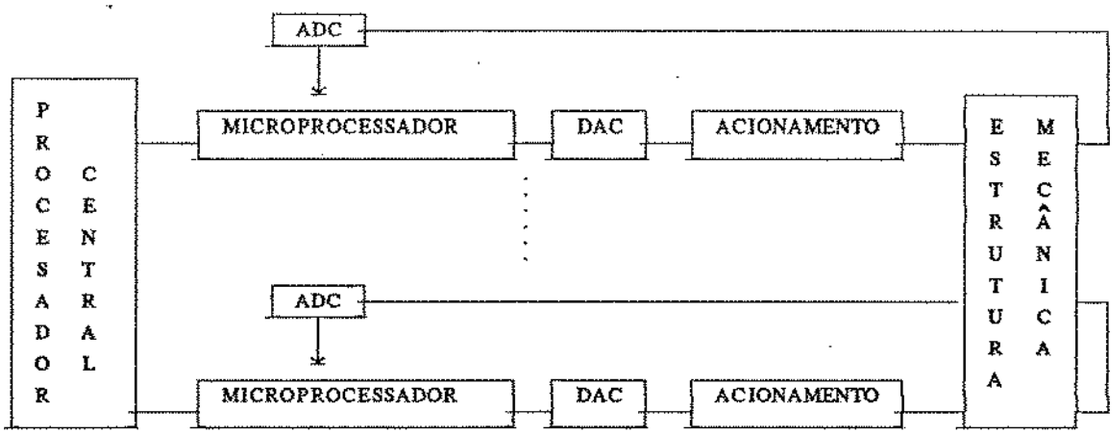


Figura 4.2 : Arquitetura proposta por Zheng

4.1.3 Arquitetura proposta por Bestaqui

A metodologia de controle proposta por Bestaqui [4] utiliza controle hierárquico e observadores descentralizados. A estrutura proposta está apresentada na figura 4.3. O procedimento utilizado é dividir o cálculo de dados das juntas necessários para as equações de *Newton-Euler* e alocar processadores dedicados nas juntas para realizar a observação e controle das mesmas.

O microprocessador central sincroniza a operação de todos os microprocessadores e calcula os torques nominais. O período de amostragem do processador central pode ser maior do que o período dos processadores locais, permitindo que se consiga calcular o modelo matemático centralizado do manipulador, usando posições, velocidades e acelerações das juntas.



onde:
ADC = Conversor analógico-digital
DAC = Conversor digital-analógico

Figura 4.3 : Estrutura proposta por Bestaqui

Nota-se nas arquiteturas apresentadas uma semelhança em alguns pontos, tais como:

1) proposição da arquitetura a multiprocessadores como um meio de aumentar a capacidade de processamento.

2) divisão das tarefas de controle do manipulador em subtarefas de modo a obter um processamento distribuído.

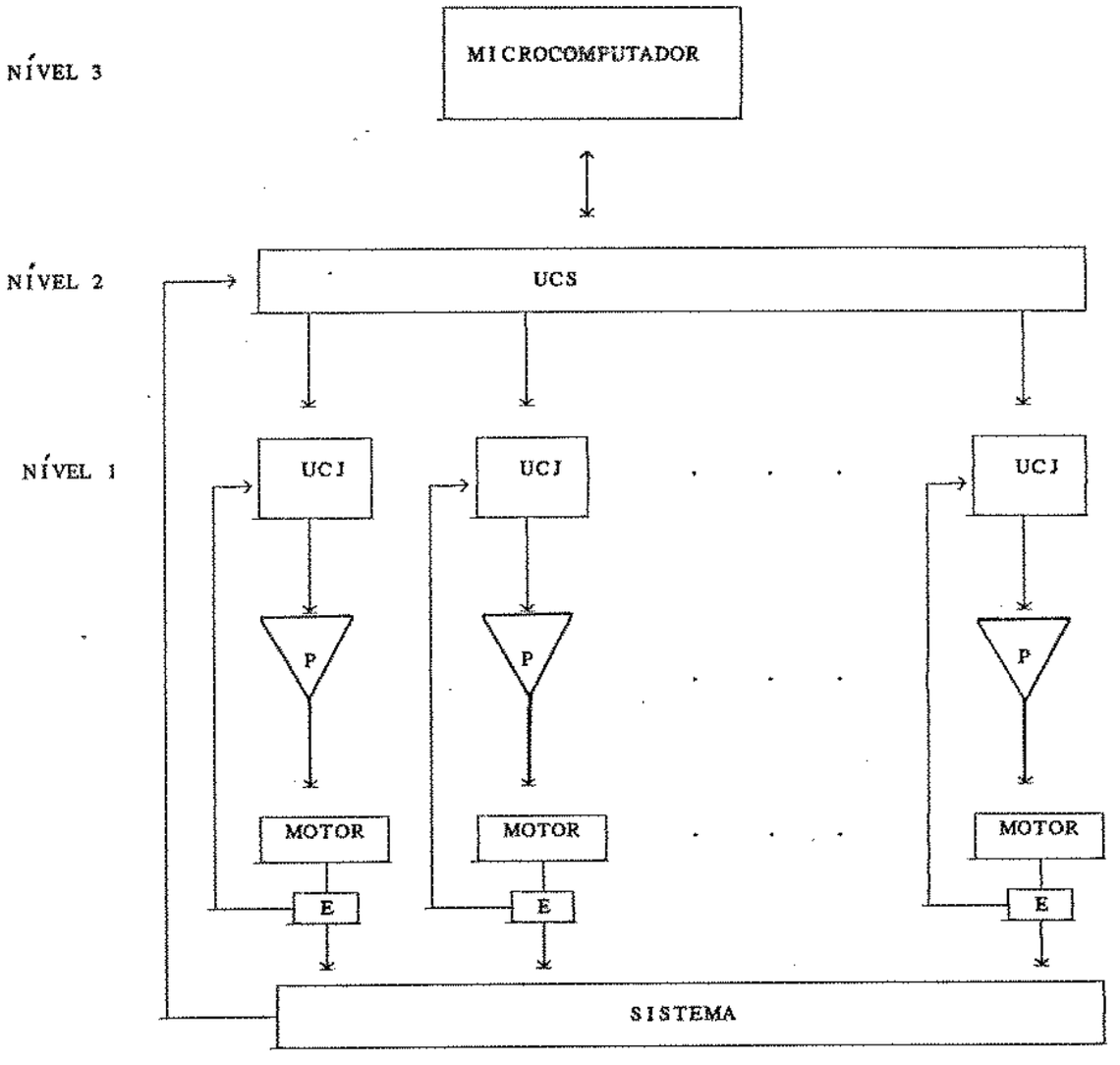
3) utilização de processadores dedicados ao controle das juntas.

4.2 ARQUITETURA DE CONTROLE PROPOSTA

A arquitetura proposta hierarquiza o controle de movimentos de robôs manipuladores em dois níveis: um responsável pelo controle dos servomecanismos das juntas e outro, hierarquicamente mais elevado, responsável pelo controle da trajetória na extremidade do efetuador do manipulador. Esta arquitetura é baseada na utilização de microprocessadores dedicados à execução de cada tarefa, onde se tem no primeiro nível, unidades processadoras para cada junta do sistema, que se comunicam com a unidade processadora responsável pelas funções de geração de trajetórias, controle dos movimentos do manipulador e supervisão do sistema.

Este tipo de estrutura, onde um elemento central, hierarquicamente superior ao controle de juntas, possui funções de controle de trajetórias, possibilita a implementação de uma segunda malha de controle no sistema. Esta malha, mais externa e comum à todas as juntas, recebe informações de posição da ferramenta do robô através da utilização de sensores de esforço (basicamente constituídos de *strain-gages*) [21]. Os dados obtidos são tratados e analisados pelo nível de supervisão . Este tratamento possibilita a minimização da margem de erro do posicionamento do efetuador do robô.

A figura 4.4 mostra os níveis hierárquicos propostos, assim como os elementos onde estarão implementados suas funções.



ONDE E = ENCODER
P = AMPLIFICADOR DE POTÊNCIA
UCJ = UNIDADE DE CONTROLE DE JUNTA
UCS = UNIDADE CENTRAL DE SUPERVISÃO

Figura 4.4 : Estrutura Hierárquica de Controle.

Esta arquitetura é baseada na seguinte distribuição de funções para cada nível hierárquico de controle :

NÍVEL 1 : Trata as malhas clássicas de controle e implementa os algorítmos de controle para cada junta do robô. Neste nível são implementadas *n* unidades processadoras, uma para cada grau de liberdade do robô. Cada unidade controla apenas uma junta com sua inércia associada, não tendo nenhum tipo de acoplamento com as demais unidades de mesmo nível hierárquico. Com a utilização de processadores dedicados, é possível a implementação de algoritmos relativamente complexos, juntamente com a elevação das taxas de amostragem dos dados colhidos pelos sensores de posição da junta. Estes fatores são essenciais na obtenção de resultados significativos na precisão do controle efetuado para o posicionamento do elemento terminal de um robô.

NÍVEL 2 : Este nível tem como função coordenar e sincronizar todas as unidades de nível 1, além de realizar o interfaceamento daquele nível com os processos de interação homem-máquina executados pelo nível 3. Os modelos direto e inverso do robô são implementados neste nível, tendo-se portanto a possibilidade de realização de várias funções de controle, tais como a de geração automática de trajetória, controle de posição final da ferramenta do robô, etc. Para tanto, na unidade onde este nível é implementado, foi desenvolvida uma interface com um dispositivo de sensor de esforço , o qual se interfaceia diretamente com a ferramenta do robô. Para a implementação dessas funções, a unidade processadora de nível 2 se comunicará em modo *full-duplex* com todas as unidades de nível 1 e também com o microcomputador que implementa o nível 3.

NÍVEL 3 : Atua nas funções de gerência do sistema. Essencialmente é o posto de trabalho no qual o operador comanda as diversas funções implementadas. Uma interface visual amigável deverá ser implementada de modo a possibilitar uma noção geométrica dos movimentos realizados pela ferramenta do robô. Outras funções como geração de trajetórias *off-line*, movimentos de uma única junta e movimentos apenas em um eixo cartesiano também serão possíveis através deste posto de comando, cuja maioria das aplicações não terão características de tempo real, isto é, sua atuação será essencialmente *off-line*. Este nível é implementado através de um microcomputador compatível com a linha IBM PC-AT.

4.2.1 Características gerais do sistema de controle

Descreve-se a seguir algumas funções fundamentais para a concepção de um sistema genérico de supervisão de controle para um robô industrial. Além das funções descritas, uma série de outras podem ser especificadas no decorrer do tempo, com o intuito de, gradativamente, possibilitar a agregação ao sistema de funções sofisticadas, que poderão ser demandadas em futuros trabalhos na área. Portanto, o objetvo inicial deste trabalho não é o de esgotar as descrições de todas as possíveis funções existentes, e sim o de especificar um elenco mínimo para sua imediata implementação. Este conjunto mínimo deve fornecer resultados que servirão como base para a validação do sistema.

4.2.1.1 Posicionamento inicial do robô

No início/fim de operação de um robô, se faz necessário um procedimento de se colocar o robô numa posição previamente conhecida. Geralmente esta é uma posição de descanso, onde inercialmente se obtém o equilíbrio estático do manipulador.

Antes do robô ser posto em operação, o operador deve manualmente posicionar junta por junta o robô nesta posição especificada. Após a realização desta operação o sistema está sincronizado, e o sistema de controle possui conhecimento do real posicionamento das diversas juntas e portanto encontra-se pronto para iniciar a operação.

Quando da movimentação de cada junta no procedimento inicial de se buscar a posição de equilíbrio do robô, o sistema busca o sinal de referência do encoder. Encontrada a posição desejada daquela junta, carrega-se o contador de impulsões com um dado valor e temporiza-se um sinal visual para o operador, indicando que a operação de posicionamento para aquela junta foi completada. Os valores que são carregados no contador de cada junta ao fim dessa operação são determinados em função do espaço de trabalho desejado para o robô.

Esta operação se processa desta forma pois quando da montagem do sistema, cada encoder será mecanicamente ajustado, de modo que o pulso de referência do mesmo seja ativado quando a junta estiver na sua posição inicial desejada .

A figura 4.5 ilustra as regiões de trabalho de uma junta, através da definição do valor inicial a ser carregado no contador de impulsões do encoder. Como margem de segurança, por software é garantido um limite máximo de amplitude, tanto num extremo quanto no outro, de modo que o sistema nunca atinja seus limites físicos de operação.

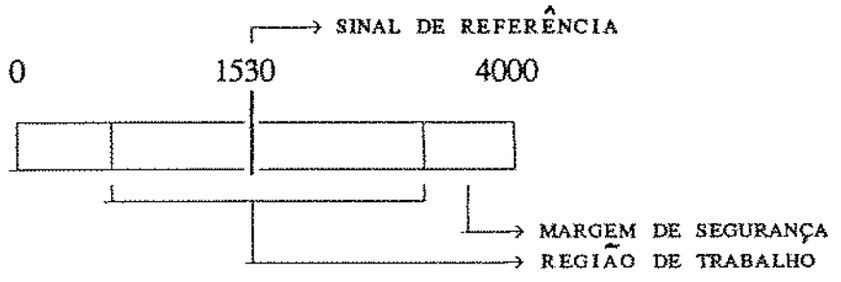


Figura 4.5 : Exemplo de região de trabalho de um encoder

4.2.1.2 Dispositivo de HALT

O sistema deve possuir uma tecla de HALT na Unidade Central de Supervisão (UCS), de modo que no momento de seu acionamento, todo e qualquer movimento que estiver sendo executado cesse imediatamente. É um dispositivo de segurança que permite ao operador evitar rapidamente que algum movimento indesejado seja executado.

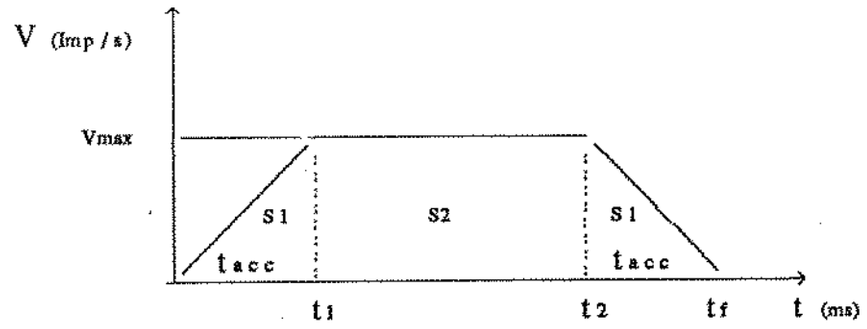
4.2.1.3 Geração de trajetória

Basicamente a geração de trajetórias pode ser executada de duas maneiras no sistema.

i) trajetória gerada no PC

O algoritmo utilizado consiste em calcular a velocidade máxima proporcionalmente ao deslocamemto de cada junta, de modo que haja simultaneidade entre elas, ou seja, as juntas devem partir de uma determinada posição ao mesmo tempo e chegar à posição desejada simultaneamente.

Isto decorre de que num movimento qualquer, o deslocamento de cada junta não é uniforme, com algumas se deslocando muito, enquanto outras se deslocam pouco. O algoritmo então parametriza a junta que terá o maior deslocamento como a junta de maior velocidade permitida pelo sistema. A velocidade das outras juntas então será proporcional à relação do seu deslocamento com o maior deslocamento ocorrido no movimento. A figura 4.6 mostra o perfil de velocidade de cada junta do sistema num determinado movimento. O equacionamento matemático das relações existentes entre número de impulsões, velocidade angular e espaço é apresentado no ANEXO C deste trabalho.



onde:

tacc:	tempo de aceleração e de frenagem em ms
vmax:	velocidade máxima em impulsões por segundo
S_1:	deslocamento angular na fase de aceleração
S_2:	deslocamento angular na fase de velocidade constante
S_1:	deslocamento angular na fase de desaceleração

Figura 4.6 : Perfil de velocidade de uma junta

Com este perfil para cada junta, garante-se a simultaneidade do movimento entre elas. Este algoritmo é executado levando-se em conta o modelo geométrico do robô, isto é, trabalha-se diretamente com os ângulos das juntas do sistema para se obter a posição espacial final.

Na arquitetura proposta neste trabalho, o microcomputador, de posse das posições inicial e final de um determinado movimento da ferramenta do robô, executa este algoritmo *off-line* e posteriormente transmite para a Unidade Central de Supervisão (UCS) um arquivo de pontos intermediários para cada junta de controle. Esta então sincroniza as amostras e as envia às Unidades Controladoras de Junta (UCJ).

ii) trajetória gerada na UCS

Neste modo, através da interface homem-máquina do PC, o operador fornece a posição inicial e a posição final da ferramenta do robô. Estes dados são então enviados para a Unidade Central de Supervisão. De posse destes pontos, através de algoritmos de modelagem cinemática inversa, são gerados pontos intermediários que são convertidos em um vetor de ângulos associados às juntas. Estes são então transferidos sincronizadamente às unidades controladoras de junta (UCJ). Com este método consegue-se uma geração autônoma da trajetória.

A figura 4.7 mostra um quadro comparativo dos métodos utilizados para a geração de trajetórias e as funções de cada nível de controle.

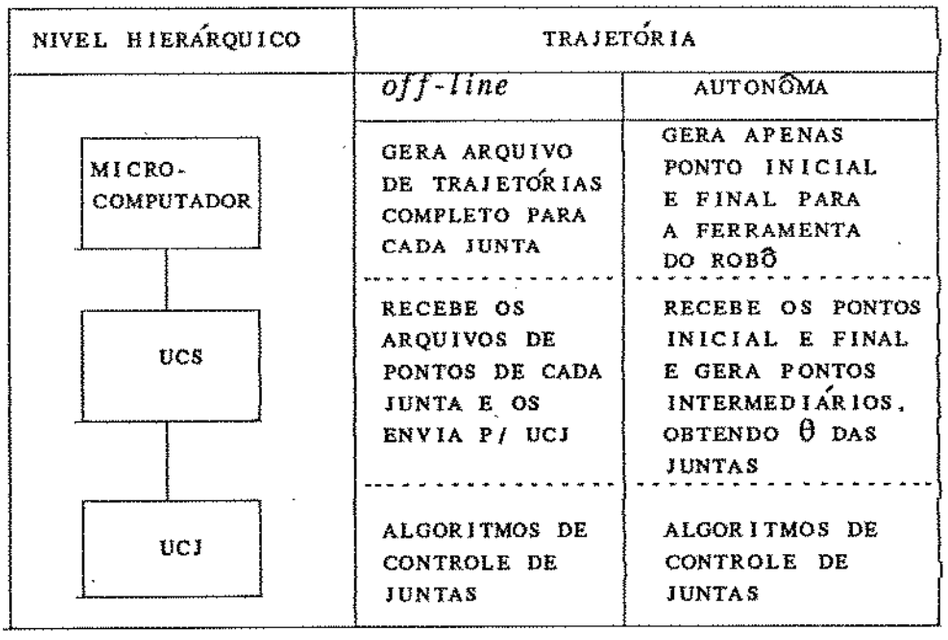


Figura 4.7 : Níveis de controle na geração de trajetória

4.3 IMPLEMENTAÇÃO DO SISTEMA PROPOSTO

Descreve-se aqui as especificações da arquitetura de hardware do supervisor de controle. Neste trabalho nos preocupamos em prover o sistema com as características necessárias, sem nos preocuparmos com os possíveis limitantes de uma implementação. A implementação do sistema aqui especificado está sendo efetuada no trabalho de tese do aluno Carlos Henrique Dias, do Departamento de Projeto Mecânico da FEM - UNICAMP [24].

4.3.1 Características mecânicas

O sistema é constituído de um sub-bastidor contendo 8 *slots*, sendo um para a fonte de alimentação, outro para a Unidade Central de Supervisão e o restante para as Unidades de Controle de Junta. Cada unidade se conecta ao painel traseiro através de um euro-conector de 96 pinos. Tanto a interface com os dispositivos de potência como a interface com o microcomputador se localizam na parte traseira do sub-bastidor.

4.3.2 Interfaces

Cada unidade de controle é colocada num determinado *slot* do sub-bastidor , sendo que a UCS se localiza num *slot* pré-determinado, pois possui comunicação serial com todas as outras unidades do sistema.

Os sinais utilizados na interface entre uma UCJ e a UCS no painel traseiro do sub-bastidorsão mostrados na figura 4.8.

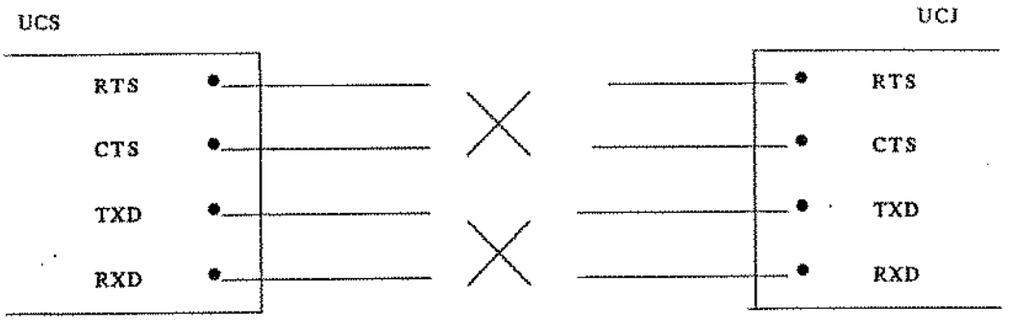


Figura 4.8 : Sinais da interface UCS - UCJ

Descrição dos sinais de interface UCS-UCJ:

- CTS - *Clear to Send* - Sinal padronizado na interface RS232, que indica que a unidade está pronta para receber dados.

- RTS - *Request to Send* - Sinal padronizado na interface RS232, que indica uma requisição de envio de dados.

- TXD - Sinal de dados de transmissão

- RXD - Sinal de dados de recepção

A unidade de supervisão possui uma interface com um microcomputador, que realiza a função de nível 3 de controle. A comunicação UCS-UCJ, assim como a comunicação UCS-PC, é realizada através de uma linha serial *full-duplex* assíncrona, situada no painel traseiro do sub-bastidor

A comunicação assíncrona entre as unidades foi utilizada por ser mais simples e também porque os cálculos estimados do volume de informação necessário a ser transmitido para as aplicações inicialmente previstas permitem o uso de interfaces com baixa taxa de transmissão. Estes e outros aspectos referentes ao dimensionamento do sistema e análise de performance serão abordados no capítulo 6 do presente trabalho.

A figura 4.9 representa um típico formato de mensagem assíncrona, iniciando por um *start bit*, seguidos de sete bits de dados, um bit de paridade e um *stop bit*. O *start bit* é uma transição de nível alto para baixo, detectada por um receptor assíncrono, sendo na realidade um bit de informação notificando o receptor do início da chegada de uma mensagem de dados.

Quando da detecção de um *start bit*, o receptor ativa um circuito de clock para prover o *latcheamento* dos pulsos durante o intervalo de bits de dados esperado. O bit de paridade é provido para detecção de erros, sendo calculado tanto no transmissor como no receptor; ao fim da mensagem transmitida, o transmissor envia este bit, sendo os dois resultados comparados pelo receptor. O *stop bit* retorna o estado da linha de transmissão para seu estado quiescente, isto é, o nível alto, até que uma nova transição de alto para baixo indique o início de uma nova sequência de bits de dados. Durante a recepção de dados, o controlador serial utilizado retira autonomamente o *start* e o *stop bit*, passando para a CPU tratar somente os bits que realmente contém informação.

START	D0	D1	D2	D3	D4	D5	D6	D7	PARIDADE	STOP
	X	X	X	X	X	X	X	X		

X = ESTADO DON'T CARE

Figura 4.9 : Formato de mensagem assíncrona

4.3.3 Unidade de Controle de Junta

A Unidade de Controle de Junta (UCJ) é responsável pela aquisição e tratamento de dados provenientes do encoder. Este tratamento é feito através do processamento de algoritmos de controle e também pelo envio de novas amostras ao dispositivo de potência que a ela está ligado. Paralelamente a estas funções, a UCJ deve se comunicar com a UCS, através de sua linha serial.

A seguir descreve-se os blocos de funções que compõem esta unidade, juntamente com seu mapeamento na estrutura de controle. A figura 4.10 mostra um diagrama de blocos da UCJ.

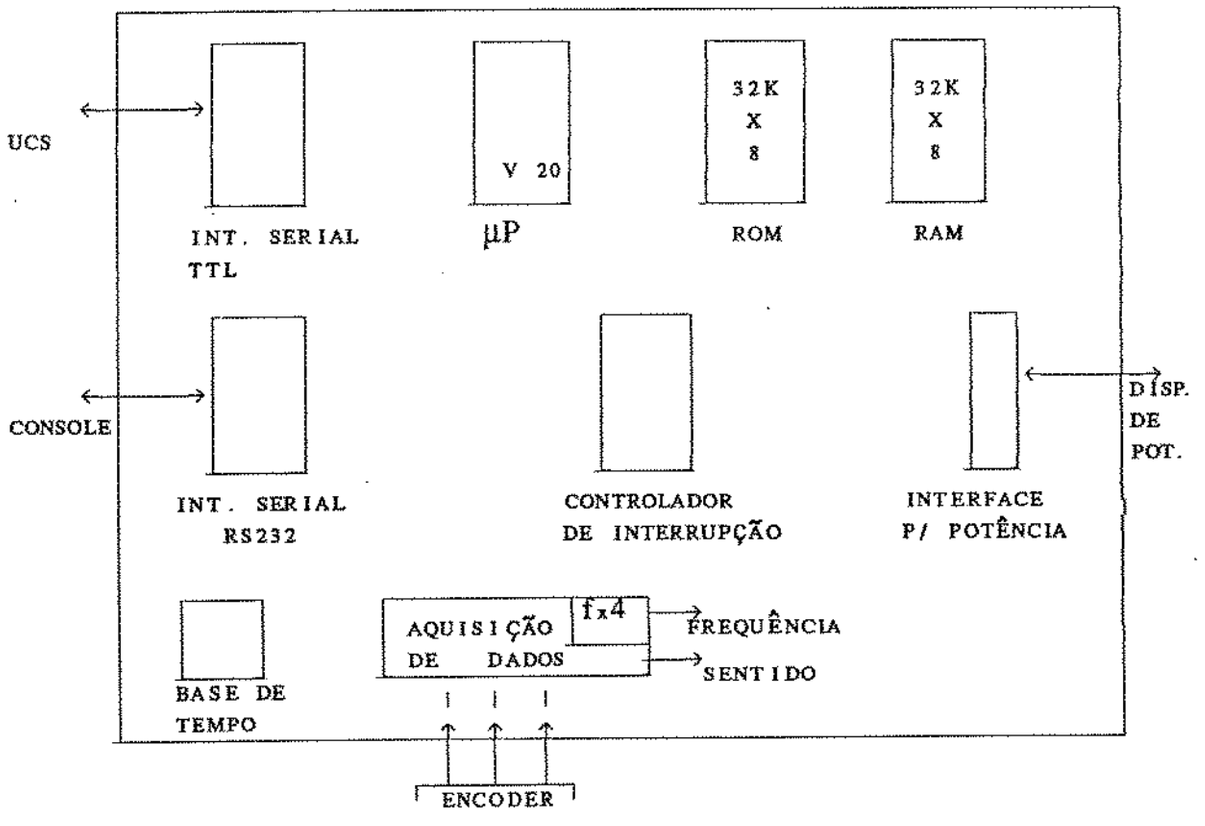


Figura 4.10 : Diagrama de blocos da Unidade de Controle de Junta

i) Microprocessador

A unidade UCJ utiliza o microprocessador V20 da NEC. Este microprocessador possui um *bus* de endereço de 20 bits, podendo portanto acessar uma capacidade máxima de memória de 1 Mega byte. O *bus* de dados é de 8 bits, porém internamente trabalha com registradores de 16 bits. A freqüência de operação do relógio utilizada na unidade é de 8 MHz.

ii) Memória

A capacidade de memória da UCJ é de 32 Kbytes de memória ROM e 32 Kbytes de memória RAM estática. O processador acessa tanto RAM como ROM sem nenhum ciclo de *wait-state*.

iii) Interface configurável com o dispositivo de potência do robô;

A unidade foi projetada de modo a se ter uma interface inteiramente substituível por outra, sem necessidade de nenhuma mudança adicional de hardware, de modo a possibilitar no futuro que outras interfaces venham a ser utilizadas. Inicialmente optou-se por implementar uma interface paralela, que se comunica com um hardware desenvolvido na tese de mestrado do aluno Márcio Fantini Miranda, que realiza o controle de um servomecanismo a partir de impulsões lógicas [6].

Na transmissão de um sinal para a unidade de potência do robô, tem-se a possibilidade de se usar também um sinal digital de saída para controle *on-off*.

iv) Interface de comunicação serial com a unidade de supervisão.

O controlador serial Z8530 da Zilog implementa duas interfaces seriais na unidade, denominadas A e B . A interface A se comunica com a UCS enquanto que a interface B está ligada a *drivers* e *receivers* RS232, de modo ser possível a comunicação com um terminal para depuração de software da unidade, ou para qualquer outra aplicação.

v) Sistema de aquisição de dados do encoder

Uma vez estando o robô sincronizado (operação realizada no início das atividades do robô), qualquer movimentação de junta ocasiona o envio de impulsões para o sistema de controle do mesmo.

A figura 4.11 apresenta um diagrama de blocos simplificado do sistema de aquisição e processamento dos sinais de encoder da unidade UCJ.

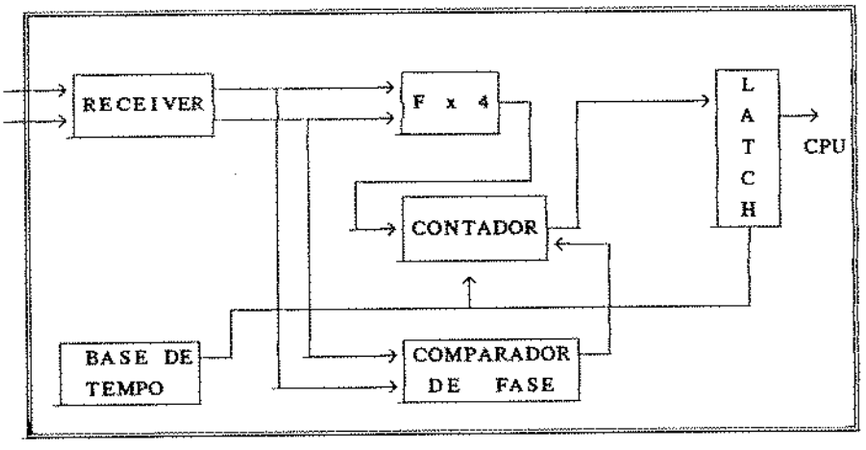


Figura 4.11 : Diagrama de Blocos do Sistema de Aquisição de Dados

A função deste sistema é a obtenção de amostras provenientes das leituras dos sinais enviados pelo encoder, já descritos no capítulo 2. A seguir descreve-se todas as etapas do processo para a transformação destes sinais em amostras a serem utilizadas nos algoritmos de controle de juntas.

a) Eliminação de ruídos

As duas pistas indicativas do número de impulsões do encoder, ao chegarem à UCJ, devem passar por *receivers*, com o objetivo de eliminação de ruídos originados durante a transmissão do sinal. Caso contrário pode haver a presença de *glitches* nos mesmos, o que causaria uma imprecisão na contagem das impulsões , acarretando uma diminuição da qualidade do controle efetuado e contribuindo também para uma perda do referencial do sistema ao longo do tempo.

b) Multiplicação de frequência dos sinais

Posteriormente à passagem pelos *receivers*, multiplica-se a frequência dos sinais do encoder, com o objetivo de aumentar a precisão na medida a ser feita pelo contador de impulsos. Esta multiplicação pode ser por um fator de 2 ou por um fator de 4. Obviamente quanto maior for este fator, melhor será a precisão do controle feito sobre a junta, pois maior será a precisão da discretização da posição angular da mesma.

A figura 4.12 ilustra os dois sinais do encoder e as duas formas de onda possíveis de serem obtidas na saída do bloco multiplicador.

Para se obter a multiplicação por dois da frequência dos sinais de entrada, basta efetuar a operação lógica de OU EXCLUSIVO nas duas pistas. Portanto, esta operação é assíncrona com qualquer outro evento da unidade UCJ.

A multiplicação por 4 necessita de um hardware mais complexo, usando lógica sequencial. Este hardware necessita de um *clock* externo, sendo portanto uma lógica síncrona.

Uma explicação detalhada da multiplicação por quatro da frequência de entrada dos sinais de um encoder é dada no Anexo B do presente trabalho, mostrando-se o circuito final que pode ser implementado num dispositivo lógico programável (PLD).

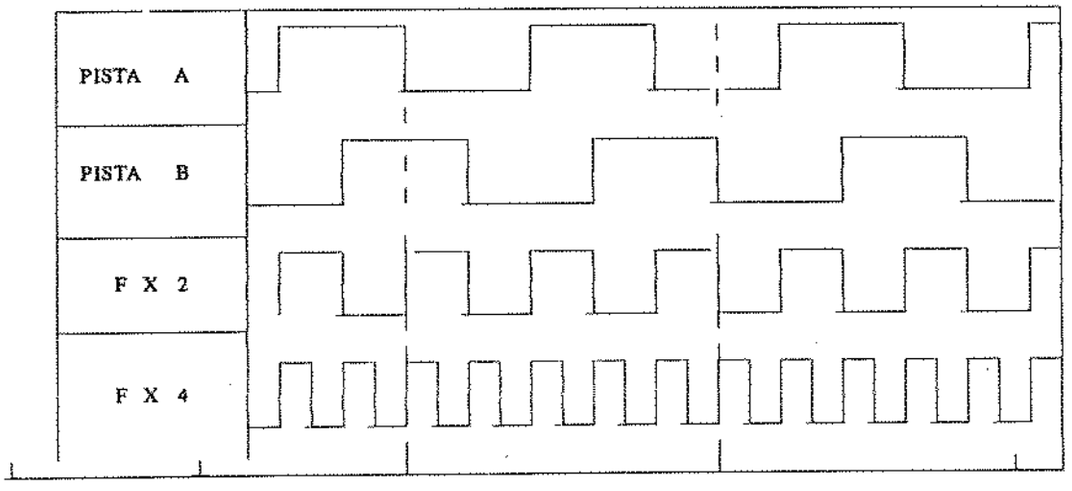


Figura 4.12 : Multiplicação dos sinais do encoder

c) Carregamento do número inicial de impulsões no contador

O contador do número de impulsões implementado no sistema é de 8 bits, do tipo *up-and-down* e com sinal de *load*.

Quando da operação de sincronização do robô, a UCJ aguarda a ativação do pulso de referência do encoder. Este evento gera uma interrupção na unidade. O tratamento desta interrupção ativa um registrador cuja saída esta ligada ao sinal de *load* do contador, fazendo a carga do número base de impulsões , que foi obtido quando da determinação da posição de equilíbrio e do espaço de trabalho do robô.

d) Contagem do número de impulsões

A contagem das impulsões se realiza em torno do número de impulsões carregado na sincronização do robô. Esta contagem será positiva ou negativa, dependendo do sinal de saída do bloco de comparação de fase. Este bloco basicamente verifica qual das pistas está defasada em relação à outra, enviando na saída um sinal lógico previamente convencionado.

e) Obtenção das amostras

O contador, durante todo tempo, registra *on-line* o número de impulsões enviados pelo encoder. A UCJ possui uma base de tempo que, a cada período previamente estabelecido (estrapeado na placa nos valores de 1, 2 e 4 ms), gera um sinal de *strobe* que congela o valor instantâneo do contador em um registrador de leitura. Este valor permanece estável durante todo o próximo período. Durante este tempo, o contador não para de registrar as impulsões. Este período é o tempo hábil de leitura onde a CPU deve ler a amostra de contagem, pois caso contrário ela será perdida.

4.3.4 Unidade Central de Supervisão

A função da UCS é executar os algoritmos de controle de geração de trajetórias e toda a parte de gerência das unidades de controle de juntas, atuando como mediadora entre o elemento de nivel 3 (microcomputador) e as UCJ.

A Unidade Central de Supervisão possui essencialmente o mesmo hardware da UCJ, sem o bloco de aquisição de sinais do encoder e da interface de potência, possuindo em adição uma interface para o sensor de esforço a ser colocado no elemento terminal do robô.

Um diagrama de blocos da UCS é mostrado na figura 4.13.

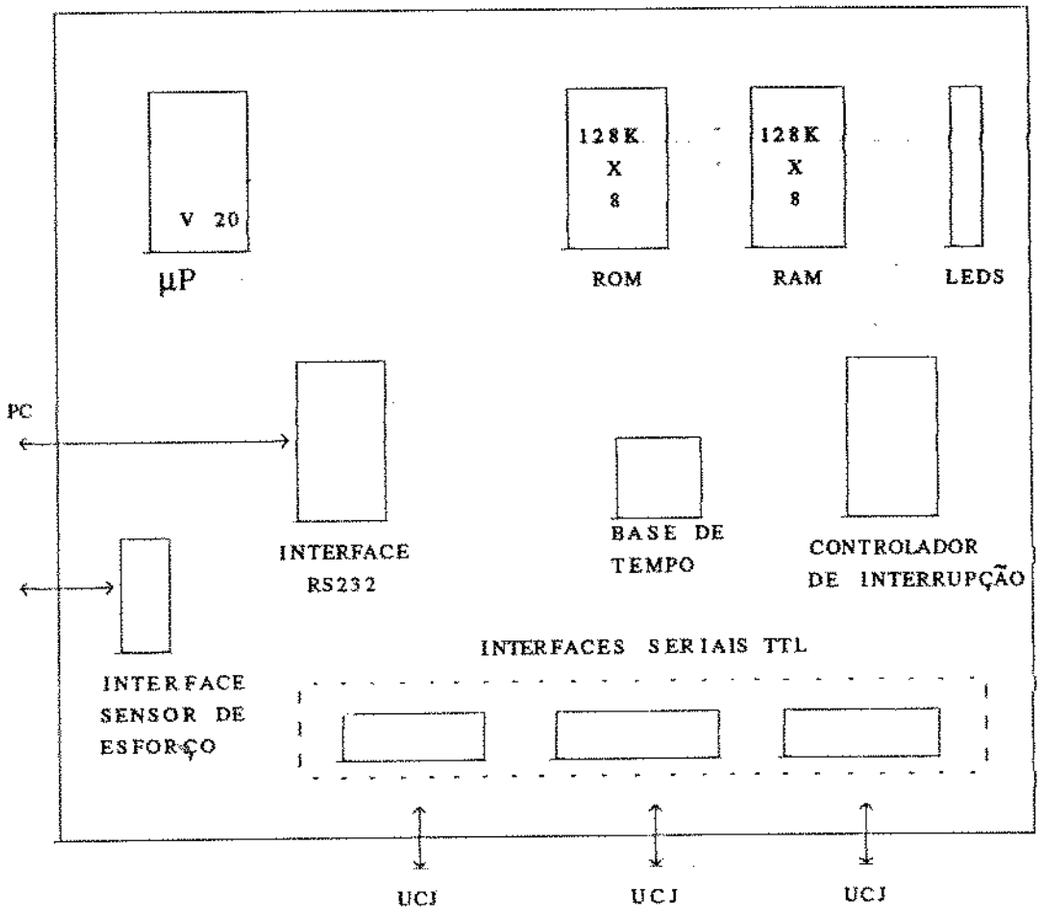


Figura 4.13 : Diagrama de blocos da UCS

A seguir destacamos algumas diferenças no hardware das duas unidades:

i) capacidade de memória

A capacidade de memória da UCS é de 128 Kbytes de memória RAM e 128 Kbytes de memória ROM, embora possa ser subequipada com os mesmos 32 Kbytes da UCJ. A previsão de maior capacidade de memória decorre do fato de que a UCS executará algoritmos mais complexos do que a UCJ, exigindo um maior espaço de armazenamento de dados.

ii) *leds* no painel frontal

Como a UCS tem funções de supervisão, ela foi equipada com 8 *leds* no painel frontal que fazem uma interface homem-máquina de propósito geral durante a operação do sistema. Possíveis aplicações para a visualização em *leds* são : a entrada em falha de alguma unidade do sistema; o recebimento de demasiado número de interrupções; etc.

iii) número de interfaces seriais

Para fins de prototipação do sistema, a capacidade de comunicação da unidade de supervisão foi especificada para três unidades de controle de junta (UCJ). Tem-se ainda nesta unidade, uma linha serial de comunicação com o nível 3 de controle. totalizando quatro linhas seriais a serem implementadas, duas a mais que a UCJ, necessitando-se portanto de dois controladores seriais Z8530. Todas essas linhas tem interface padrão RS-232.

4.4 CONCLUSÕES

Neste capítulo foram descritas algumas arquiteturas de controle da literatura e apresentada a estrutura de controle proposta por este trabalho. Descreveu-se as unidades de controle que compõem o sistema em suas características de hardware. No próximo capítulo aborda-se a arquitetura de software proposta, com os módulos específicos para cada aplicação.

CAPÍTULO 5 : ARQUITETURA DE SOFTWARE

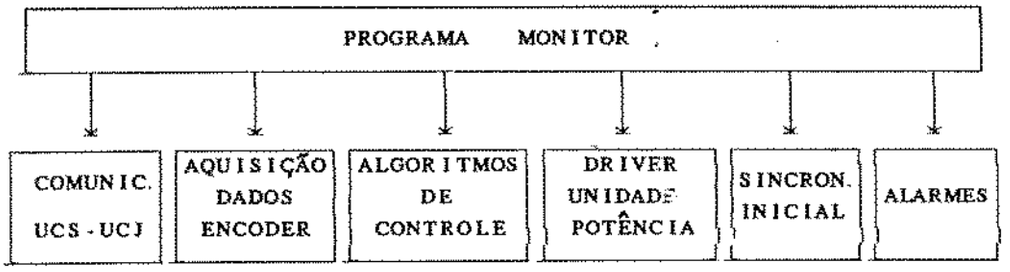
Neste capítulo mostra-se a estrutura de programação especificada para o supervisor de controle. Tanto na Unidade Central de Supervisão como na Unidade de Controle de Junta, esta estrutura é baseada no desenvolvimento de módulos que executam funções específicas. Para a interação entre as várias funções, são definidas as interfaces entre os módulos, que se compõem de mensagens com campos de informação bem definidos.

5.1 ESTRUTURA DE PROGRAMAÇÃO

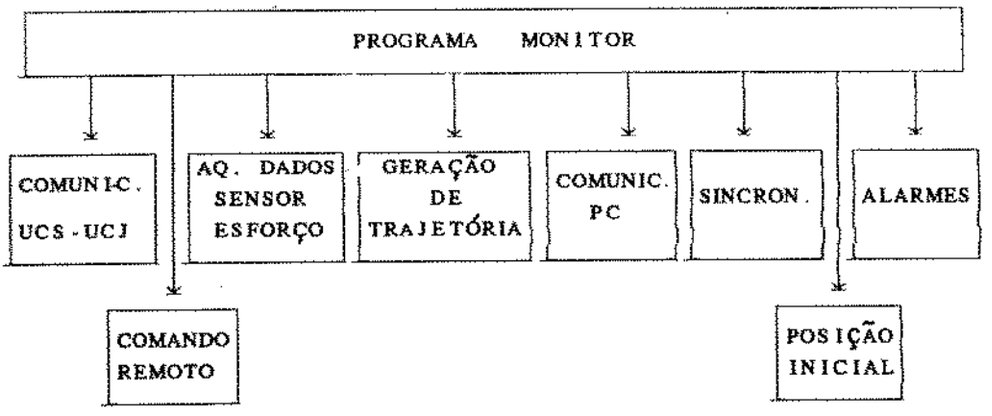
A estrutura de programação é modular. Os módulos são implementados em linguagem C, com as partes que necessitam de execução em tempo real em Assembly.

A estrutura dos diversos módulos nos dois níveis de controle propostos é mostrada na figura 5.1. Essas figuras mostram as funções a serem implementadas numa primeira fase do sistema. Cada função pode ser desenvolvida separadamente em relação às demais, visto que as interfaces são padronizadas.

A padronização das interfaces dos diversos módulos permite tornar o sistema aberto para qualquer novo tipo de aplicação que venha a ser desenvolvido no futuro, como também para um eventual *upgrading* dos módulos já desenvolvidos.



(a) Unidade de Controle de Junta



(b) Unidade de Central de Supervisão - UCS

Figura 5.1 : Estrutura de programação

Para a Unidade Central de Supervisão (UCS), um número maior de módulos foi especificado, visto a maior complexidade de funções necessárias para o seu desempenho.

A seguir são descritos estes módulos, juntamente com suas interfaces de comunicação.

5.1.1 Programa Monitor

O programa monitor tem por objetivo prover a ativação dos diversors módulos e possibilitar a interconexão entre os módulos funcionais para a execução das tarefas especificadas.

A estrutura de comunicação utilizada no sistema, tanto entre as unidades processadoras como entre módulos, é baseada na troca de mensagens. Por facilidade de implementação, o programa monitor foi definido para trabalhar por máquina de estado, onde a cada mensagem gerada ou recebida por um módulo deve-se gerar uma interrupção especificada, de modo a fazer com que uma sequência de atividade previamente definida se processe.

Estas mensagens são padronizadas, definindo-se número de bytes, campo de endereço, campo de controle, campo de dados, bem como a especificação dos possíveis valores desses campos.

A função do monitor é, uma vez gerada ou recebida determinada mensagem, analisar seu conteúdo e tomar as decisões necessárias a seu encaminhamento, que por exemplo, pode ser a ativação de algum outro módulo da mesma unidade ou ser a transmissão de uma mensagem à outra unidade.

5.1.2 Driver de comunicação UCS-UCJ

A função deste módulo é realizar a interface entre a UCS e uma UCJ. Esta interface deve operar através de um protocolo assíncrono por interrupção byte a byte, como descrito no capítulo 4.

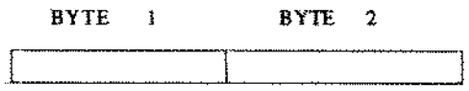
Para padronização de um protocolo de sinais *hardware*, que permita uma melhor performance da interface sem sobrecarregar o processador, a seguinte convenção é estabelecida:

Uma transição no sinal *Request To Send* (RTS) [25] da estação emissora indica que esta terminou a transmissão de uma determinada mensagem. Esta transição deve gerar uma interrupção de mudança de *status* no controlador serial da unidade receptora, visto este sinal estar no painel traseiro do sub-bastidor do sistema. Deste modo, uma transição no sinal RTS do emissor indica à unidade receptora que o programa monitor pode analisar os campos da mensagem recebida.

As mensagens que trafegam nesta interface podem ser dos seguintes tipos:

i) Mensagem de dados

Quando do envio de um arquivo de trajetória da UCS para as UCJ, os diversos pontos devem ser enviados um por vez para cada UCJ, de modo a possibilitar o sincronismo do movimento do manipulador. A figura 5.2 apresenta a estrutura da mensagem de dados.



ONDE:

BYTE 1 : INDICADOR DE MENSAGEM DE DADOS

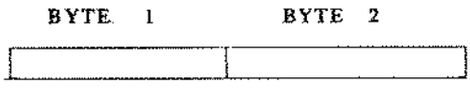
BYTE 2 : DADO

Figura 5.2 : Estrutura de mensagem de dados

ii) Mensagem de supervisão

Vários tipos de mensagens de supervisão podem trafegar na interface UCS-UCJ. No entanto, uma estrutura única de mensagem deve ser definida de modo a permitir uma análise uniforme e rápida no processamento de tais mensagens.

A estrutura do quadro de supervisão é mostrada na figura 5.3.



ONDE:

BYTE 1 : INDICADOR DE MENSAGEM DE SUPERVISÃO

BYTE 2 : TIPO DE MENSAGEM DE SUPERVISÃO

Figura 5.3 : Estrutura da mensagem de supervisão

A seguir descreve-se os vários tipos de mensagens de supervisão e sua condição de uso:

• Mensagem de início de trajetória

Mensagem originada na UCS, usada para indicar às unidades controladoras o início de um ciclo de transmissão de dados.

• Mensagem de solicitação à UCJ de posição atual

Mensagem originada na UCS, quando da solicitação às unidades controladoras para fornecimento da atual leitura de seus encoders.

• Mensagem de indicação de junta no estado inicial

Mensagem originada na UCJ para sinalizar à UCS que a unidade recebeu uma interrupção originada pelo pulso de referência do encoder, indicando que a junta controlada atingiu a posição de referência.

• Mensagem de alinhamento

Solicitação da UCS às unidades controladoras para que estas se preparem para o início da sequência de alinhamento para posição inicial.

• Mensagem de unidade operacional

Enviada pela UCJ ou UCS para indicação de unidade pronta para operação.

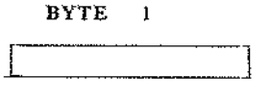
iii) Mensagem de alarme

Apenas um tipo de condição de alarme foi especificado. Este alarme é de caráter urgente, devendo ter máxima prioridade em seu tratamento.

• Mensagem de HALT

Mensagem de alarme que trafega nos dois sentidos, com o intuito de parar imediatamente o processamento efetuado.

A estrutura do quadro é mostrada na figura 5.4 :



ONDE:

BYTE 1 : INDICADOR DE HALT

Figura 5.4 : Estrutura de mensagem de alarme

5.1.3 Driver de comunicação UCS-PC

Este módulo tem por objetivo o gerenciamento da interface entre a Unidade Central de Supervisão e o nível 3 de controle, que é implementado em um microcomputador PC.

Sua função é receber e transmitir as mensagens provenientes do nível 3 de controle. O módulo Monitor examina as mensagens e as distribui para os outros módulos.

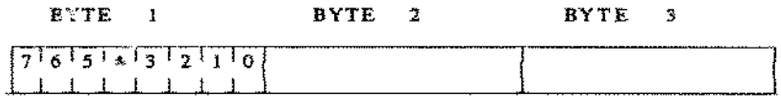
O tipo de transmissão utilizado é o assíncrono em modo bidirecional por interrupção byte a byte. Numa primeira versão, o protocolo será implementado com janela 1, isto é, cada mensagem enviada necessitará de uma mensagem de confirmação, antes de qualquer outra mensagem ser enviada.

Este método foi inicialmente escolhido pois não necessita do gerenciamento de nenhum *buffer* auxiliar de memória, o que é necessário quando várias mensagens necessitam ser armazenadas para possível retransmissão, nem da análise de campos de numeração das mensagens, como é o caso quando da utilização de protocolos de janela N.

Uma desvantagem do uso deste tipo de protocolo é o *overhead* de mensagens de controle que trafegam na interface, porém caso no futuro se deseje aumentar a performance desta interface, outros tipos de protocolos poderão ser empregados, possivelmente com a utilização de protocolos síncronos com taxas de transmissão mais elevadas.

Na interface com o microcomputador, não podemos contar com sinais de controle da interface RS232, tais como *Clear to Send* e *Request to Send*, de modo que a confirmação de mensagens será via software, com uma mensagem específica para tal finalidade, obrigando também a mensagem da estação transmissora conter o número total de bytes que a compõe.

A estrutura do quadro desta interface é mostrada na figura 5.5.



ONDE :

BYTE 1 :

BITS 7 a 2 : NUMERO DO MODULO PARA QUAL A MENSAGEM E DIRIGIDA

BITS 1 a 0 : NUMERO TOTAL DE BYTES DA MENSAGEM

BYTE 2 : TIPO DE MENSAGEM

BYTE 3 : BYTE OPCIONAL DE DADO, DEPENDENTE DO TIPO DE MENSAGEM

Figura 5.5 : Estrutura da mensagem da interface UCS-PC

No byte 1, a codificação do endereçamento do módulo deve estar conforme especificado na tabela 5.1 :

MÓDULO	NÚMERO
MONITOR	01
DRIVER PC-UCS	02
MODO DE OPERAÇÃO	03
GERAÇÃO DE TRAJETÓRIA	04
DRIVER UCS-UCJ	05
SINCRONISMO	06
POSIÇÃO INICIAL	07
ALARMES	08
DADOS DE SENSOR DE ESFORÇO	09

Tabela 5.1 : Codificação dos módulos na interface UCS-PC

5.1.4 Módulo de posição inicial

A função deste módulo é coordenar a execução da tarefa de posicionar todos os graus de liberdade do robô numa posição conhecida que denominaremos de posição de repouso.

A figura 5.6 indica a sequência de eventos que deve ocorrer na UCS para que a tarefa seja realizada:

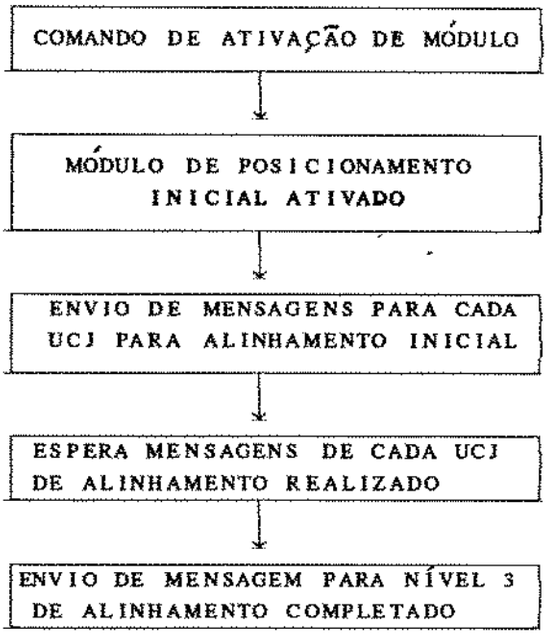


Figura 5.6 : Sequência de eventos para alinhamento inicial

Temos então os seguintes tipos de mensagens definidos neste módulo:

i) mensagem de pedido de alinhamento inicial

Enviada pelo nível 3 para Módulo de Posição Inicial iniciar procedimentos.

ii) mensagem de alinhamento inicial completado

Enviada pelo Módulo de Posição Inicial para o nível 3 notificando que o procedimento foi realizado com êxito.

5.1.5 Módulo de geração de trajetória

Este módulo tem por função gerar uma trajetória para cada grau de liberdade do manipulador, a partir de mensagens provenientes do nível 3 de controle, as quais especificam a posição inicial e final do efetuador em um dado movimento.

Quando este módulo é ativado, a posição atual do efetuador do robô geralmente não coincide com a posição inicial especificada pela mensagem oriunda do nível 3. Então, o procedimento é, uma vez conhecida a posição atual do mesmo, atingir-se o ponto inicial especificado através da utilização do modelo direto, e somente a partir deste ponto utiliza-se o modelo cinemático inverso. A sequência de eventos que ocorrem neste módulo está descrita na figura 5.7.

Descreve-se a seguir os tipos de mensagens definidas para a interface entre o nível 3 e o módulo de geração de trajetórias:

- mensagem de pedido de ativação de módulo

Enviada pelo nível 3 para o Módulo de Geração de Trajetória para que este inicie procedimentos.

- mensagem de envio de ponto inicial

Enviada pelo nível 3 para o Módulo de Geração de Trajetória contendo o ponto inicial especificado da trajetória.

- mensagem de envio de ponto final

Enviada pelo nível 3, contendo o ponto final da trajetória especificada.

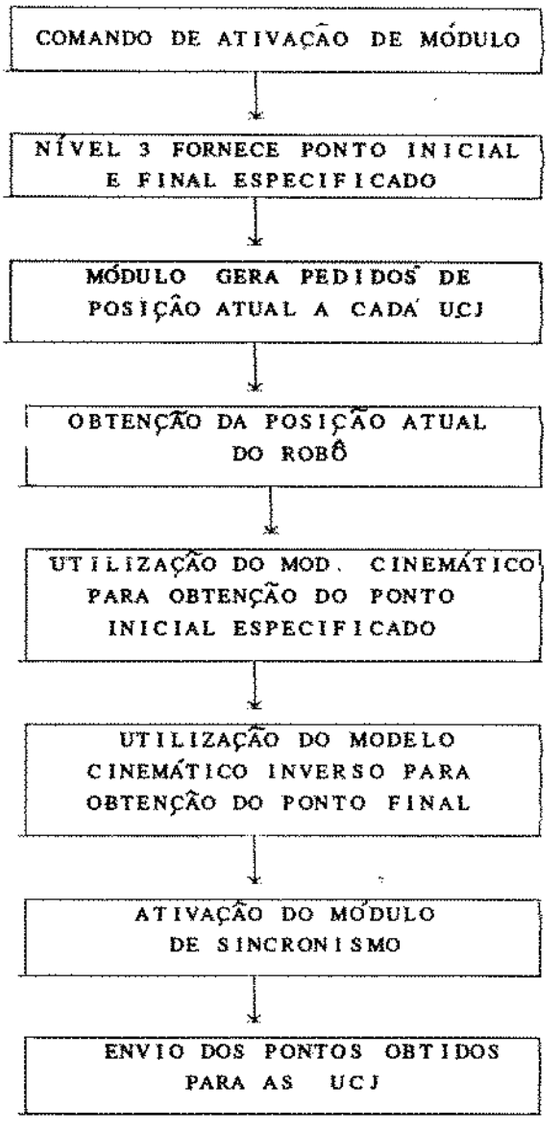


Figura 5.7: Sequência de eventos para geração de trajetórias

5.1.6 Módulo de sincronismo

Este módulo *a priori* não se comunica diretamente com o nível 3, embora na estrutura geral de comunicação isto possível. No entanto, o módulo de geração de trajetórias se utiliza dos serviços prestados por este módulo.

A função deste módulo é garantir o sincronismo entre os pontos de trajetória enviados pela UCS às unidades UCJ. Isto é feito da seguinte forma:

A cada iteração do algoritmo executado pelo módulo de geração de trajetória, tem-se como resultado um vetor de deslocamento $\Delta\theta$, que é dado pela expressão 5.1.

$$\Delta\theta = [\ \Delta\theta_1,\ \Delta\theta_2 \Delta\theta_3,\ \ldots\ ,\ \Delta\theta_n] \qquad (5.1)$$

onde $\Delta\theta_n$ = deslocamento angular da junta n.

Considerando $\Delta\theta_{max}$ o deslocamento angular máximo permitido ao sistema, conhecido o vetor $\Delta\theta$, obtém-se o maior valor de $\Delta\theta_n$ e divide-se por $\Delta\theta_{max}$. O número inteiro mais próximo maior que o valor obtido será então o número de amostras pelo qual todos os valores de $\Delta\theta_n$ serão divididos. Todos os valores obtidos são assim enviados às unidades UCJ sincronizadamente, garantindo-se um número igual de amostras para cada junta do robô.

5.2 CONCLUSÕES

Outros módulos podem ser especificados para o sistema, mas estes inicialmente compõem um conjunto mínimo para que o sistema possa operar e os testes iniciais possam ser realizados.

No próximo capítulo é feita uma análise de desempenho do sistema, com o objetivo de validação da arquitetura proposta e identificação dos pontos críticos relativos à performance do sistema.

CONFIGURAÇAO INICIAL											
POSITION (mm)			ORIENT. (deg)			coordenadas articulares (					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
500	0	1495	0	0	0	0	90	-90	0	0	0

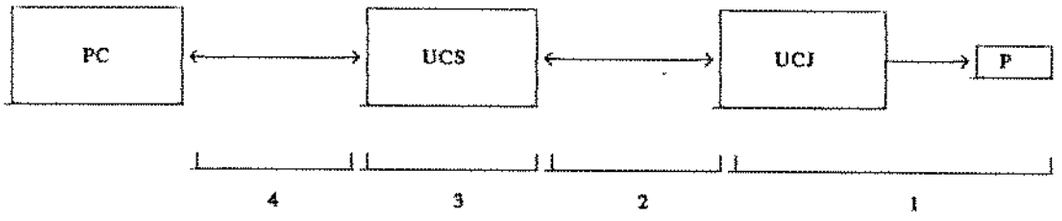
FINAL CONFIGURATION											
POSITION (mm)			ORIENT (deg)			JOINT COORDINATES (deg)					
X	Y	Z	ψ	θ	ϕ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
100	100	100	0	0	0	47	175	-48	-9	379	-97

CAPÍTULO 6 : ANÁLISE DE DESEMPENHO E CONCLUSÕES

Neste capítulo é discutido o desempenho esperado das interfaces de comunicação e das unidades processadoras que compõem o sistema. Faz-se uso aqui de dados práticos e teóricos de performance de processadores para se estimar o desempenho das unidades UCJ e UCS. O objetivo da análise, além de detectar os "gargalos" do sistema, é possibilitar a identificação das prioridades sobre os pontos que a curto, médio e longo prazo poderão ser reprojetados, e as relações que estes reprojetos acarretarão na performance total do equipamento.

6.1 ANÁLISE DE DESEMPENHO

A figura 6.1 mostra o fluxo de informação no sistema e indica os trechos da estrutura que serão analisados:



onde :

1 - PROCESSAMENTO DA UNIDADE UCJ

2 - INTERFACE SERIAL TTL ASSÍNCRONA

3 - PROCESSAMENTO DA UNIDADE UCS

4 - INTERFACE SERIAL RS-232 ASSÍNCRONA

Figura 6.1 : Trechos de análise do sistema

6.1.1 Processamento da UCJ

Para análise da UCJ, primeiramente faremos um cálculo da quantidade de interrupções por unidade de tempo existentes na unidade.

São duas as fontes de interrupção que em regime exigirão da UCJ um tempo significativo de processamento:

i) Interrupções do relógio de tempo real

ii) Interrupções de troca de mensagens com a UCS

O processador deve ter tempo suficiente para transmitir/receber mensagens na linha serial com a UCJ, e de realizar o tratamento das informações oriundas das leituras do contador de impulsões do encoder nas interrupções de tempo real .

i) Interrupções do relógio de tempo real

Tomando como 20 KHz a frequência máxima de impulsõe provenientes do encoder, obtemos a taxa de uma impulsão a cada 50 µs.

A capacidade da palavra do contador projetado na UCJ é de 8 bits, podendo portanto contar até 256 impulsões . Numa análise de pior caso, o tempo máximo em que o processador teria para ler o contador, sob pena de, se não o fizer, perder todas as 256 impulsões, é de 256 x 0.05 ms, que resulta em 12.8 ms .

Se a frequência para a geração de interrupção de tempo real ser de 1 KHz, isto é, a cada 1 ms o processador da UCJ é interrompido para realizar uma leitura do contador de impulsões do encoder, obtemos uma leitura máxima da ordem de 256/12.8, que resulta em aproximadamente 20 impulsões, resultado que nos permite afirmar que a CPU pode realizar um monitoramento com folga da contagem. Porém uma frequência muito alta de amostragem pode onerar muito o processamento da CPU.

Sistemas clássicos de controle operam com taxa de amostragem de aproximadamente 4 ms, tendo portanto na leitura do contador uma contagem máxima da ordem de 80 impulsões.

Sendo o tempo médio de execução de uma instrução no microprocessador V.20 @ 8MHz da ordem de 2 µs, temos que durante um intervalo de 4 ms o processador executará aproximadamente 2000 instruções, o que é bastante razoável para o processamento de algorítmos relativamente simples de controle.

ii) Interrupções de troca de mensagens com a UCS

O número de interrupções oriundas da linha serial nesta análise é desprezível, pois em regime apenas uma mensagem será recebida e nenhuma será transmitida durante o intervalo de 4 ms.

6.1.2 Vazão da linha serial UCJ-UCS

Quando do envio de dados da trajetória pela UCS a cada unidade UCJ, é esperado que a grande maioria do volume de mensagens trafegadas pela linha serial seja no sentido da UCS para a UCJ, pois a UCJ só enviará mensagens de supervisão para o nível 2 de controle.

Supõe-se então que durante um intervalo de tempo T, o volume de mensagens de dados trafegados numa linha serial UCS-UCJ seja de T/4 [ms], visto que 4 ms é a taxa com que a UCJ enviará as mensagens de dados de trajetória.

Como foi definido no capítulo 4, as mensagens de dados de trajetória possuem 3 bytes. Com estes dados pode-se estimar a taxa mínima necessária para que a interface serial suporte a execução de tal aplicação.

Este cálculo é executado observando-se que, se em 4 milisegundos um volume de 24 bits é escoado, então em 1 segundo o volume será de 6 Kbits, resultando numa taxa de transmissão mínima a ser suportada pela interface da ordem de 6 Kbit/s.

Mesmo considerando que a interface utilizada é assíncrona, podemos avaliar esta taxa de transmissão como sendo baixa. Numa primeira fase, a taxa de transmissão na linha serial será de 20 Kbit/s, suportando portanto um tráfego de até 80 bits num intervalo de 4 milisegundos, que correspondem a aproximadamente 3 mensagens em cada sentido.

6.1.3 Processamento da UCS

Para a análise de desempenho da UCS devemos supor dois casos de operação hipotéticos. No primeiro caso a função de geração de trajetória é realizada na própria UCS, sendo somente os pontos inicial e final passados como parâmetros pelo nível 3 de controle. No segundo caso analisa-se a situação onde todo o arquivo de trajetória é gerado no nível 3 e transmitido *on-line* para as UCJ via UCS.

• Trajetória gerada na UCS

De acordo com as taxas de mensagens já discutidas nos ítens anteriores e analisando a figura 6.2 podemos afirmar que:

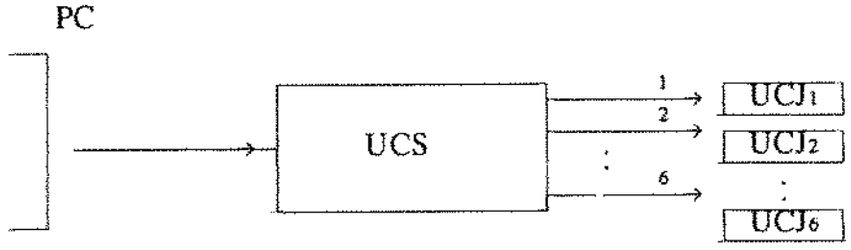


Figura 6.2 : Análise de desempenho da UCS

O fluxo de mensagens num intervalo de tempo de 4 ms que entram e saem da UCS pode ser considerado como sendo de :

- 1 mensagem enviada para cada UCJ

- nenhuma mensagem proveniente do nível 3 para a UCS

Portanto o fluxo total numa configuração máxima do sistema é de 6 mensagens a cada 4 ms.

Como o comprimento total de cada mensagem é de 3 bytes e a transmissão é feita por interrupção, temos neste intervalo de tempo um total de 18 interrupções a cada 4 ms. Supondo mais 2 interrupções oriundas de outras fontes do sistema neste intervalo, temos um total de 20. Isto resulta num tempo médio de 200 µs por interrupção.

Supondo 2 µs o tempo médio de execução de uma instrução no microprocessador V20 @ 8 Mhz, temos aproximadamente 100 instruções por interrupção.

Neste caso pode-se perceber que o processador encontra-se bastante ocupado e que preocupações a nível de otimização de código devem ser tomadas.

• Trajetória gerada *on-line* no PC

Neste caso, teríamos todos os pontos da trajetória sendo enviados para a UCS e esta enviando os para as UCJ durante o movimento do robô. Certamente esta situação é a que mais exige do sistema em termos de processamento. Uma visão geral do fluxo de mensagens durante um período de amostragem de 4 ms é mostrado na figura 6.3

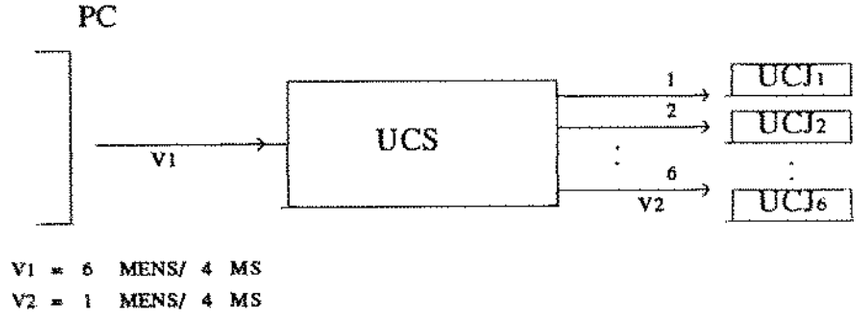


Figura 6.3 : Trajetória gerada na UCS

Calculando a taxa de transmissão necessária na linha serial para comunicação com o PC, teríamos 6 x 3 = 18 bytes num intervalo de 4 ms, isto é, 144 bits/4 ms. Disto resulta uma taxa mínima de 36 Kbit/s, que está fora da especificação da norma RS-232, cuja taxa de transmissão máxima é de 20 Kbit/s, embora na prática se utilize taxas maiores que este valor nesta interface.

Analisando a capacidade de processamento da UCS neste caso, temos da figura 6.3 que o número total de mensagens a serem tratadas num período de 4 ms durante esta aplicação é de 12.

O número de 12 interrupções a cada 4 ms resulta em 36 interrupções neste mesmo intervalo de tempo, visto que cada mensagem contém 3 bytes. Portanto temos uma taxa de 9 interrupções por milisegundo. Supondo o tempo de execução de uma instrução de 2 microsegundos, temos:

$$\text{Número de intruções / interrupção} = \frac{500}{9} = 55$$

Esta análise não nos permite afirmar que este tipo de aplicação é viável com a atual estrutura hardware do sistema, visto o processsamento da UCS ficar extremamente carregado pelo grande número de interrupções a ser processado. Testes práticos de sistema deverão ser efetuados para a verificação de tais valores [24].

6.2 PERSPECTIVAS DE EVOLUÇÃO

a) Capacidade de processamento UCS/UCJ

Do ponto de vista de evolução de performance dessas unidades, o primeiro passo seria o aumento do *clock* do processador. Versões de V20 a velocidades maiores já se encontram comercialmente disponíveis. Este tipo de modificação é simples e tem um impacto muito pequeno no reprojeto das unidades.

Uma outra modificação um pouco maior seria a troca do microprocessador. Citando como exemplo, a substituição do microprocessador V20 por um 80286 @ 16 Mhz daria um ganho mínimo de 4 vezes na capacidade de processamento do sistema.

b) Interfaces seriais

A atual interface serial com o PC é RS-232. Esta interface por não ser balanceada é bastante limitada na sua capacidade máxima de transmissão. Caso alguma aplicação futura necessite de transmissão de dados *on-line* às UCJ, esta interface será, juntamente com a UCS, um possível gargalo do sistema. Neste caso, uma evolução natural seria transformar esta interface numa interface balanceada padrão V.11 [26] , onde taxas de transmissão de 64 Kbit/s são possíveis.

Já as interfaces seriais UCS-UCJ são de nível elétrico TTL e podem ser transformadas em interfaces síncronas, onde além dos sinais de dados, envia-se também um relógio de transmissão em uma via separada. Para a utilização desta interface, que permite altas taxas de transmissão, seria necessário o reprojeto tanto da UCS como da UCJ, para o acréscimo de um Controlador de Acesso Direto à Memória (DMAC), que torna possível a liberação da CPU do trabalho de buscar/enviar dados através das linhas seriais.

Na análise em regime, desprezando-se o tráfego de mensagens de supervisão, a performance esperada do sistema mostrou-se boa em relação aos requisitos exigidos. Apenas no caso crítico do nível 3 enviar mensagens de informação de trajetória *on-line* às UCJ é que a capacidade de processamento na UCS se mostra insuficiente.

Com estes resultados, vê-se claramente que com aplicações mais sofisticadas, o "gargalo" natural do sistema será o volume de processamento na UCS, vindo as linha de comunicação num segundo plano e posteriormente as UCJ.

6.3 COMENTÁRIOS E CONCLUSÕES

O objetivo principal deste trabalho foi a especificação de um supervisor de controle para um robô industrial. As duas grandes tarefas básicas nas quais este trabalho pode ser resumido são a definição de uma arquitetura de hardware com uma performance compatível com as funções inicialmente especificadas e a definição da estrutura de controle a ser utilizada para a realização das funções de geração de trajetória autônoma e controle no espaço cartesiano.

Para as funções acima mencionadas serem realizadas, foi necessário um estudo sobre os diversos tipos de modelagem utilizadas para o controle de manipuladores, com o enfoque na modelagem cinemática inversa. A diferença entre os vários algoritmos de modelagem cinemática inversa situa-se no modo como a matriz Jacobiana é invertida, e é esse fator que fornece ao algoritmo as características necessárias para que este seja utilizado numa determinada aplicação.

Portanto, características como número de operações realizadas (que influencia diretamente no tempo de inversão) e número de pontos intermediários gerados num determinado movimento, foram estudados nos vários métodos de inversão para a escolha do algoritmo a ser implementado na Unidade Central de Supervisão (UCS). Dentre estes métodos, o escolhido foi o de Miss, que se caracteriza pela simplicidade de implementação, execução de poucas operações matemáticas e a geração de pontos intermediários que possibilitam a realização de uma trajetória suave, que para a aplicação de controle no espaço cartesiano é fundamental.

O sistema especificado está em fase de implementação e faz parte do trabalho de tese de Carlos Henrique Dias [24]. A unidade UCJ já foi implementada e está em fase de integração com a parte de potência do robô. A unidade UCS está em fase final de construção em *wire-wrap*, devendo ocorrer, logo após seu término, uma fase de integração entre ambas.

Um dos problemas mais delicados da UCS foi a definição de uma plataforma de software a ser utilizada. Logo no início do trabalho constatou-se a necessidade de uma linguagem de alto nível para a elaboração dos algoritmos de inversão da UCS, sendo que esta linguagem deveria residir num ambiente sem o suporte de um sistema operacional, já que a UCS é implementada num microprocessador dedicado. Após o estudo de algumas alternativas de definição deste ambiente, optou-se por desenvolver a estrutura de software em linguagem C, utilizando-se de uma ferramenta que gera automaticamente um código C "romável", isto é, um código que, após o programa ter sido desenvolvido, possa ser diretamente gravado em memória PROM.

Os módulos de software já desenvolvidos são os módulos de geração de trajetória e os *drivers* de comunicação, sendo que os restantes ainda necessitam ser elaborados.

A intenção agora é, como parte da tese de Carlos Henrique Dias [24], testar as funcionalidades mínimas do sistema num robô didático de três graus de liberdade, especialmente projetado para esta finalidade e que já está concluído, e a seguir realizar todos os testes de performance do equipamento neste mesmo ambiente.

Como consequências deste trabalho, além da formação de pessoal qualificado no assunto e da disponibilidade de um sistema que pode atuar como plataforma para várias outras aplicações, é importante também ressaltar a utilidade de alguns resultados obtidos, tais como o estudo e definição dos algoritmos de modelagem cinemática inversa para aplicação de controle no espaço cartesiano, para o projeto de pesquisa que se desenvolve entre a UNICAMP, a Petrobrás e o GKSS da Alemanha. Estes resultados certamente serão utilizados de maneira que alguns problemas que aqui ocorreram possam ser evitados, possibilitando assim um aumento de produtividade nas atividades realizadas.

BIBLIOGRAFIA

[1] Dias, Marcus de Aguiar; "Controlador Programável a Multimicroprocessadores para Controle Hierárquico de Robôs", Tese de Mestrado, Faculdade de Engenharia Elétrica, UNICAMP, (1991).

[2] Klafter R. D., Chmielewski T. A, Negin M.; "Robotic Engineering an Integrated Approach", Prentice Hall, Englewood Cliffs, (1989).

[3] Zheng Y. F., Hemani H. ; "Computation of Multibody System Dynamics by a Multiprocessor scheme", IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, N. 1, pp. 102-110, (1986).

[4] Bestaqui Y. D. M.; "Descentralised PD and PID Robotic Regulators", IEEE Proceedings, vol 136, N.4, pp. 133-145, (1989).

[5] Denavit, J., Hartemberg, R.; "A Kinematic notation for lower-pair mechanisms based on matrices", ASME J. on Applied Mech., (1955).

[6] Miranda, M. F.; "Controle de um servomecanismo por um microcomputador dedicado: Uma contribuição ao estudo de controladores para robôs industriais", Tese de Mestrado, Faculdade de Engenharia Mecânica, UNICAMP, (1992).

[7] Krishnamurthy, E. V., De Vel, O. Y.; "An iterative pipelined array architecture for the generalized matrix inversion", Information Processing Letters 26, pp. 263-267, (1988).

[8] Miss, R. W., Schultheiss, G. F.; "The design of an algorithm for the control of the redundant kinematic chains", 5th ICAR - Pisa, (1992).

[9] Rosário, J. M.; "Relatório Técnico de Pesquisa para FAPESP : Modelagem geométrica e cinemática inversa", Faculdade de Engenharia Mecânica, UNICAMP, (1992).

[10] Marshall C. Pease, III; "Methods of Matrix Algebra" ; Stanford Research Institute, (1965).

[11] Seraji, H. et all : "Two-link robot", jor. of Robotic Systems 3(4), pp. 349-365, (1986).

[12] T.N.E Greville. The pseudo-inverse of a rectangular or singular matrix and its application to the solution of systems of linear equations, SIAM Review 1, pp. 38-43, (1959).

[13] Rosário, J.M.; Weber, H.I.; Morooka, C.: "Development and Control of Advanced Robots for Underwater Tasks, 5º ICAR - Int. Conf. on Advanced Robotics, PISA, Italia, (1991).

[14] Rosário, J.M.; Weber, H.I.; Aust, E. ; Schultheiss, G.F. : " A Study of Advanced Robots for Underwater Tasks", VI Brasil Offshore, Rio de Janeiro, (1991).

[15] Rosário, J.M. "A systematic . methode the kinematic analysis of industrial robots and the MANUTEC robot r-15", GKSS, Alemanha, (1990).

[16] Zampieri, D.E.; Rosário, J.M.; Martins, J.L.F.; Saramago, M.A.P.: "Projeto e Modelagem de um Manipulador", Congresso de Engenharia Mecânica Norte-Nordeste, Natal, Rio Grande do Norte, (1991).

[17] Rosário, J.M.: "Feasibility Study of Non-Linear Controller for Robotic Manipulators, IV DINAME, Pouso Alegre, Minas Gerais, (1991).

[18] Rosario, J.M.; Davy, R.; Leclerc, O.: "Programme d'Acquisition et de Traitement du Mouvement d'un Robot", ISMCM, Paris, (1989).

[19] Pruski, Alain; Robotique Générale,. Ed. Ellipses, (1988).

[20] Rosário, J.M.; Verdin, L.; "Etude de faisabilité d'un asservissement de position par commande tout-ou-rien", ISMCM, Paris, (1987).

[21] Shimano, B.; Roth, B.; "On force sensing information and its use in controlling manipulators", Proceedings of the Eight Industrial Symposium on Industrial Robots, pp. 119-126.

[22] Craig, John J.; Introduction to Robotics Mechanics & Controls, (1986).

[23] Rosario, J.M.: "Analyse du comportement dynamique d'un robot industriel partir de ses capteurs proprioceptifs", ISMCM, Paris, (1986).

[24] Dias, Carlos Henrique; Realização Experimental de um Supervisor de Controle de um Robô Industrial", Tese de Mestrado a ser apresentada na FEM, UNICAMP, (1993).

[25] V.24, International Standard; CCITT, Blue Book, (1988).

[26] V.11, International Standard; CCITT, Blue Book, (1988).

ANEXO A

OBTENÇÃO DA MATRIZ JACOBIANA

Este anexo pretende abordar os procedimentos necessários para a obtenção da matriz Jacobiana de um determinado robô. Como já anteriormente descrito, a matriz Jacobiana pode ser vista como a função de transferência entre as coordenadas cartesianas da ferramenta e as coordenadas articulares dos *n* graus de liberdade de um robô.

A mudança de coordenadas de um robô com *n* graus de liberdade se faz da seguinte maneira:

Partindo de uma configuração principal do robô na qual as variáveis articulares θ_0 são dadas, o elemento terminal (ferramenta) é posicionado num determinado ponto X_0 do volume de trabalho.

A relação de mudança de coordenadas consistirá numa função de transferência que descreverá a correspondência que existe entre um conjunto de variáveis articulares dada por θ e uma posição cartesiana X qualquer. Sob forma vetorial, a transposição direta de coordenadas se escreverá:

$$X - X_0 = F(\theta - \theta_0) \qquad (A.1)$$

onde o vetor F possui *n* componentes descrevendo a posição e a orientação do efetuador.

Para a transposição inversa, uma posição X do volume de trabalho será alcançado pelo robô a partir da posição de repouso X_0, desejando-se conhecer os valores das variáveis articulares que correspondem ao ponto final desejado. Esta relação pode ser expressa por:

$$\theta - \theta_0 = F^{-1}(X - X_0) \qquad (A.2)$$

No caso geral, esta equação não terá solução única. Mas a "melhor" solução poderá ser encontrada e utilizada para o comando cinemático de mecanismos de controle. Esta transposição de coordenadas é de fundamental importância, visto ser necessário o uso de coordenadas articulares para os comandos dos elementos de controle das juntas de um robô.

Para resolver as dificuldades geradas pela transposição de coordenadas inversa descrita em A.2, um procedimento sistemático de cálculo é utilizado. Esse método de transposição é baseado na geração de uma matriz Jacobiana inversa.

Considere uma configuração principal θ_0, X_0 de um robô. Para "pequenos deslocamentos" δx correspondentes a deslocamentos das variáveis $\delta\theta$ podemos escrever:

$$\delta \bar{X}_{mx1} = [\ \delta F / \delta \theta\ ]_{mxn} \cdot \delta\bar{\theta}_{nx1} \qquad (A.3)$$

A notação m x 1, m x n e n x 1 indica as dimensões das matrizes . Define-se por *n* o número de graus de liberdade do robô, e *m* o número variáveis necessárias para se definir a posição/orientação do efetuador no volume de trabalho do robô.

A matriz Jacobiana J(θ) será definida por:

$$\left[J(\theta)_{ij} \right] = \left[\delta F_i / \delta \theta_j \right] \qquad \text{(A.4)}$$

Esta matriz poderá ser "montada" a partir das relações cinemáticas que descrevem a arquitetura do robô, onde do modelo geométrico temos:

$$\begin{aligned} X_1 &= F_1 (\theta_1, \theta_2, \ldots \theta_n) \\ X_2 &= F_2 (\theta_1, \theta_2, \ldots \theta_n) \\ &\vdots \\ X_m &= F_m (\theta_1, \theta_2, \ldots \theta_n) \end{aligned} \qquad \text{(A.5)}$$

Do modelo cinemático, tenta-se obter relações entre pequenas variações na posição do efetuador e as variações produzidas nas coordenadas articulares e vice-versa. Estas relações podem ser obtidas a partir de diferenciações parciais das equações A.5. Portanto a matriz Jacobiana será escrita como:

$$J(\theta) = \begin{bmatrix} \delta F_1 / \delta \theta_1 & \delta F_1 / \delta \theta_2 & \ldots & \delta F_1 / \delta \theta_n \\ \delta F_2 / \delta \theta_1 & \delta F_2 / \delta \theta_2 & \ldots & \delta F_2 / \delta \theta_n \\ \vdots & \vdots & & \vdots \\ \delta F_m / \delta \theta_1 & \delta F_m / \delta \theta_2 & \ldots & \delta F_m / \delta \theta_n \end{bmatrix} \qquad \text{(A.6)}$$

- Inversão da matriz Jacobiana

Como os sinais de comando de um manipulador são enviados diretamente aos acionadores, e os mesmos operam sobre variáveis articulares, para controlar um robô no espaço das tarefas, será necessário realizar uma transposição de coordenadas. Esta transposição será escrita como:

$$\delta\theta = J(\theta)^{-1}.\ \delta x \qquad (A.7)$$

J^{-1}: Matriz Jacobiana inversa (desde que exista)

A relação (A.7) indica a variação de $\delta\theta$ (variáveis articulares) que realiza um deslocamento dado por δx do elemento terminal do robô.

Como a relação A.7 é válida apenas para pequenos deslocamentos, pode-se obter os deslocamentos totais desejados através de algoritmos que implementem soluções iterativas e que, calculando $J(\theta)$ à cada passo, gerem uma trajetória X(t) que poderá ser descrita em função da variação dos ângulos das juntas do robô.

ANEXO B :

SÍNTESE DO CIRCUITO DETECTOR DE VELOCIDADE E SENTIDO

Neste anexo apresenta-se uma análise do procedimento de obtenção de um sinal de frequência quatro vezes superior à frequência de entrada de dois sinais provenientes do encoder de cada junta do manipulador. Além da multiplicação da frequência, obtém-se também a informação do sentido de rotação da junta, através da análise da fase dos sinais.

O objetivo é a síntese de um circuito digital para ser implementado em um dispositivo lógico programável, com vistas à sua utilização nas unidades UCJ.

A figura B.1 mostra os sinais provenientes do encoder e o sinal de saída desejado. Geralmente um encoder fornece dois sinais (pistas A e B) periódicos e simétricos para uma velocidade de rotação constante, defasadas de um tempo T/4, onde T é o período dos sinais.

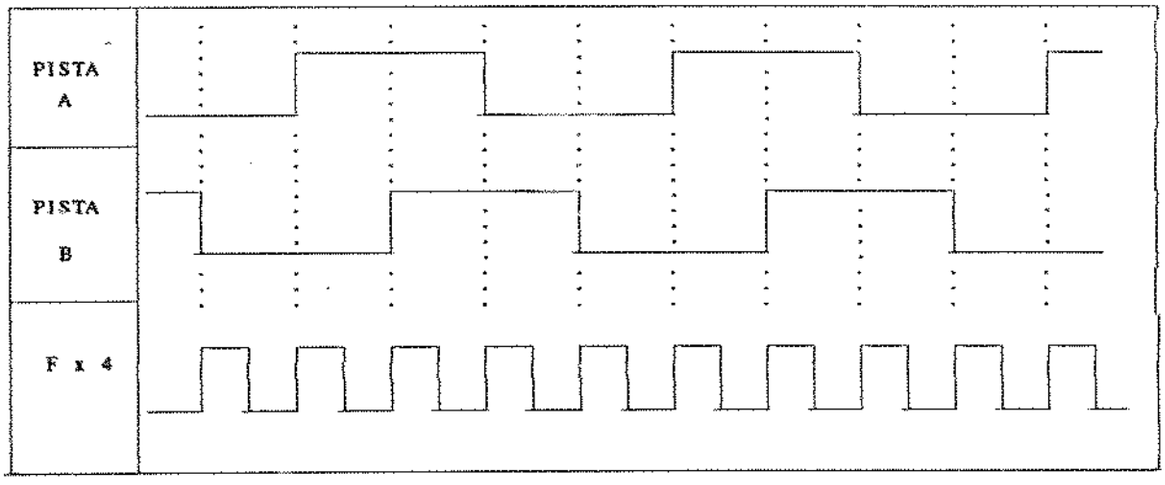


Figura B.1 : Sinais de entrada e saída do dispositivo

Para a realização da função de multiplicação por quatro e da função de detecção de sentido, a figura B.2 mostra uma outra forma de obtenção desta função, através da geração de dois sinais denominados I^+ e I^-, onde o sinal I^+, por convenção, é ativado quando a pista B está adiantada em relação à pista A e o sinal I^- é ativado quando se tem a situação inversa. Estes sinais, como pode ser observado, possuem frequência quatro vezes superior à frequência dos sinais de entrada.

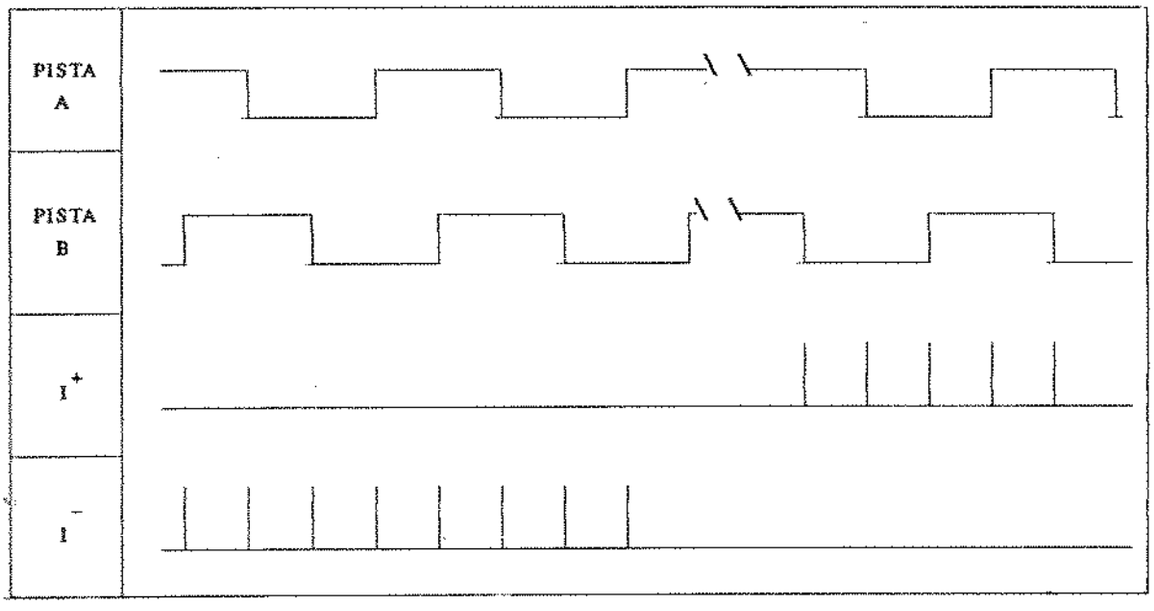


Figura B.2 : Informações das bordas dos sinais

Logicamente, os sinais I^+ e I^- podem ser descritos pelas seguintes equações:

$$I^+ = (\bar{B} \cdot dA^+) + (A \cdot dB^+) + (B \cdot dA^-) + (\bar{A} \cdot dB^-) \qquad (B.1)$$

$$I^- = (\bar{B} \cdot dA^-) + (A \cdot dB^-) + (B \cdot dA^+) + (\bar{A} \cdot dB^+) \qquad (B.2)$$

onde I^- = velocidade no sentido negativo

I^+ = velocidade no sentido positivo

dA^+ = borda de subida da pista A

dA^- = borda de descida da pista A

dB^+ = borda de subida da pista B

dB^- = borda de descida da pista B

Com a utilização de circuitos combinacionais e sem a utilização de um sinal periódico de amostragem do estado dos sinais de entrada do sistema, pode-se apenas realizar uma duplicação da frequência dos sinais de entrada através da função OU EXCLUSIVO entre os dois sinais de entrada.

Para a implementação dos sinais I^+ e I^-, é necessário o projeto de um circuito síncrono, onde é feita uma amostragem do estado das pistas do encoder, com a consequente geração de sinais intermediários que, combinados adequadamente, gerarão a função desejada. A figura B.3 ilustra a geração destes sinais.

Os sinais mostrados na figura B.3 se referem apenas à geração dos sinais a partir da pista A. Como se pode verificar, o que se utiliza são atrasos sucessivos de um período do relógio de amostragem e posteriormente algumas combinações destes sinais.

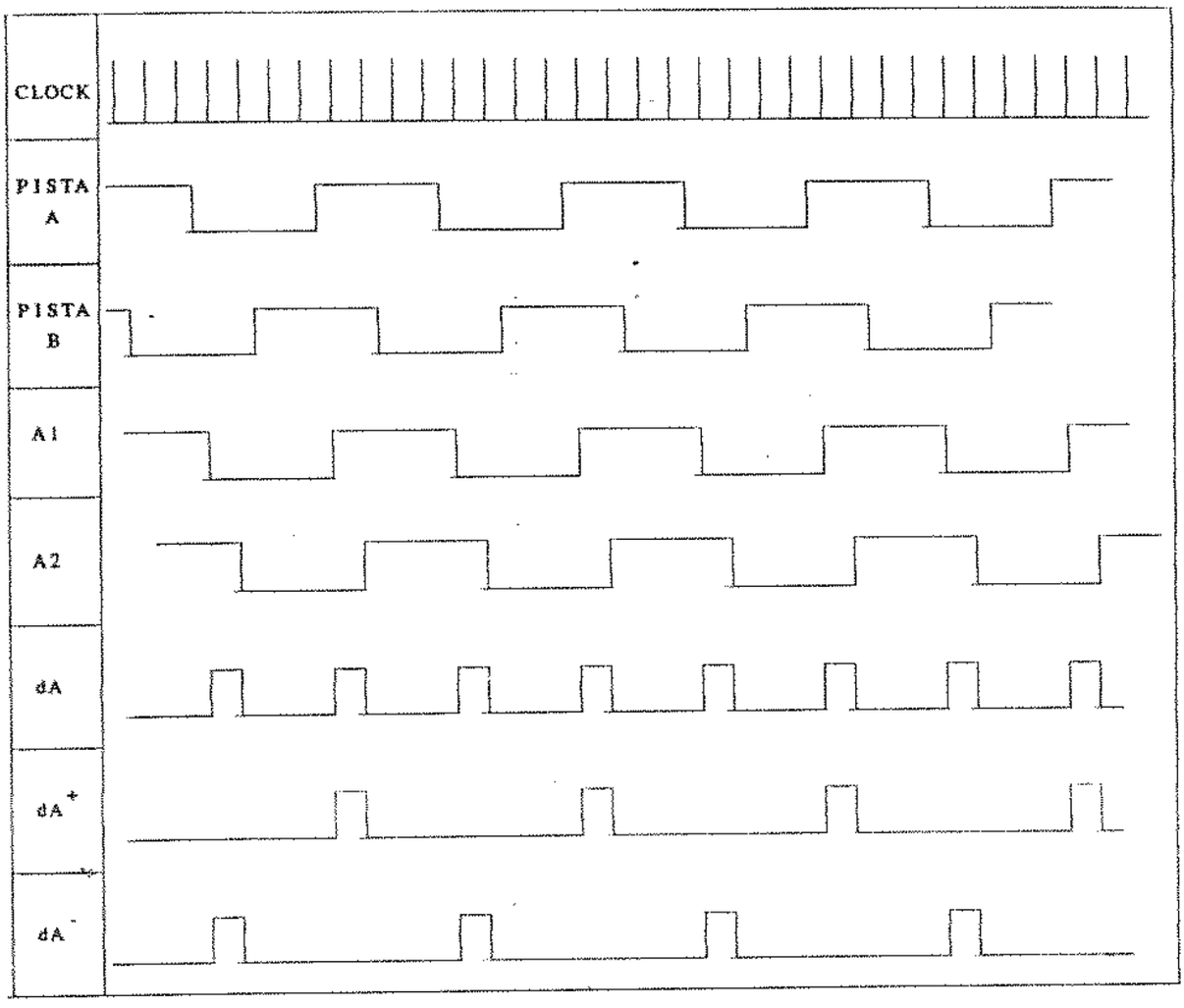


Figura B.3 : Sinais intermediários

Segue-se o mesmo princípio para a obtenção dos sinais relativos à pista B. Consequentemente podemos escrever as seguintes equações lógicas dos sinais intermediarios:

$dA = A1 \text{ xor } A2$

$dB = B1 \text{ xor } B2$

$dA^{+} = A1 \text{ and } dA$ (B.3)

$dA^{-} = \bar{A}1 \text{ and } dA$

$dB^{+} = B1 \text{ and } dB$

$dB^{-} = \bar{B}1 \text{ and } dB$

substituindo as equações acima na equação (B.1) temos:

$$I^{+} = (\bar{B} \cdot dA^{+}) + (A \cdot dB^{+}) + (B \cdot dA^{-}) + (\bar{A} \cdot dB^{-})$$
$$= \bar{B}1 \cdot A1 \cdot (A1 \times A2) + \bar{A}1 \cdot \bar{B}1 \cdot (B1 \times B2)$$
$$+ B1 \cdot \bar{A}1 \cdot (A1 \times A2) + \bar{A}1 \cdot \bar{B}1 \cdot (B1 \times B2)$$
$$= (A1 \times A2)[\bar{B}1 \cdot A1 + B1 \cdot \bar{A}1] + (B1 \times B2)\,[A1 \cdot B1 + \bar{A}1 \cdot \bar{B}1] \quad (B.4)$$

onde x indica a operação lógica de OU EXCLUSIVO.

Portanto, a equação do sinal para impulsões positivas é :

$I^{+} = (A1 \times A2).(B1 \times A1) + (B1 \times B2).\overline{(B1 \times A1)}$ caso S = 0 (B.5)

$I^{+} = (A1 \times A2).\overline{(A1 \times B1)} + (B1 \times B2).(A1 \times B1)$ caso S = 1 (B.6)

O sinal S é obtido a partir da análise da fase dos sinais de entrada e por convenção define-se o valor "0" indicando sentido positivo e o valor "1" sentido negativo de rotação da junta.

Da mesma forma temos a partir da equação B.2 as seguintes equações:

$I^{-} = (A1 \times A2).(A1 \times B1) + (B1 \times B2).\overline{(A1 \times B1)}$ caso S = 1 (B.7)

Chamando A1 x B1 = X, temos :

$$I^{+} = dA.\overline{X} + dB.X$$
$$I^{-} = dA.X + dB.\overline{X} \qquad (B.8)$$

As formas de ondas associadas a estas equações são mostradas na figura B.4, onde neste exemplo, a pista B1 está adiantada em relação à pista A1, consequentemante somente o sinal I^{+} é ativado.

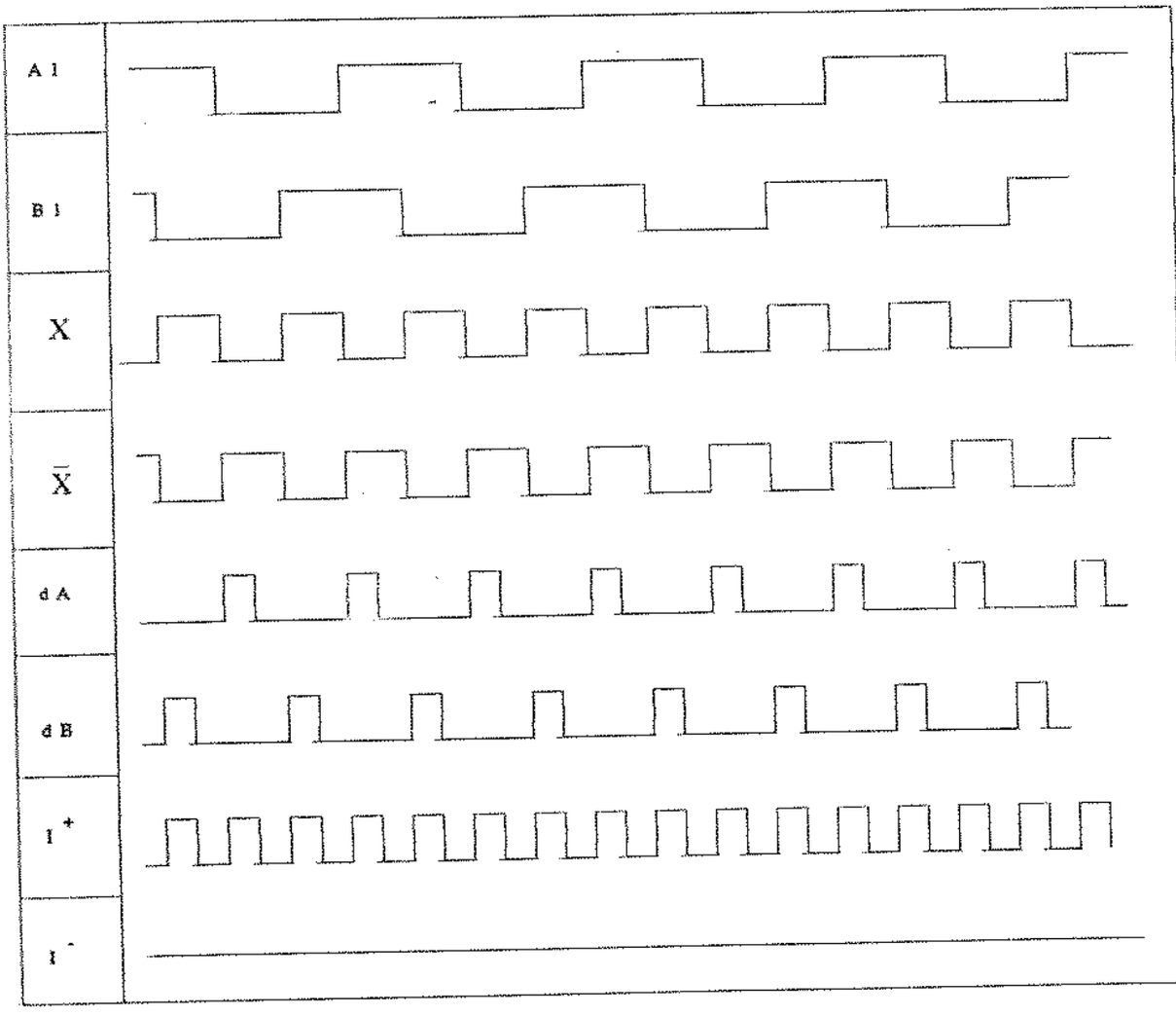


Figura B.4 : Geração dos sinais I^{+} e I^{-}

Com estas equações implementa-se o circuito lógico mostrado na figura B.5.

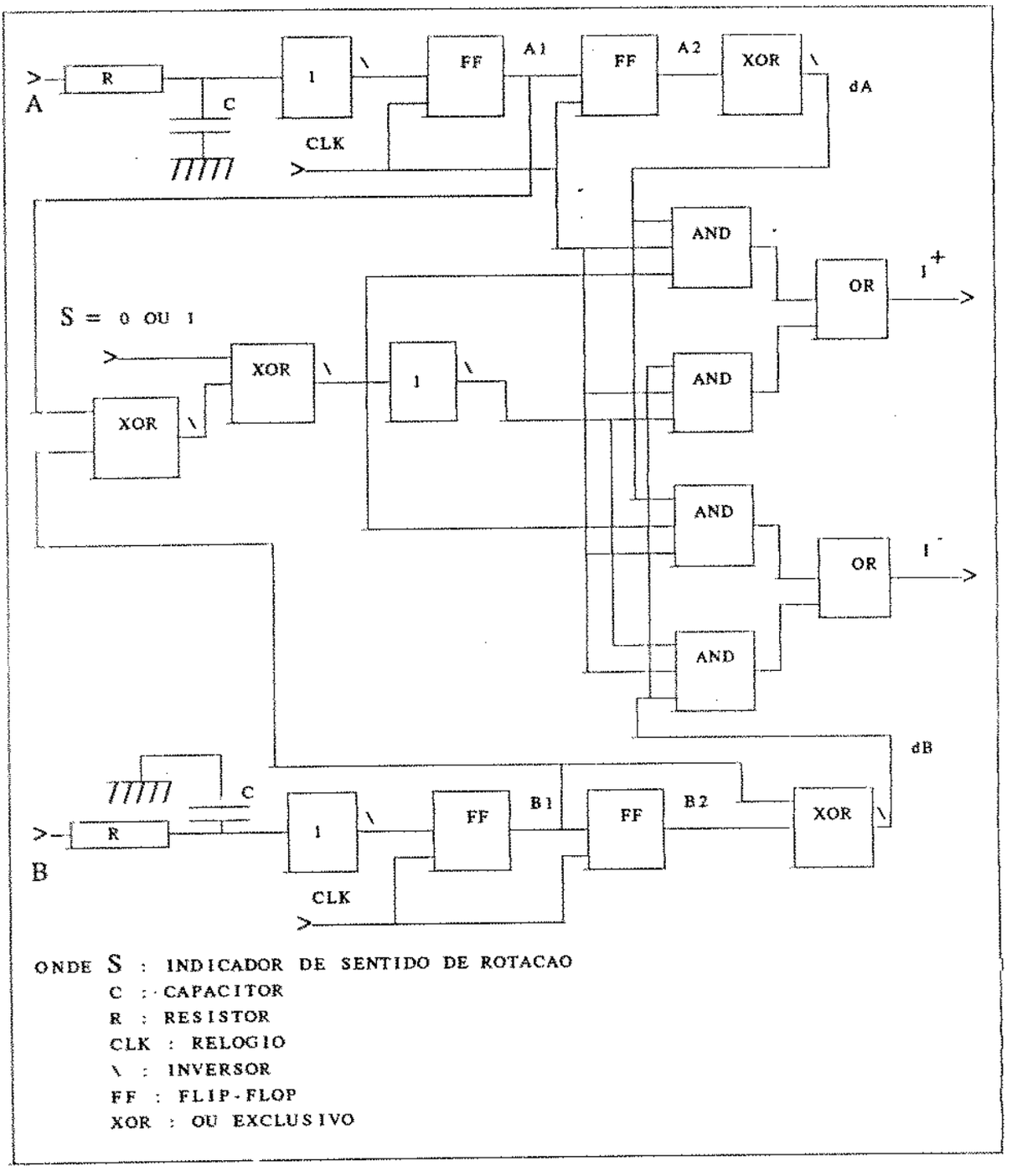


Figura B.5 : Circuito multiplicador fx4

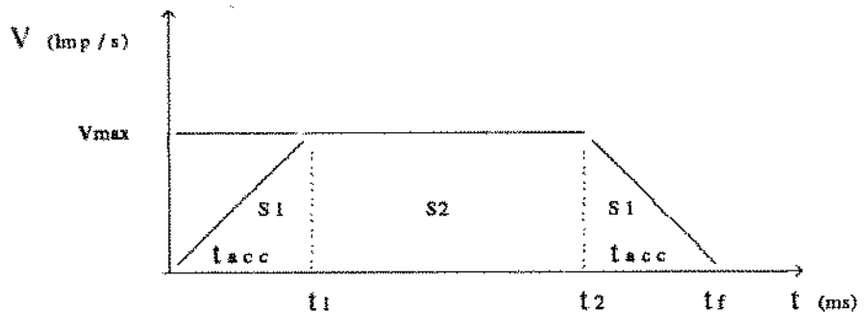
ANEXO C :

LEI GERAL DE POSICIONAMENTO DAS JUNTAS DE UM MANIPULADOR

Este anexo descreve as equações relacionadas ao movimento das juntas de um manipulador. Estas equações são utilizadas na Unidade Central de Supervisão (UCS), que implementa os algoritmos responsáveis para a geração de trajetórias, quando da utilização do modelo geométrico no modo de controle junta a junta.

A lei de movimento das juntas de um manipulador é baseada no princípio que esta terá uma aceleração inicial, velocidade constante e uma desaceleração.

Com base neste fato, o gerador de trajetória enviará à Unidade de Controle de Junta um sinal de referência que seguirá uma lei de movimento e cujo perfil de velocidade imposta à junta está representada na figura C.1.

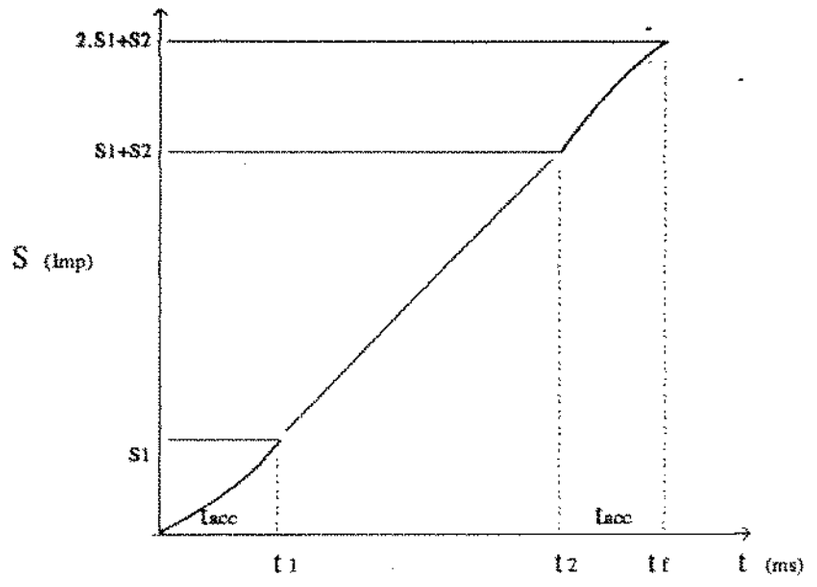


onde:

tacc: tempo de aceleração e de frenagem em ms
S_1: deslocamento angular na fase de aceleração/desaceleração
S_2: deslocamento angular na fase de velocidade constante

Figura C.1 : Velocidade da junta em função do tempo

Dado o perfil de velocidade das juntas mostrado na figura C.1, temos representada na figura C.2 o deslocamento angular de uma junta em função do tempo.



onde:

tacc: tempo de aceleração e de frenagem em ms
vmax: velocidade máxima em impulsões por segundo
S: deslocamento angular em impulsões
S_1: deslocamento angular na fase de aceleração
S_2: deslocamento angular na fase de velocidade constante
S_1: deslocamento angular na fase de desaceleração

Figura C.2 : Deslocamento angular em função do tempo

O trecho de velocidade constante será dada pela velocidade máxima da junta. Na fase de aceleração a posição angular de uma junta do manipulador será dada por:

$$x(t) = \frac{1}{2} \cdot a \cdot t^2 \qquad (C.1)$$

Onde a aceleração é dada por:

$$a = \frac{Vmax}{t\ acc} \qquad (C.2)$$

No caso geral onde várias juntas se deslocam ao mesmo tempo, a velocidade máxima será calculada proporcionalmente ao deslocamento respectivo das juntas de modo que elas partam ao mesmo tempo e cheguem no mesmo instante à posição desejada. A velocidade de deslocamento é ainda ponderada pelo fator de velocidade (% V) que pode valer de 1 a 100% da velocidade máxima das juntas.

C.1 RELAÇÕES BÁSICAS

Das figuras anteriormente apresentadas, pode-se obter as seguintes relações básicas:

1) Cálculo do deslocamento máximo

$$\Delta X_{max} = \max\ (\Delta x_1, \Delta x_2, \ldots, \Delta x_6)$$

onde Δx_i = deslocamento da junta i

2) Cálculo do tempo total

$$t_{ac} = t_{fr} \qquad t_{total} = 2.t_{ac} + t_{vel} \qquad (C.3)$$

onde:

t_{ac}: tempo de aceleração
t_{fr}: tempo de desaceleração
t_{vel}: tempo de velocidade constante

3) Cálculo do espaço total percorrido

$$S_{ac} = S_{fr} \qquad S_{total} = 2.S_{ac} + S_{vel} \qquad (C.4)$$

onde:

S_{ac}: deslocamento angular na fase de aceleração
S_{fr}: deslocamento angular na fase de desaceleração
S_{vel}: deslocamento angular na fase de velocidade constante

C.2 EQUAÇÕES BÁSICAS

Pode-se então obter as seguintes relações básicas para o deslocamento, aceleração e velocidades das juntas:

1) $$2.S_{aci} = t_{aci} \cdot \%V.\frac{\Delta X_i}{\Delta X_{max}} \qquad (C.5)$$

2) $$a_i = \frac{\%V.\frac{\Delta X_i}{\Delta X_{max}}}{t_1} \qquad (C.6)$$

3) $$v_i = \frac{\Delta X_i}{\Delta X_{max}} \qquad (C.7)$$

ANEXO D :

EXEMPLO DA OBTENÇÃO DE UMA MATRIZ INVERSA GENERALIZADA PELO MÉTODO DE GREVILLE

Este anexo, exemplifica o método de Greville, apresentado no capítulo 3 deste trabalho. Para maior clareza, apresentamos um exemplo numérico de obtenção de uma matriz inversa generalizada a partir de uma matriz 3 x 4, que não possui a sua inversa convencional.

Seja uma matriz Jacobiana que representa o modelo de um robô com quatro graus de liberdade que está sendo controlado pelo Supervisor de Controle. Dentro do processo iterativo dos algoritmos de geração de trajetória, tem-se a necessidade de inversão da matriz a cada novo ponto a ser calculado.

Suponha então a seguinte matriz A, que deverá ser invertida dentro de uma determinada iteração do algoritmo de modelagem cinemática inversa [7].

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

A partir desta matriz, pode-se obter os vetores colunas, descritos no ítem 3.3.1.2, que neste caso são:

$$A_1 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \quad a_1 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \quad a_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad a_3 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \quad a_4 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

Com estes vetores, utilizando o fluxograma apresentado na figura 3.2 do capítulo 3, e aplicando as fórmulas ali descritas, as seguintes operações são realizadas para a inversão da matriz A.

Na primeira iteração, calcula-se a matriz inversa generalizada parcial de A e alguns elementos necessários para a próxima iteração do algoritmo.

$$A_1^+ = (a_1^T \cdot a_1)^{-1} \cdot a_1^T = \begin{bmatrix} 1/5 & 0 & 2/5 \end{bmatrix};$$

$$d_2 = A_1^+ \cdot a_2 = \begin{bmatrix} 1/5 & 0 & 2/5 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2/5 \end{bmatrix},$$

$$c_2 = a_2 - A_1 \cdot d_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \begin{bmatrix} 2/5 \end{bmatrix} = \begin{bmatrix} -2/5 \\ 1 \\ 1/5 \end{bmatrix} \neq 0$$

Como o vetor c_2 é diferente de zero, utiliza-se a fórmula para b_i mostrada abaixo, e com a obtenção deste parâmetro, pode-se calcular a matriz inversa generalizada da segunda iteração do algoritmo.

$$b_2 = c_2^+ = (c_2^T c_2)^{-1} c_2^T = \left[\begin{bmatrix} -2/5 & 1 & 1/5 \end{bmatrix} \begin{bmatrix} -2/5 \\ 1 \\ 1/5 \end{bmatrix} \right]^{-1} \begin{bmatrix} -2/5 & 1 & 1/5 \end{bmatrix}$$

$$= 5/6 \begin{bmatrix} -2/5 & 1 & 1/5 \end{bmatrix} = \begin{bmatrix} -1/3 & 5/6 & 1/6 \end{bmatrix},$$

A matriz inversa generalizada na segunda iteração é portanto dada por:

$$A_2^+ = \begin{bmatrix} A_1^+ - d_2 b_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1/3 & -1/3 & 1/3 \\ -1/3 & 5/6 & 1/6 \end{bmatrix},$$

Novamente repete-se então o procedimento anterior de cálculo de alguns fatores para o início de nova iteração.

$$d_3 = A_2^+ . a_3 = \begin{bmatrix} 2/3 \\ -7/6 \end{bmatrix},$$

$$c_3 = a_3 - A_2 . d_3 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2/3 \\ -7/6 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/6 \\ -1/6 \end{bmatrix} \neq 0$$

Novamente o fator c_i é diferente de zero, sendo portanto b_i obtido pela seguinte relação:

$$b_3 = c_3^+ = (c_3^T c_3)^{-1} c_3^T = 6 \begin{bmatrix} 1/3 & 1/6 & -1/6 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \end{bmatrix}.$$

A matriz inversa generalizada da matriz A na terceira iteração do algoritmo é dada por:

$$A_3^+ = \begin{bmatrix} A_2^+ - d_3 b_3 \\ b_3 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 2 & 2 & -1 \\ 2 & 1 & -1 \end{bmatrix}$$

Para a quarta iteração temos os seguintes fatores:

$$d_4 = A_3^+ . a_4 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},$$

$$c_4 = a_4 - A_3 . d_4 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \mathbf{0}$$

(onde $\mathbf{0}$ é o vetor coluna nulo),

Nesta iteração o fator c_i resultou num vetor nulo, sendo então necessário o uso do outro ramo do fluxograma apresentado na figura 3.2 do capítulo 3. Os cálculos são mostrados a seguir.

$$b_4 = (1 + d_4^T d_4)^{-1} d_4^T A_3^+ = \begin{bmatrix} 0 & 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \\ 2 & 2 & -1 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 4/3 & 1 & -2/3 \end{bmatrix},$$

Finalmente então, obtemos a inversa generalizada final da matriz A, que é dada por:

$$A_4^+ = \begin{bmatrix} A_3^+ - d_4 b_4 \\ b_4 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 2/3 & 1 & -1/3 \\ 2/3 & 0 & -1/3 \\ 4/3 & 1 & -2/3 \end{bmatrix} = A^+.$$

Note-se que no exemplo exposto, a matriz inversa convencional $\tilde{A}^{1}$ não existe.

É preciso ficar claro que essas iterações do método de Moore-Penrose não são as mesmas iterações do algoritmo de modelagem cinemática inversa, isto é, as iterações aqui apresentadas situam-se dentro de iterações maiores daquele algoritmo, configurando assim o que pode-se chamar de um *loop* do algoritmo de Moore-Penros dentro de outro *loop* do algoritmo de modelagem cinemática inversa.