UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE SISTEMAS E CONTROLE DE ENERGIA

TESE DE DOUTORADO

CONTROLE DE TRAJETÓRIAS CONTÍNUAS POR SECCIONAMENTO EM SUB-TRAJETÓRIAS USANDO INTELIGÊNCIA ARTIFICIAL NUM ROBÔ MULTI-TAREFAS

Autor: *Marconi Kolm Madrid*
Orientador: *Prof. Dr. Alvaro Geraldo Badan Palhares*
Agradecimento Especial: *Prof. Dr. Félix Monastério Huelín y Maciá*

O Prof. Félix Monastério Huelín y Maciá colaborou nas discussões para a elaboração do Capítulo III durante o estágio de Doutorado *Sandwich* realizado na Escuela Técnica Superior de Ingenieros de Telecomunicación da Universidad Politécnica de Madrid, com bolsa de estudos concedida pela Secretaria de Ciência e Tecnologia - Programa de Recursos Humanos em Áreas Estratégicas - RHAE/CNPq.

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas-FEE/UNICAMP, como parte dos requisitos exigidos para a obtenção do título de DOUTOR EM ENGENHARIA ELÉTRICA.

MARÇO - 1994

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE SISTEMAS E CONTROLE DE ENERGIA

CONTROLE DE TRAJETÓRIAS CONTÍNUAS POR SECCIONAMENTO EM SUB-TRAJETÓRIAS USANDO INTELIGÊNCIA ARTIFICIAL NUM ROBÔ MULTI-TAREFAS

Aprovada em 28 de Abril de 1994 pela seguinte comissão julgadora:

Prof.Dr. Alvaro Geraldo Badan Palhares
Departamento de Sistemas e Controle de Energia
Faculdade de Engenharia Elétrica
Universidade Estadual de Campinas

Prof.Dr. Félix Monastério Huelin y Maciá
Departamento de Ingenieria de Control
Escuela Técnica Superior de Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

Prof.Dr. João Maurício Rosário
Departamento de Projeto Mecânico
Faculdade de Engenharia Mecânica
Universidade Estadual de Campinas

Prof.Dr. Celso Pascoli Bottura
Departamento de Máquinas, Controle e Sistemas Inteligentes
Faculdade de Engenharia Elétrica
Universidade Estadual de Campinas

Prof.Dr. Sigmar Maurer Deckmann
Departamento de Sistemas e Controle de Energia
Faculdade de Engenharia Elétrica
Universidade Estadual de Campinas

RESUMO

Este trabalho de tese propõe uma técnica para a realização do planejamento e o controle de trajetórias espaciais de robôs através de uma abordagem que utiliza conhecimentos da inteligência artificial, como alternativa para realizar este tipo de tarefa, em tempo-real, mesmo que se utilizem sistemas eletrônicos de controle que não tenham grande capacidade computacional. Esta é uma técnica numérica que não exije o conhecimento da modelagem cinemática inversa do robô, daí a sua enorme vantagem computacional perante as técnicas clássicas de abordagem do problema, e até onde nossas pesquisas chegaram, ela demostrou-se muito eficiente para o rastreamento veloz e preciso, tanto de trajetórias simples como de trajetórias definidas por equações de grande complexidade matemática. Além de intrinsicamente dar grande capacidade de recuperação do rastreamento frente à eventuais perturbações que o robô possa sofrer em sua estrutura física durante os movimentos. O estudo de tal assunto situa-se na área da Programação Heurística da Inteligência Artificial, e mais especificamente, ela utiliza algoritmos de busca que visam encontrar pelo menos uma solução de configuração estrutural do robô para rastrear trajetórias completas definidas dentro do seu volume de trabalho.

Criamos uma filosofia para o gerenciamento da operação de robôs através de um modelo de interface homem-máquina traduzida para um pacote de *software* apto para ser processado por um *hardware* eletrônico de dois níveis hierárquicos, que permite a realização do planejamento, da simulação, da execução, e da análise do desempenho dos robôs na execução de tarefas.

Situamos o problema do planejamento e controle de trajetórias através de comentários sobre os assuntos envolvidos por esta área da robótica, baseados em pesquisas bibliográficas atualizadas, e através de nossa perspectiva para propor soluções. Apresentamos resultados de simulações que demonstram as potencialidades da técnica proposta, através da utilização do pacote de *software* criado, com comentários conclusivos baseados na utilização das características de um robô cuja construção evoluiu juntamente com este trabalho. E apresentamos nossas perspectivas futuras de trabalho nesta linha de pesquisa.

'A Juvenil e Zelma meus queridos pais,
e ao meu pé direito...'

« O Valor da Pessoa é o Valor dos Valores...»

AGRADECIMENTOS

Foram muitos os amigos que contribuiram comigo para que eu possa agora estar editando esta finalização da redação do texto, preparado para apresentar meu trabalho de tese de doutorado. E com cada uma dessas amáveis pessoas quero compartilhar minha alegria com um abraço forte e sincero de agradecimento.

Desde que saí de minha terra natal Erechim, no Rio Grande do Sul, em busca da realização do meu primeiro grande sonho: Ser Técnico de Eletrônica, depois realizando o curso de Graduação em Engenharia Elétrica e parte significativa do curso de Bacharelado em Física, até a realização do curso de Mestrado em Engenharia Elétrica, e agora o curso de Doutorado em Engenharia Elétrica, realizei uma trajetória que hoje me dá grande satisfação de relembrar, porisso hoje estou muito grato à todos os professores que colaboraram na minha educação, e à minha família que me deu apoio em todos estes momentos proporcionando as condições fundamentais para isso.

Durante este meu trabalho de tese de doutorado houveram algumas pessoas que contribuíram comigo de forma muito especial, e a todos estes amigos também quero agradecer de modo especial. São eles:

♥ Meu orientador de tese Prof. Dr. *Alvaro Geraldo Badan Palhares*, porque foi mais do que um professor atencioso, competente e amigo. Foi um colega que me ensinou coisas muito complexas de uma forma muito agradável, teve paciência para conduzir-me de forma brilhante até a conclusão deste trabalho, e tomou muitas decisões importantíssimas à meu favor, com enorme confiança pessoal, que me foram de grande valia, abrindo-me caminhos e perspectivas de muita importância para a realização desta tese, e para futuros trabalhos de pesquisa em áreas de alta tecnologia.

♥ Prof. Dr *Félix Monastério Huelín y Maciá*, outro professor que dedicou grande atenção, interesse e participação para com este trabalho. Ajudou-me durante o tempo dos estudos de doutorado *sandwich*, colaborando nas discussões, dando sugestões técnicas, e fazendo análises para que esta parte da pesquisa fosse finalizada com exito. Agradeço sua cortesia e hospitalidade por ter-me alojado em sua casa nos primeiros dias de adaptação em Madrid, sua gentileza de emprestar-me um bom microcomputador para meu uso particular numa época que foi de enorme valia no contexto do trabalho. E por ter-me aceito como pesquisador no projeto *Robot-Saltador Acionado Por Motores de Acoplo-Directo*, sob sua coordenação.

♥ Os ex-colegas de pós-graduação, *Marcus de Aguiar Dias* e *Adilson Sakahi Ohfugi*, hoje Mestres em Engenharia Elétrica, que colaboraram muito na fase dos projetos do *hardware* eletrônico que hoje forma o sistema de controle do robô que construímos.

♥ O Técnico de Mecânica *Eduardo Gavira Bonani*, pelos seus trabalhos habilidosos e criativos nos projetos, usinagens e montagens mecânicas das peças de precisão que compõem a estrutura do robô construído, pelas suas sugestões eficazes dadas para a solução de problemas de construção mecânica, e por sua colaboração constante no auxílio às execuções de trabalhos variados de montagens durante o desenvolvimento da tese.

♥ Os alunos de Graduação em Engenharia Elétrica da Unicamp *Antônio E. R. Neger* e *Alexandre F. S. Osório*, e os Técnicos de Eletrônica *Paulo César Menegon* e *Renato Pinto Nazario* pelos seus trabalhos habilidosos no desenvolvimento e nos testes do *hardware* eletrônico do robô. E o aluno de Graduação em Engenharia Elétrica da Unicamp *Alexandre C. R. Caminha* pela sua colaboração no desenvolvimento do *software* para o robô.

♥ O Engº *Valmir Tadeu Fernandes* pela sua grande colaboração nos trabalhos de pesquisa em laboratório, por suas idéias e sugestões para o desenvolvimento da estrutura mecânica do robô, e juntamente com sua esposa *Silvia*, por terem cuidado do meu apartamento durante o tempo que realizei o doutorado *sandwich*.

♥ O aluno de Mestrado em Engenharia Elétrica da Unicamp *Paulo Comim* pelo bom trabalho fotográfico que realizou para auxiliar nas ilustrações da tese.

♥ Os funcionários da Faculdade de Engenharia Elétrica da Unicamp: *Masamiti Motoyama* pelo seu excelente trabalho de apoio nos serviços de almoxarifado; *Regina M. A. Gazola Floriano*, pelo apoio na confecção das placas de circuito impresso para as montagens do *hardware* eletrônico do robô; E *Marisa Geraldino Pereira* pela constante e eficiente colaboração nos serviços de secretaria que foram necessários para o bom andamento do trabalho.

♥ Os colegas de laboratório da TELECO na Universidad Politécnica de Madrid: *Benito*, *Javi*, *Cristina*, *Luis Magdalena*, *Rafa*, e *Paloma*, que em muitos momentos me apoiaram com favores, informações e idéias úteis, e também me deram muitas alegrias durante o doutorado *sandwich*.

♥ Os colegas de Apartamento de Madrid: *Everson*, *Paulo* e *Clodeinir*, pelo coleguismo, pelos inúmeros favores que me prestaram quando necessitei de suas ajudas, e pelas alegrias que me proporcionaram.

♥ A *Monika* e o *Ulrich* pelas suas amáveis companhias que me permitiram conhecer a Alemanha num tempo muito importante para minha formação.

♥ A *Lissandra*, *Zeca*, *Cássia*, *Maria Eugênia*, *Silvia*, *Liléa*, *Célia*, e *Gilson*, que foram companheiros muito legais que tive durante esta época de estudos. E em especial para a *Sandra* que além da sua amável companhia, me ajudou enormemente em todos os momentos, proporcionando muitas alegrias e incentivos importantes para que eu tivesse exito nesta jornada.

♥ E ainda quero agradecer a população brasileira que através do *CNPq* (Conselho Nacional de Pesquisas), e do *Programa RHAE* (Programa de Recursos Humanos em Áreas Estratégicas) da *Secretaria de Ciência e Tecnologia*, possibilitaram a realização deste trabalho através da concessão de bolsas de estudos.

Marconi Kolm Madrid

PREFÁCIO

Em 1994 ainda há muita polêmica sobre a utilização de robôs para realizar trabalhos que os seres humanos podem e não podem realizar. Continuam havendo controvérsias no campo social, ainda existem muitas dúvidas entre a população menos esclarecida sobre os avanços tecnológicos, quanto a questão dos robôs serem máquinas rivais de competição, com capacidade de substituir completamente os homens. Acreditamos que a formação dessas opiniões que persistem, deva-se ao fato histórico criado principalmente por novelas e contos de ficção científica, desde que o homem começou a sonhar com a criação de máquinas que pudessem servir de ferramentas para facilitar seu trabalho, aumentando sua capacidade e rapidez para executá-lo.

O problema é, que de acordo com a organização das sociedades existe uma exigência natural para que o homem trabalhe de alguma forma e com isso consiga sobreviver. E também por questões de organização social, surgem grandes diferenças entre as formações pessoais dos indivíduos, e grandes quantidades desses indivíduos acabam por ter formações muito pobres, que não lhes permitem ter facilidade de adaptaçao à mudanças, mesmo que elas sejam pequenas.

É inegável o benefício que as máquinas e ferramentas trazem para qualquer campo de aplicação. A automação, principalmente com o uso de robôs, permite grandes incrementos na produtividade do trabalho. A qualidade dos produtos gerados fica melhorada porque os robôs uniformizam a produção, diminuindo as perdas e a geração de refugos. Com a automação pode-se eliminar os tempos mortos dos processos permitindo o melhor aproveitamento dos recursos humanos, físicos e financeiros. Com ela também pode-se dar maior flexibilidade aos processos de fabricação, fazendo-se as devidas adaptações às necessidades do mercado, e evitando a formação de estoques desnecessários devido a ter-se a possibilidade de aumentar a velocidade da produção, e da rápida

captação das necessidades atualizadas desse mercado.

Também é inegável que a automação provoca o remanejamento dos setores de trabalho onde é necessário o trabalho humano. Com isso vem a necessidade de deslocar-se adequadamente estes trabalhadores, e sempre que existem mudanças dessa natureza, podem ocorrer problemas sociais associados a elas, pelo menos é o que nos mostra a história. Mas nas sociedades com maiores índices de desenvolvimento, já há formação cultural suficiente que frena, ou impede, as ocorrências de impactos negativos sobre os trabalhadores, indicando que estes possíveis problemas têm solução. Por exemplo, a ação sindical pode possibilitar que os contratos coletivos regulem todos os assuntos de interesse dos trabalhadores na adoção da automação, como: Intensidade, ritmo, e jornada de trabalho, salários, investimentos em recursos humanos para atualização técnica, supervisão, acesso à informação, higiene, segurança do trabalho, etc...

Resultados de pesquisas internacionais feitas com fabricantes de equipamentos automatizados e respectivos usuários em países industrializados, indicam que um equipamento de comando numérico pode afetar/deslocar de 4 a 8 trabalhadores; Um robô pode afetar/deslocar de 5 a 7 trabalhadores; Uma estação *CAD* pode afetar/ deslocar de 2 a 20 engenheiros; E um SFM (Sistema Flexível de Manufatura) pode afetar/ deslocar até 10 trabalhadores. Estimava-se em meados dos anos 80, que a manufatura brasileira poderia deixar de absorver algo entre 800.000 e 2.400.000 trabalhadores até 1990, caso o processo de automação industrial brasileiro seguisse o mesmo padrão que tem prevalecido nos Estados Unidos da América **[63]**. Estes são dados preocupantes, que devem ser interpretados com muito cuidado no contexto brasileiro, porque existem várias diferenças sócio-econômicas entre estes e o Brasil. A mais importante delas é o custo da mão-de-obra industrial, que certamente afeta muito o assunto pesquisado. Acreditamos que a postura mais adequada é permitir que a sociedade e ao governo avaliem conjuntamente em quais situações é melhor permitir aumentos nos níveis de desemprego e ganhar em produtividade para garantir competitividade internacional, e em quais situações é melhor impedir ou restringir a modernização no sentido de preservar os empregos naqueles casos onde ele é inevitável, e atuem de modo a evitar ao máximo a aparição de malefícios sociais.

Há quem defenda a tese de que quanto mais desenvolvido for o país, menor será o impacto da automação nos níveis de emprego. A explicação para isso está no nível de conscientização e organização dos trabalhadores, o que lhes dá maior poder de pressão e barganha **[66]**.

Outros especialistas dizem que os artigos de necessidade como: Roupas, alimentos, carros, móveis, etc..., tendem a ser produzidos por apenas 10% da força de trabalho braçal humana, e que a saída para resolver o problema dos 90% restantes é que os governos adotem políticas maciças de educação gratuita em todos os níveis. Estes programas teriam não somente o mérito de absorver o tempo das pessoas, mas também aumentariam a capacitação dos indivíduos, criando maiores possibilidades de descobrir-se novas formas de utilização para a automação, novos serviços e novos produtos **[74]**.

Segundo um estudo, feito pela entidade sindical inglesa *APEX* **[3]**, só existe uma forma de atenuar os males do desemprego: Uma atuação unida entre governo, sindicatos e empresários, no sentido de que os três estudem conjuntamente cada setor da economia, estabelecendo impostos para as empresas que obtivessem lucros às custas da automação de processos outrora manuais.

Talvez a medida mais adequada para estas circunstâncias é a utilização da reciclagem ou do aprimoramento do treinamento pessoal, a exemplo do que acontece no Japão, na Alemanha, na Itália e na Suécia. O ponto principal, neste caso, é a aceitação por parte dos empresários, de não dispensar nenhum empregado como decorrência do emprego da automação, mas que, ao contrário, devam procurar adaptá-lo em outras atividades em suas próprias empresas. Essa medida possui vantagens sobre a redução da jornada de trabalho e o seguro-desemprego **[57]**.

Quanto a utilidade da robótica, existem seis razões básicas para estimular o uso dos robôs no setor industrial. A principal é a otimização do trabalho,, depois vem a substituição do trabalho humano em ambientes hostis ou perigosos **[64]**, a flexibilização do sistema de produção, a obtenção de um controle de qualidade mais acurado, o aumento da produtividade, e a contribuição para atenuar a escassez do trabalho especializado **[76]**.

Hoje existem poucas indústrias de sucesso em suas atividades, excluíndo algumas que exploram o artesanato artístico, principalmente perante o aspecto de competição tecnológica para impor-se no mercado, que não estejam adotando algum tipo de automação nos seus processos de fabricação e gerenciamento, e muitas das que estão se impondo fortemente, usam robôs e inclusive desenvolvem robôs. Porém, como ainda o domínio sobre os projetos, a construção e o controle de máquinas com estas tecnologias, está muito restrito a alguns setores que se adiantaram nesta corrida tecnológica, promovendo seu desenvolvimento com altos investimentos financeiros, resulta que eles só podem ser adquiridas por preços muito elevados, variando desde U$ 6.000 até U$ 200.000 ou mais, dependendo de aspectos como capacidade de carga, número de graus de liberdade, velocidade máxima de trabalho, precisão, suavidade e repetibilidade dos movimentos, acessórios para a flexibilização da utilização da máquina, volume de trabalho, facilidades oferecidas para a sua programação e manutenção, tipos possíveis de controle de trajetórias, e outros...

Motivados por estudar com profundidade os aspectos tecnológicos que envolvem o projeto, a construção, e o controle dos robôs, propusemos o desenvolvimento desta tese de doutorado na certeza de poder dar grande contribuição para o ensino da robótica e ampliar sua difusão em nossos meios científicos e industriais. Com este estudo aplicado, obtivemos resultados teórico/práticos importantes sobre a construção de robôs, que podem oferecer alternativas muito atraentes para aplicações nos ambientes industriais, com propostas inovadoras para a realização do planejamento e do controle de suas trajetórias espaciais necessários a execução de tarefas úteis. E para apresentarmos nossas idéias, nossos procedimentos teóricos e experimentais, os resultados e nossas conclusões, produzimos este texto composto por cinco capítulos e três anexos, cuja distribuição global de conteúdo é a seguinte:

Reservamos o **Capítulo I** para fazer uma descrição do robô **JECAII** no qual podem ser aplicadas as técnicas que desenvolvemos para o planejamento e gerenciamento do controle de trajetórias de robôs. Subdividimos este capítulo em duas partes, uma de *hardware*, onde apresentamos as principais características de sua construção mecânica e eletrônica, e uma de *software* onde fazemos a apresentação dos detalhes da programação que concebemos para o sistema de controle eletrônico, que serve para operacionalizar o robô e controlar o processo de Interface Homem-Máquina, para permitir a interação com usuários que irão operá-lo. Apresentamos toda sua engenharia de programação dentro de uma estrutura hierárquica, que é formada por um pacote de programas que facilitam as tarefas de planejamento, simulação, execução, e análise de desempenho dos movimentos

do robô. Baseamos todo o seu desenvolvimento visando a modularidade da programação, e a fácil manutenção para aplicá-lo ao robô que construímos, e para permitir que com pequenas modificações ele possa ser aplicado ao controle de outras máquinas com características semelhantes.

No **Capítulo II** situamos o problema do planejamento e controle de trajetórias através de comentários sobre os assuntos envolvidos por esta área da robótica baseados em pesquisas bibliográficas atualizadas, e sobre nossa perspectiva para propor soluções.

Aprofundando-nos mais na questão do planejamento e gerenciamento do controle das trajetórias do robô, apresentamos no **Capítulo III**, as abordagens que nos conduziram à criação de uma técnica original, que utiliza conhecimentos da inteligência artificial, como alternativa para realizar este tipo de controle sobre os robôs, em tempo-real, mesmo que se utilizem sistemas de controle eletrônico que não tenham grande capacidade computacional. Esta é uma técnica numérica que não exije o conhecimento da modelagem cinemática inversa do robô, daí a sua enorme vantagem computacional perante as técnicas clássicas de abordagem do problema, e até onde nossas pesquisas chegaram, ela demostrou-se muito eficiente para o rastreamento veloz e preciso, tanto de trajetórias simples como de trajetórias definidas por equações de grande complexidade matemática. Além de intrinsicamente dar grande capacidade de recuperação do rastreamento frente à eventuais perturbações que o robô possa sofrer em sua estrutura física durante os movimentos. O estudo de tal assunto situa-se na área da Programação Heurística da Inteligência Artificial, e mais especificamente, ela utiliza algoritmos de busca que visam encontrar pelo menos uma solução de configuração estrutural do robô para rastrear trajetórias completas definidas dentro do seu volume de trabalho.

O **Capítulo IV** foi reservado para apresentarmos alguns dos resultados mais significativos que obtivemos, através da aplicação da técnica de planejamento e controle de trajetórias proposta, fazer comentários e apresentar nossas conclusões baseados nestes resultados. Por eles pode-se verificar que dependendo dos níveis de precisão requisitados para os movimentos, as soluções encontradas convergem para as soluções dadas pelas técnicas analíticas que utilizam a modelagem cinemática inversa dos robôs.

Como este trabalho terá prosseguimento, apresentamos no **Capítulo V** algumas de nossas perspectivas para a seqüência das pesquisas, com propostas intencionando que haja aumento da flexibilidade do sistema para ser aplicado a uma maior variedade de tarefas, comentando aspectos sobre sua modularização e fazendo propostas para que a técnica de planejamento e controle de trajetórias apresentada, seja incrementada com características que possam ser úteis para melhor mais seu desempenho.

Os **Anexos A, B e C** foram reservados para apresentar em detalhes o *hardware* eletrônico do **JECAII**. O **Anexo A** contém informações importantes sobre a arquitetura e a programação de baixo nível dos seus componentes. O **Anexo B** contém informações sobre o projeto do sistema de Acionamento do robô. E o **Anexo C** contém informações sobre a programação de baixo nível que operacionaliza o seu funcionamento através de dois níveis hierárquicos de controle.

CONTEÚDO

CAPÍTULO I

DESCRIÇÃO DO ROBÔ MULTI-TAREFAS JECA II

PRIMEIRA PARTE - HARDWARE

1.1 ☞ BREVE HISTÓRICO

Projeto Jeca foi o nome que escolhemos para referir-nos ao trabalho de pesquisa científica em robótica iniciado em 1986 com o desenvolvimento da tese de mestrado entitulada: Robô-Manipulador Mecânico TRRR Para Posicionamento Espacial com Controle Hierárquico a Microprocessadores, apresentada em agosto de 1988 pelo mesmo autor e mesmo orientador que apresentam esta tese de doutorado. Nesta época se integraram à equipe vários alunos de graduação da Faculdade de Engenharia Elétrica da Unicamp em trabalho de iniciação científica, e lançamos um plano para iniciar a construção de um robô que servisse para ser utilizado no ensino, e que pudesse ser aplicado para fazer trabalhos industriais, tendo como objetivo principal criar uma tecnologia brasileira para a construção de robôs, e aprofundar nossos conhecimentos sobre os assuntos referentes às áreas envolvidas em robótica.

Com utilização principalmente de materiais e componentes retirados de equipamentos em desuso, construímos um braço mecânico de quatro graus de liberdade fazendo as devidas adaptações para conseguir que a estrutura pudesse ser movida acionada por motores elétricos de corrente contínua controlados pela corrente de armadura, e através de um sistema de controle digital baseado em microprocessadores Z-80. Este robô foi chamado de **JECA I**, justamente porque sabíamos que o projeto poderia ser continuado com a construção de outros robôs que fossem incorporando os conhecimentos adquiridos através da pesquisa e experimentação em cada etapa

desenvolvida. Isso realmente aconteceu; hoje a construção do robô **JECA II** incorpora muito conhecimento adquirido com o desenvolvimento do **JECA I**.

Dotamos o **JECA II** com a mesma geometria para posicionamento espacial dos três graus de liberdade rotacionais do **JECA I**, apesar de agora termos desenvolvido um projeto mecânico bem mais elaborado e para cinco graus de liberdade mais uma garra de dois dedos, com materiais especialmente usinados para isso. Mantivemos a idéia de utilizar engrenagens para fazer a amplificação de torque dos motores para cada junta, mas ao invés de utilizarmos engrenagens com transmissão direta entre dentes, utilizamos sistemas de correias e polias sincronizadoras para reduzir ao máximo os problemas de folgas nas transmissões (*backlash*). Incorporamos também as tecnologias adquiridas sobre o sistema de nivelamento do robô no solo, e métodos para usinar peças que devem ter alinhamento de precisão. Passamos a utilizar materiais mais resistentes, como o latão e o aço ao invés do alumínio, para a construção de peças das partes da estrutura que devem suportar maiores esforços, segundo constatações feitas em experimentos com o **JECA I**.

Também mantivemos a arquitetura eletrônica de dois níveis hierárquicos do robô **JECA I** (*Master/Slaves*), apesar de agora o **JECA II** possuir um sistema de controle eletrônico mais potente e com padronização para permitir a fácil incorporação de novas tecnologias, e a fácil modularização desta arquitetura para que o robô possa ser adaptado para realizar tarefas em uma maior gama de ambientes.

Reservamos este capítulo para fazer a descrição da tecnologia empregada neste robô, o qual projetamos durante o desenvolvimento desta tese de doutorado, e cujo protótipo se encontra em fase final de construção mecatrônica.

O conhecimento dos aspectos apresentados neste capítulo é essencial porque eles apresentam o ambiente utilizado para basear o desenvolvimento deste trabalho de tese que culminou com a geração da técnica original que apresentamos para o controle de trajetórias contínuas de robôs com inteligência artificial em tempo-real.

1.2 ☞ ESTRUTURA MECÂNICA

Como o robô possui cinco graus de liberdade mais uma garra, os trabalhos de montagem mecânica foram divididos em seis etapas, sendo cada etapa referente a construção da respectiva junta da estrutura.

Como optamos por utilizar motores de corrente contínua já existentes no laboratório para mover a estrutura mecânica, baseamos a capacidade operativa do robô neste fato, projetando as demais partes para comportar tais motores.

Para aproveitarmos de forma adequada a capacidade de cada motor para mover sua respectiva junta, minimizando os problemas de folgas nas transmissões, e adequando torques e velocidades, optamos por construir relações de redução de velocidades com uso de polias, correias sincronizadoras e fusos. Sendo que cada junta é capaz de imprimir uma velocidade tangencial de 1m/s na garra do robô, quando somente ela for acionada com sua velocidade máxima. Essa velocidade de garra é suficiente para inúmeras aplicações bastando que se verifique na literatura especializada em robótica.

O desenvolvimento de cada peça da estrutura começou por seu projeto com auxílio de programas CAD (*AUTOCAD*), com produção do respectivo desenho técnico, após houve o respectivo trabalho de usinagem na oficina mecânica da Faculdade de Engenharia Elétrica da Unicamp, com uso de materiais como aço, alumínio, latão, bronze, etc.., determinados por fatores como peso, tipo de esforço que a peça sofrerá, adaptação a soldas por arco elétrico e acetileno, e preço no mercado especializado. A oficina mecânica utilizada possui máquinas como fresadora, torno, plainadora, dobradora, serra-fita, guilhotina, jateadora de areia, furadeira de coluna, prensa de 5 toneladas, equipamentos de solda a arco e acetileno, e outras ferramentas necessárias a usinagem de peças mecânicas de precisão.

1.2.1 ☞ MECANISMO DE POSICIONAMENTO

♦ *1º GRAU DE LIBERDADE*

O primeiro grau de liberdade é colocado sobre uma base composta por três pés com ângulos de 120° entre si, construídos em aço, com possibilidade de nivelamento quando adaptados a assoalhos irregulares, através de parafusos em suas extremidades.

Sua junta é tipo rotacional, com liberdade de movimentos de 280°. O motor utilizado possui velocidade máxima de 1500rpm quando submetido a uma tensão de 24 volts à vazio. Em função desta característica, e da velocidade tangencial desejada na garra, optamos por construir um sistema de redução de velocidade com fator de redução de 128:1 (**figura 1.1**).

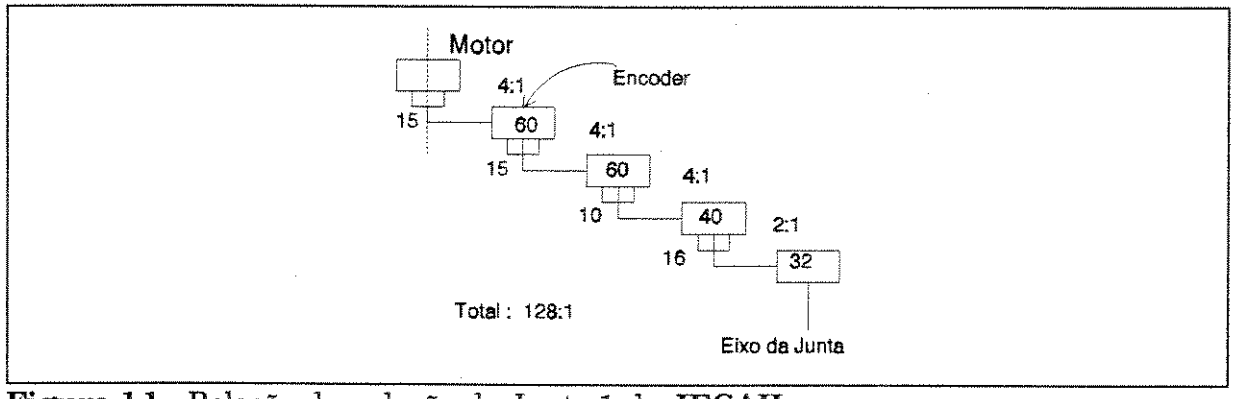


Figura 1.1 - Relação de redução da Junta 1 do **JECAII**.

Para fazermos as medidas das variáveis de posição, velocidade e aceleração, utilizamos, respectivamente, um codificador óptico incremental de 1024 pulsos/rotação que fica solidário ao 2º eixo do sistema de redução, um tacômetro analógico com relação de 5v/1000rpm solidário ao eixo do motor, e um resistor shunt de 0,01Ω em série com a alimentação do motor, como artifício para medir a corrente elétrica que, no caso dos motores que utilizamos, é proporcional às suas acelerações. Isso também é feito para cada uma das juntas do robô (**figura 1.2**).

Também construímos este grau de liberdade de tal forma que a posição da

origem do seu sistema de referências fica no ponto de cruzamento da reta vertical central do seu corpo com o plano horizontal definido pela estrutura da sua base, veja a **figura 1.3**.

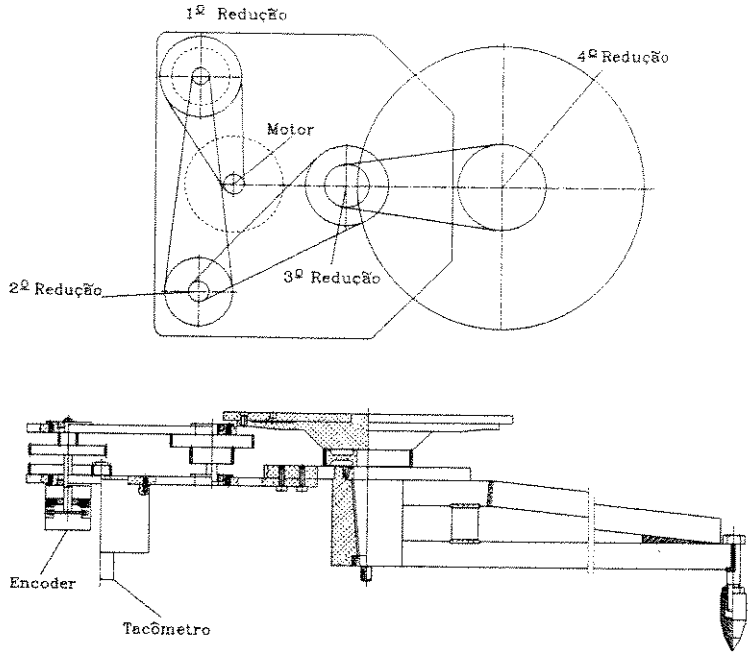


Figura 1.2 - Vista do Sistema de Redução da Junta 1.

Figura 1.3 - Vista do robô **JECAII**.

♦ *2º GRAU DE LIBERDADE*

O primeiro grau de liberdade possibilita na verdade um movimento de cintura para o robô, o 2º grau é o que faz o movimento do braço, se feita uma analogia com um braço humano. Ele é acoplado, conforme explicamos acima, ao 1º grau, ficando seu motor, seu sistema de redução, e seu eixo de giro, montados sobre o corpo do 1º grau. Ele é rotacional, e permite ao robô uma excursão de movimentos de 260°. Seu corpo é feito em alumínio e possui um comprimento de 400mm, em sua extremidade próxima ao eixo de giro, ficam montados os sistemas motor-redutores dos 3º, 4º, 5º e 6º graus de liberdade (**figura 1.4**), e em seu corpo ficam também montados alguns sensores dos demais graus de liberdade conforme explicamos mais adiante, e por ele passam os sistemas de transmissão por cabos dos demais graus.

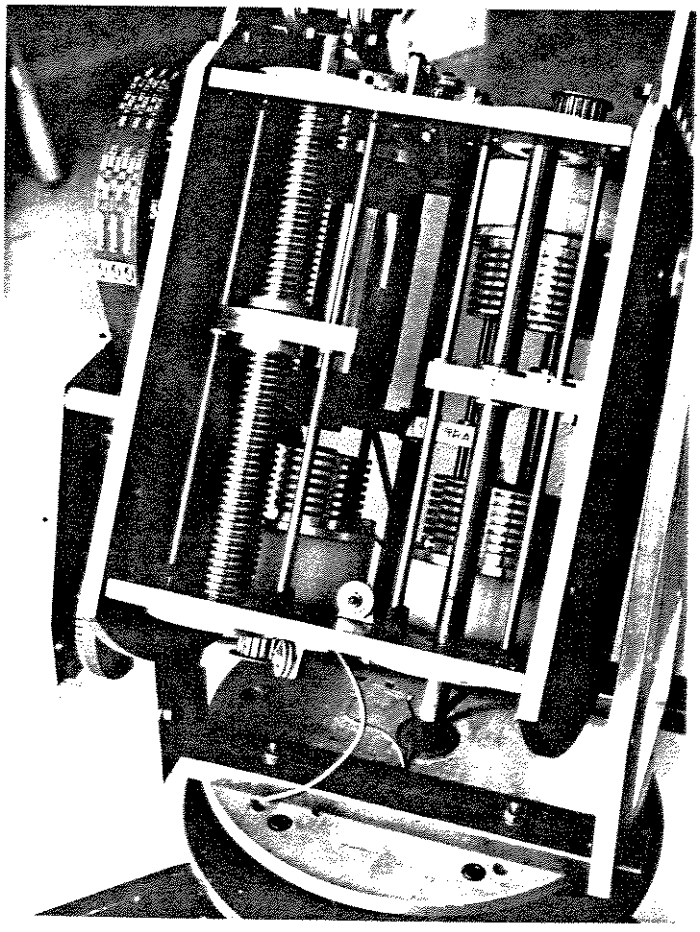


Figura 1.4 - Vista da Junta 2 do robô **JECAII**.

Em função do motor utilizado e da velocidade tangencial desejada na garra, projetamos seu sistema de redução de velocidade conforme o esquema mostrado na **figura**

1.5 com fator de redução de 160:1, isso para que este movimento seja executado com maiores forças, pois ele sofre os efeitos da gravidade, ao contrário de 1º grau.

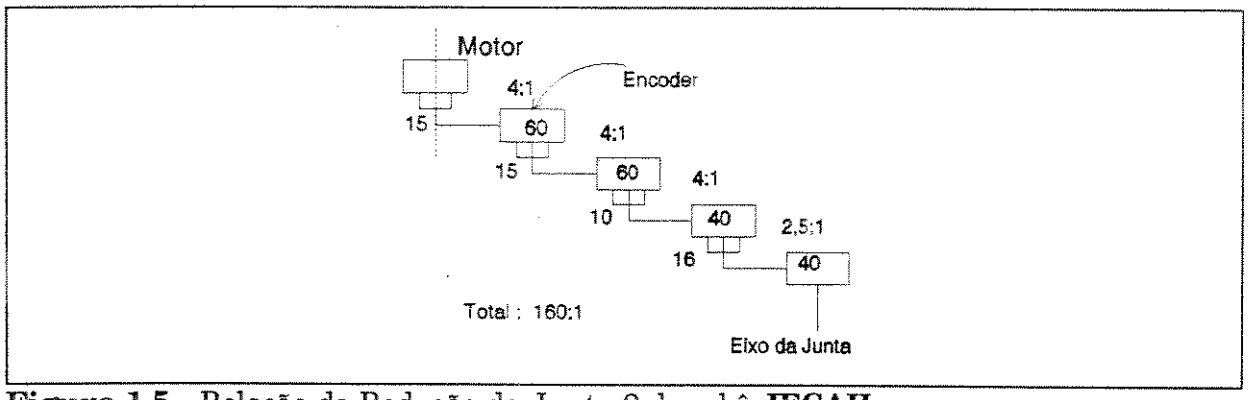


Figura 1.5 - Relação de Redução da Junta 2 do robô **JECAII**.

- ***3º GRAU DE LIBERDADE***

O terceiro grau de liberdade, na analogia com o corpo humano, permite ao robô o movimento principal de um ante-braço. Ele é acoplado ao segundo grau de liberdade através de uma junta (seu eixo de giro), denominada cotovelo.

Seu corpo também é construído em alumínio, e possui várias características semelhantes ao 1º e ao 2º graus. Ele possui um comprimento de de 300mm, e dá ao robô a possibilidade de uma excursão de movimentos de 260°.

Conforme explicado anteriormente, seu motor, seu sistema de redução de velocidade e transmissões são construídos sobre o 2º grau de liberdade. e em função do motor utilizado e da velocidade tangencial desejada na garra, projetamos seu sistema de redução de velocidade com relação de 152,9:1 conforme o esquema da **figura 1.6**.

Visando um pré balanceamento de parte da estrutura, que sofre influência da gravidade, o motor e a parte do sistema de redução deste grau de liberdade são colocados de forma a contrabalançar o peso de toda a estrutura do braço, no lado oposto ao braço em relação ao eixo de giro do segundo grau de liberdade.

O sistema de redução de velocidade neste caso, possui um limitante, que é o espaço disponível para sua montagem, e portanto necessita de um projeto bem mais sofisticado. As distâncias entre os eixos das relações intermediárias de redução devem ser as menores possíveis, e para que isso aconteça, como elas são funções dos diâmetros das polias e dos comprimentos das correias sincronizadoras, a melhor opção é adquirir os menores tamanhos possíveis para as forças que devem suportar.

Para fazer a transmissão dos movimentos deste sistema de redução até o eixo de movimentos do grau de liberdade, optamos por utilizar um sistema de transmissão por cabos de aço para que também consigamos bastante precisão, isso é feito através de

correias e polias sincronizadoras, que transmitem primeiramente o movimento rotacional do motor para o fuso de passo 4mm acoplados ao corpo do braço. Este fuso transforma o movimento rotacional em movimento linear com auxílio de um arrastador que o percorre balizado por duas guias para mantê-lo no curso correto dentro de uma excursão de 220mm.

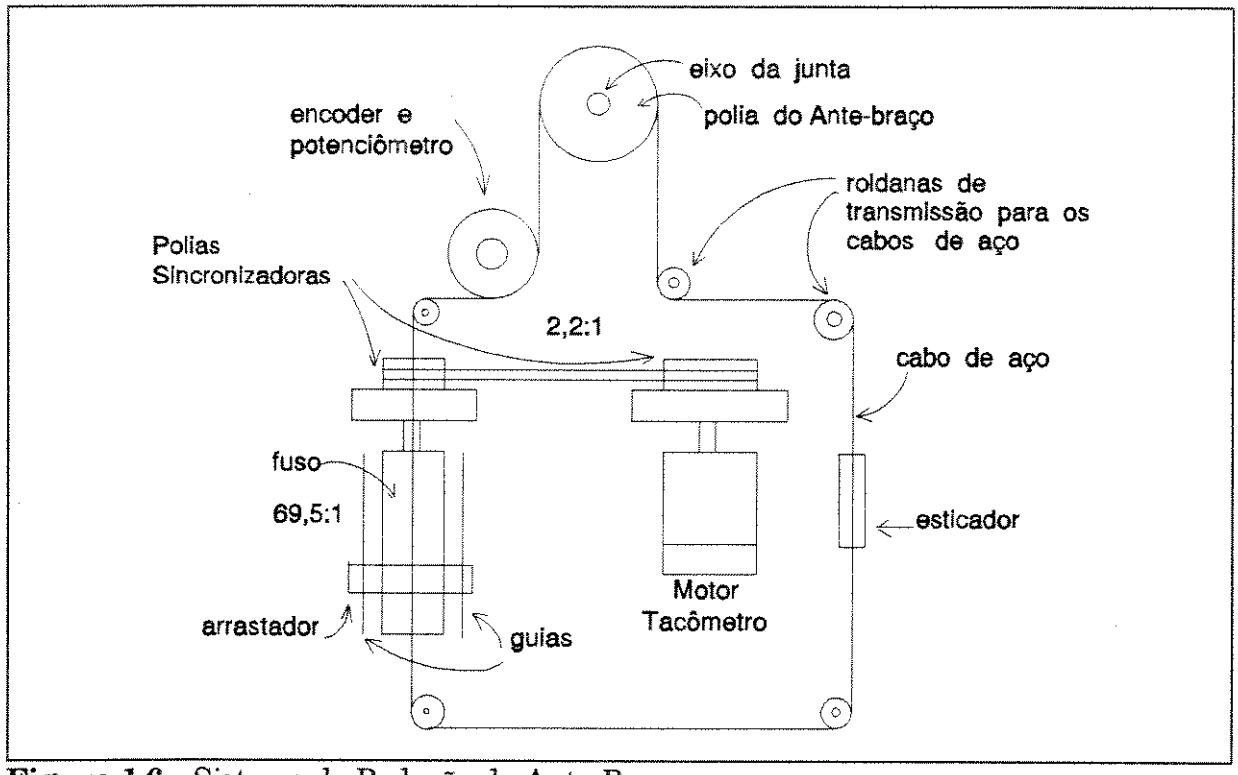


Figura 1.6 - Sistema de Redução do Ante-Braço.

Os cabos de aço que transmitem o movimento do fuso até a polia solidária ao eixo da junta são fixados neste arrastador, e conduzidos por roldanas estrategicamente distribuidas na estrutura do braço. Este sistema de transmissão também possui um esticador para os cabos de aço, como pode-se ver no esquema da **figura 1.6**. Este recurso é importante, pois permite que ajustemos a tensão dos cabos em um ponto ótimo para a transmissão. As roldanas são construídas em latão, e giram suportadas por rolamentos. O cabo deve ser bobinado em ambas as bobinas para que não ocorram deslizamentos, fato que causaria imprecisão nos movimentos, e aproveitamos o próprio caminho dos cabos para fazer a conexão dos sensores *encoder* e potenciômetro da junta.

1.2.2 ☞ MECANISMO DE ORIENTAÇÃO E PREENSÃO DE OBJETOS

♦ *4º, 5º, E 6º GRAUS DE LIBERDADE*

Estes três graus de liberdade estão colocados em um mesmo ítem porque foram construídos de forma paralela. São os graus de liberdade do punho e da garra, ou

seja, o 4º e o 5º graus responsáveis pela orientação da garra (punho), e o 6º grau a própria garra (**figura 1.7**).

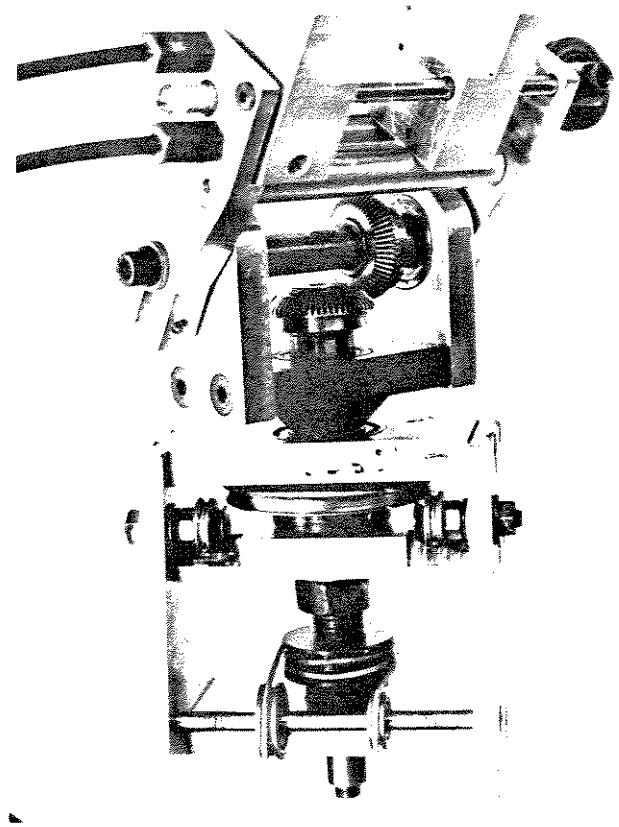


Figura 1.7 - Vista dos Graus de Liberdade Para Orientação e Garra do **JECAII**.

Projetamos seus sistemas de redução e transmissão de movimentos, de tal forma que os respectivos motores responsáveis por eles fiquem montados na estrutura do 2º grau de liberdade, para que minimizemos os efeitos de suas massas durante os movimentos do robô. Em função deste critério optamos por fazer as transmissões destas juntas através de cabos de aço. O sistema de redução de velocidade destas juntas possui características semelhantes do responsável pelo 3º grau. Cada motor através de uma única relação de redução feita com duas polias e uma correia sincronizadora movimenta um sistema baseado em um fuso, no qual os movimentos rotacionais dos motores são transformados em movimentos translacionais, por uma determinada relação. E nesses fusos estão colocados cursores arrastadores onde as extremidades dos cabos são afixadas. Estes cabos são conduzidos, através de roldanas, e de estruturas para manter os mesmos esticados em qualquer posição que o robô se encontre, até os respectivos eixos de movimento de cada um destes graus de liberdade (**figura 1.8**).

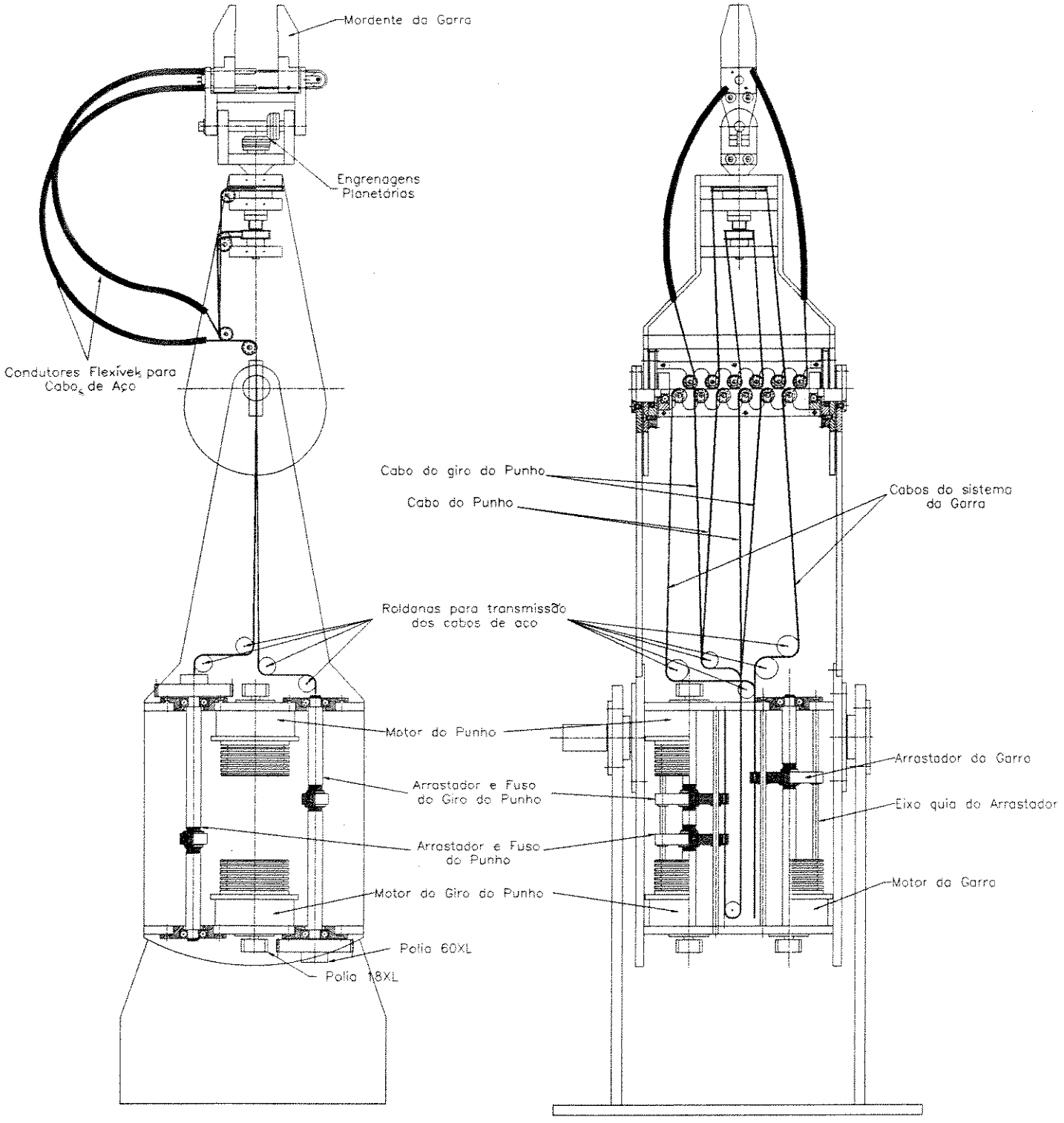


Figura 1.8 - Sistema de Transmissão e Redução dos 4º, 5º e 6º Graus de Liberdade.

○ 4º GRAU DE LIBERDADE

Este grau de liberdade permite que o robô possa fazer movimentos de rotação na direção do eixo do corpo do terceiro grau de liberdade (*ROLL*). Ele permite uma excursão de até 320° em torno do eixo citado. Seu corpo possui um comprimento de 150mm e é construído em alumínio. Devendo ser acoplado na extremidade do 3º grau e suportar a fixação, sobre seu corpo, do 5º grau. O esquema do sistema de redução de velocidade e transmissão é semelhante ao do utilizado para a junta três (**figura 1.6**) apenas que, para este grau de liberdade a relação de redução é de 236,16:1. Com relação entre as polias sincronizadoras de 3,33:1 e entre polias e fuso de 70,92:1, sendo que o cursor arrastador onde os cabos de aço estão fixados tem um curso de 100mm.

Os cabos de aço após transmitidos adequadamente, deste sistema de redução até o eixo de movimentos do grau de liberdade, são bobinados em uma roldada que gira sobre este eixo e está solidária ao corpo do grau de liberdade.

○ 5º GRAU DE LIBERDADE

O quinto grau de liberdade, permite ao robô executar movimentos perpendiculares ao 4º grau (*PITCH*), assim, com esses dois movimentos é possível orientar parcialmente a garra nas direções partindo do ponto central do corpo do 4º grau de liberdade e que passam por pontos de uma esfera com centro neste ponto e de raio igual ao comprimento do corpo do quinto grau mais a distância até o centro da garra. Sua estrutura é montada sobre o quarto grau, e feita em alumínio. Seu motor, seu sistema de redução de velocidade e suas transmissões são totalmente semelhantes às do quarto grau de liberdade, mas existe grande diferença na estrutura que transmite os movimentos do quarto grau para seu eixo de movimentos, a qual é baseada num sistema de engrenagens diferenciais que transfere os movimentos para um eixo perpendicular a esta, conforme citado anteriormente. O fator de redução desta transmissão é de 118,08:1. Com relação entre as polias sincronizadoras de 3,33:1, e entre polias e fuso de 35,46:1, sendo que o cursor arrastador tem um curso de 50mm.

O punho do **JECAII** não tem o movimento angular (*YAW*) por dificuldades encontradas para a sua implementação, principalmente com relação a custos. Entretanto, na maioria das aplicações, o volume de trabalho atingido para a orientação da garra com os dois movimentos angulares acima descrito é satisfatório para os nossos objetivos.

○ 6º GRAU DE LIBERDADE

O sexto grau de liberdade é a própria garra, projetada de forma a poder dar ao robô a possibilidade de prender e soltar objetos de diversas formas e tamanhos, bastando para isso que se anexem diferentes pinças ou ferramentas em suas extremidades. Ela é construída em aço e alumínio, tem basicamente dois dedos móveis, e uma excursão de movimentos de 50mm. Seus dedos tem um comprimento aproximado

de 40mm, modificáveis por anexação de outras ferramentas. Seu sistema (motor/redutor de velocidade/transmissão) é muito semelhante aos sistemas dos quarto e quinto graus de liberdade. A maior diferença é que como seu movimento é translacional, a extremidade onde no quarto e quinto graus os cabos de aço bobinam roldanas para finalizar a tranmissão não existem, sendo que neste caso as extremidades dos cabos de aço que transmitem seus movimentos são fixadas diretamente nos dedos da garra. Do terceiro grau de liberdade partem dois guias para estes cabos que os conduzem até às roldanas que servem para guiá-los adequadamente aos dedos, colocadas na própria estrutura da garra. Ver **figura 1.8**. Citamos as referências **[55][75]** para maiores detalhes sobre a modelagem matemática e a construção de garras.

Como explicamos, as transmissões realizadas por cabos de aço para os graus de liberdade 4, 5 e 6 devem passar pela estrutura de todo o braço até chegarem aos seus respectivos eixos de movimento. Estas transmissões são guiadas por sistemas de roldanas que asseguram a passagem dos cabos sem que sofram variações de tensões enquanto os outros graus de liberdade se movimentam, pois, caso contrário, haveria a necessidade de movimentos de compensação por parte de cada uma destas juntas, o que tornaria seu controle muito mais complexo ainda. Estas estruturas são complexas e exigem construção de peças mecânicas com muitos detalhes e de grande precisão, principalmente porque devem possibilitar uma perfeita concordância entre pontos de tangência formados pelos cabos, roldanas e os eixos de movimentos do robô. Para que as tensões sobre toda a estrutura do robô, causadas pelos cabos sejam mínimas, resolvemos projetar o sistema de suas transmissões de tal forma que todos estejam colineares com uma reta perpendicular aos eixos de movimento dos 2º e 3º graus de liberdade (**figuras 1.9 e 1.10**). O fator de redução para as transmissões dos movimentos da garra é de 119,16:1 com relação entre as polias sincronizadoras de 6:1 e entre polias e fuso de 19,86:1, sendo que o curso do arrastador onde são fixados os cabos é de 28mm. Também foi necessária a construção de esticadores para cada par de cabos destas transmissões, de forma a poder-se tensioná-los para um ponto ótimo de funcionamento, e sua respectiva colocação na estrutura do robô.

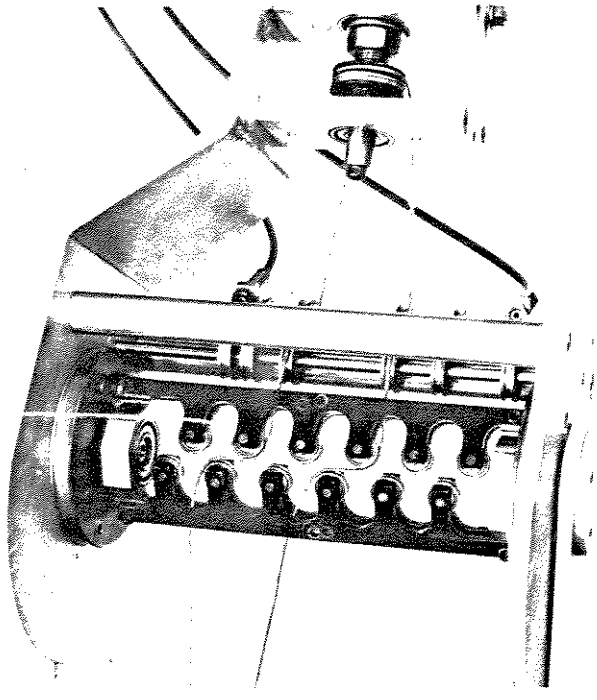


Figura 1.9 - Detalhe do Sistema de Transmissão Para os 4º, 5º, e 6º Graus de Liberdade.

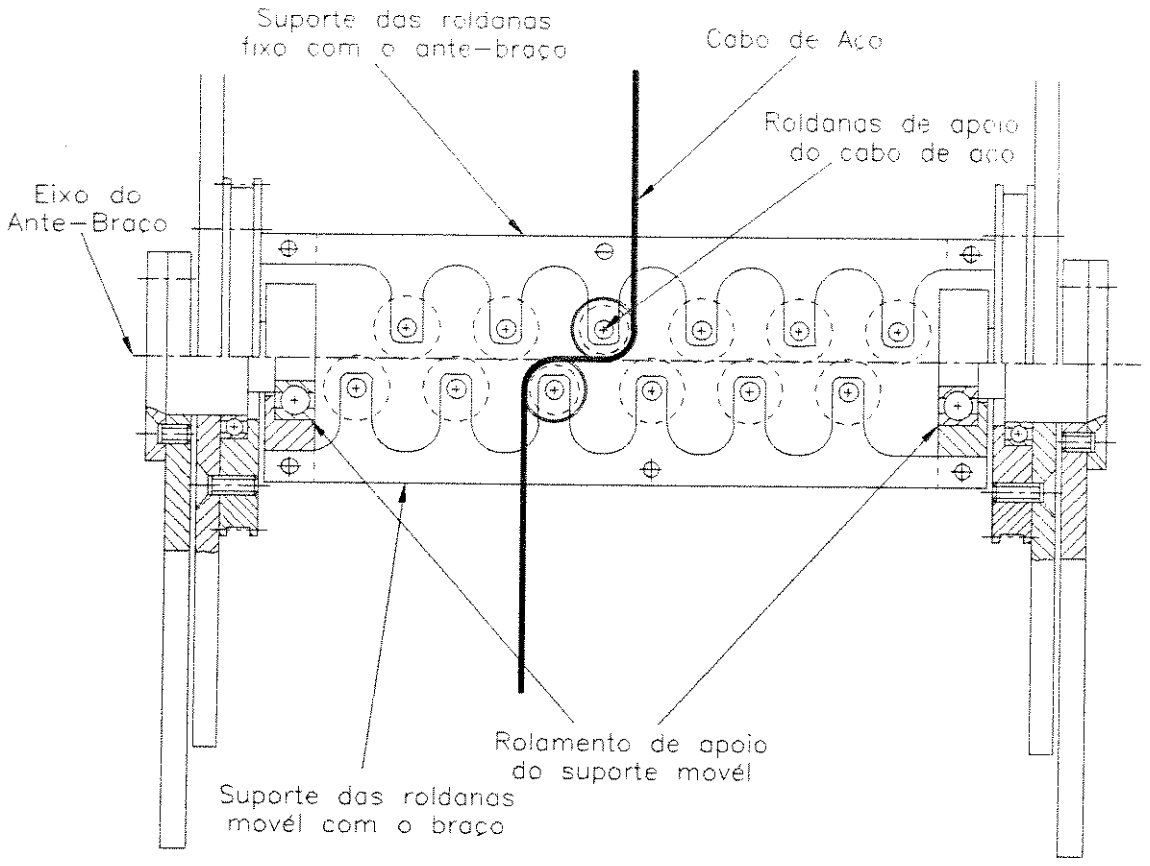


Figura 1.10 - Sistema de Transmissão dos Cabos de Aço no Centro do Eixo do Ante-Braço.

A **figura 1.11** mostra um exemplo de aplicação para o robô.

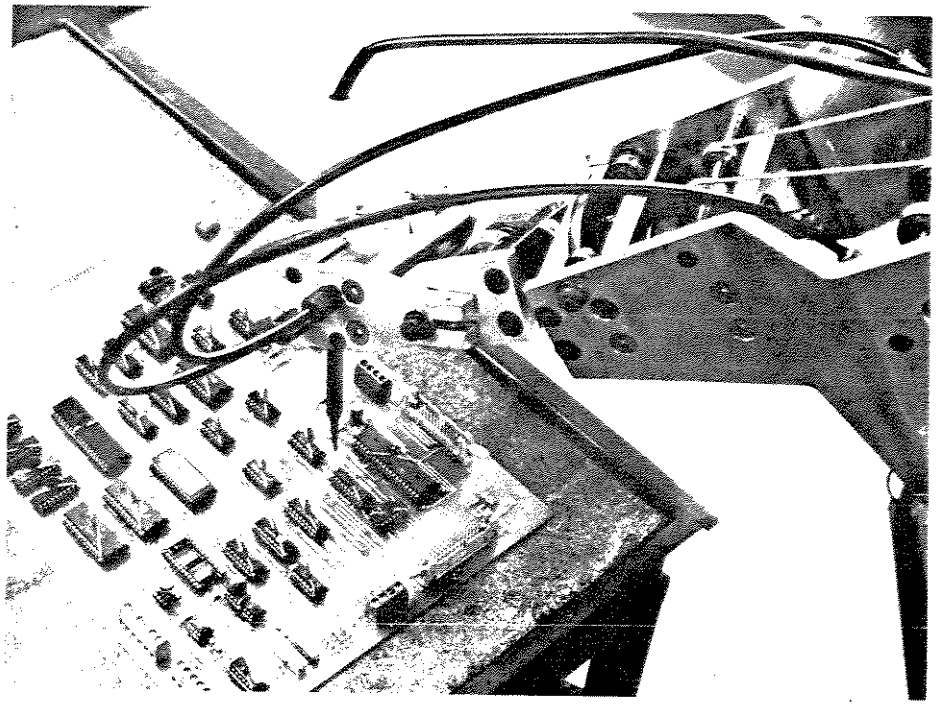


Figura 1.11 - Exemplo de Aplicação Para o Robô **JECAII**.

1.3 ☞ DISPOSIÇÃO DOS SENSCRES NA ESTRUTURA MECÂNICA

Utilizamos um *encoder* de 1000 pulsos/rotação na primeira junta do robô, primeiramente fizemos a suposição de que ele seria acoplado ao 3º eixo, então existe uma relação de redução de 8:1. Para 280° de giro no 5º eixo o 3º eixo gira 2240° ~6 voltas, isso provocaria uma contagem de 6222 pulsos pelo *encoder*. Lembrando que o braço todo esticado possui um comprimento de 820mm. Então a resolução espacial para 280° de curso vista do efetuador será:

$$\Delta d = \frac{4,8869}{6222} \times 820 = 0,6440mm$$

que ainda pode ser minimizada. A seguir fizemos a suposição de que este mesmo *encoder* estivesse acoplado no 2º eixo, portanto existe agora uma relação de redução de 32:1, e para 280° de giro no 5º eixo, o 2º eixo gira 8960° ~25 voltas, o que equivaleria a uma contagem de 24888 pulsos pelo *encoder*, e a resolução para os 280° de curso vista do efetuador seria de 0,1610mm.

Se fosse feita a suposição de que o *encoder* estivesse acoplado no eixo da última engrenagem (5º eixo), junto ao elo, para uma extensão de giro de 0° até 280°, o *encoder* contaria 777 pulsos, e a precisão por pulso para este curso seria de ~5,15mm, que pioraria a resolução. Porisso há a necessidade de posicionar o *encoder* mais afastado do eixo da junta.

Se o *encoder* fosse acoplado ao eixo do motor, então a relação de redução a considerar é de 128:1, para 280° de giro no 5º eixo, o 1º eixo gira 35840°, e o encoder contaria 99555 pulsos, o que extrapola a contagem possível do contador/ registrador de 16bits projetado para interpretar seus sinais e informar cada Escravo do *hardware* eletrônico projetado para controlar o sistema (veja **Anexo A**). Então, como sugestão, para otimizar ao máximo esta contagem possível e portanto a resolução, pode-se fazer um multiplicador de 1:2 do 2º eixo da engrenagem de redução para o *encoder*; neste caso, a relação de redução para ele seria de 64:1, o eixo do *encoder* giraria 17920° (~50 voltas) para uma excursão de 280° no eixo da junta, e portanto contaria 49777 pulsos com precisão por pulso vista do efetuador de 0,0805mm, que seria a melhor opção.

Os cálculos para definir a alocação dos *encoders* para todas as juntas seguem o mesmo raciocínio visto anteriormente. Como estamos utilizando engrenagens feitas com polias e correias sincronizadoras, temos a garantia de que estas precisões são garantidas pelas transmissões mecânicas, e portanto podemos alocar os *encoders* para acoplagem em quaisquer eixos das engrenagems de redução de acordo com os valores calculados.

Supondo que o motor de uma junta gire até uma velocidade de 1500rpm, para o caso citado o *encoder* daria 1/2rpm para cada rpm do motor, isso equivaleria a 12,5rps. Como o *encoder* suposto dá 1000 pulsos/rotação, a freqüência para este caso seria de 12,5KHz. Logo os componentes eletrônicos que fazem a interpretação dos sinais do *encoder* devem ser capazes de processarem sinais com estas freqüências.

1.4 ☞ ESTRUTURA ELETRÔNICA

Projetamos o sistema de controle eletrônico para o robô **JECAII** tendo como principal objetivo permitir o máximo de flexibilidade de programação para que o robô possa executar uma ampla gama de tarefas industriais, visando ainda ser um projeto didático, que sirva para o ensino da robótica, e com baixo custo de produção. Esse é um objetivo muito difícil de ser atingido, porque uma maior flexibilização implica em maiores custos de desenvolvimento e produção. Então estabelecemos como critério utilizar componentes eletrônicos facilmente encontráveis no mercado brasileiro que permitissem a construção de circuitos digitais microprocessados com capacidade de processamento de sinais adequados para resolver problemas de controle inerentes às estruturas de robôs com até seis graus de liberdade e acionados com motores de corrente contínua controlados por armadura, construídos em módulos com a possibilidade de se utilizar vários tipos de microprocessadores dependendo do tipo de algoritmos que serão processados. Por exemplo, para uma junta que requer uma estratégia de controle mais robusto com algoritmo mais complexo, poderá ser utilizada uma placa Escrava baseada em microprocessadores mais poderosos, tipo *80286*, *80486*, etc..., ou inclusive com utilização de processadores digitais de sinais, *DSP's*. E em juntas menos críticas onde os algoritmos de controle podem ser mais simples utilizamos microprocessadores Z-80 que são baratos e eficientes para este tipo de aplicação.

Adotamos uma arquitetura eletrônica composta por dois níveis hierárquicos, onde o nível hierarquicamente superior é o responsável pelo processamento de sinais de gerenciamento e controle de trajetórias do robô, e o nível inferior responsável pelo controle direto das variáveis de juntas. Por simplicidade, para o nível superior, fizemos a adaptação de um microcomputador compatível IBM-PC 386DX /40MHz para esta finalidade, principalmente porque para a maioria das funções a serem processadas para o controle do robô, esta máquina possui capacidade adequada para os tempos exigidos pela dinâmica do sistema, e além disso, possuem um projeto padrão amplamente difundido e conhecido nos meios científicos e industriais, fato que facilita a outros usuários operarem este sistema de forma muito mais rápida. Então todas as interfaces também desenvolvidas para estes tipos de máquinas podem ser utilizadas, como por exemplo, um teclado padrão, monitores de vídeo, impressoras, etc. Em nosso projeto chamamos este nível de controle de Micro Mestre.

Compusemos o nível inferior da arquitetura com seis subsistemas microprocessados, cada um com capacidade de realizar o controle de um dos servomecanismos de juntas, prevendo capacidade para controlar posição, velocidade e aceleração, e mais um subsistema de aquisição de dados, também microprocessado, com finalidade de processar sinais de sensores, e atuar em circuitos externos instalados no ambiente onde o robô se movimenta. Estes subsistemas foram baseados originalmente em microprocessadores Z-80H, que são componentes de fácil aquisição no mercado, e muito conhecidos nos meios científicos e industriais, por permitir que se construam controladores com arquitetura eletrônica simples, ter boa capacidade de processamento, e de fácil programação. O nível inferior foi concebido assim, para permitir o processamento paralelo de todos estes subsistemas, uma vez que o robô é um processo que pode ser paralelizado, e com isso aumentamos sua potencialidade de trabalho. Chamamos cada um destes subsistemas de controlador Escravo, porque trabalham sob a supervisão do Micro Mestre, e chamamos de subsistema digital de entrada e saída ao subsistema encarregado de processar sinais do ambiente externo ao robô.

Adotamos um padrão de comunicação entre o Mestre e os Escravos, utilizando um barramento de linhas físicas em paralelo, com conectores apropriados para que todos os Escravos possam receber e transmitir sinais, e denominamos Barramento Principal de Comunicação. Este barramento possui um circuito de amplificação de sinais, para que a relação sinal/ruído seja maior, e a probabilidade de erros nas comunicações seja diminuída.

Também foi necessário projetar e construir um circuito de interface entre o Mestre e os Escravos, que é responsável pelo selecionamento e direcionamento de dados que transitam pelo barramento de comunicação nos dois sentidos, Micro Mestre → Escravos, e Escravos → Micro Mestre. Chamamos este circuito de Interface Micro Mestre/Escravos.

Como foi dito, cada Escravo é responsável pelo processamento do algoritmo de controle direto de juntas, e processamento dos sinais de sensores, gerando os sinais para o sistema de acionamento, e para isso existem circuitos apropriados que fazem a adequação de sinais. A técnica de acionamento escolhida é a Modulação por Largura de Pulsos (*PWM - Pulse Width Modulation*), e para realizá-la sobre cada motor existe um circuito de geração de sinais digitais controlado pelo respectivo Escravo, e um amplificador de potência baseado em topologia ponte H, colocado em cascata para suprir potência adequada. Para o processamento dos sinais dos sensores particulares de cada junta, existem circuitos de filtragem de ruídos e compatibilização de níveis dos sinais, para que cada Escravo possa receber a informação precisa.

Esta arquitetura juntamente com um *software* de operação que coordena a utilização do barramento compartilhado pelos Escravos, permite o remanejamento de algoritmos independentes para cada junta através de instalação nos respectivos Escravos com apoio de um pacote de *software* de Sistema de Comunicação Homem-Máquina (SCHM) via teclado do Mestre. Desta forma o sistema torna-se bastante flexível no que diz respeito a substituição de algoritmos de controle independentes de juntas quando se deseja mudar o tipo de aplicação para o robô, ou quando para o mesmo tipo de aplicação se quer especificações de desempenho mais estritas.

Isso tudo foi previsto no projeto, porque intencionamos que o robô possa ser utilizado em diversas aplicações com requisitos de desempenho que exigem estratégias de controle completamente distintas para cada caso, no que diz respeito a capacidade de processamento do *hardware* eletrônico, maior ou menor complexidade dos algoritmos de controle, que em muitos casos podem ser fixos e com tratamento por técnicas lineares monovariáveis e chegando-se até aos casos em que devem ser tratados por técnicas de controle não-linear e multivariáveis, capacidade de carga e flexibilidade da estrutura mecânica com relação aos acoplamentos dinâmicos e maior atingibilidade do volume de trabalho, capacidade de maior ou menor resolução, precisão e repetibilidade, etc...

Um robô projetado para atender todos estes requisitos certamente teria um custo elevado para uma empresa que o utilizaria em aplicações com requisitos de desempenho menos estritos, e além disso podemos considerar que em alguns casos existem os dois tipo de exigências, ou seja, num Sistema Flexível de Manufatura, onde um mesmo robô realiza várias tarefas em períodos diferentes, pode haver a necessidade de alterações na sua estrutura, tanto de *software* como de *hardware* eletrônico ou mecânico a fim de atender as novas especificações de desempenho exigidas pela nova tarefa. Estas podem ser necessárias em uma parte da estrutura ou na sua totalidade. A solução menos dispendiosa para a empresa que pretende utilizar robôs na sua linha de produção seria

ter a possibilidade de encontrar um robô construído em módulos projetados para utilização em campos de aplicações que não implicam em grandes variações de desempenho quando operando numa determinada configuração. O usuário compraria somente os módulos que permitissem configurar o robô adequadamente para atender os requisitos dos diversos campos de aplicações da sua demanda trabalho e que seriam previamente discriminados no projeto de um robô especialmente projetado de forma modular para executar múltiplas tarefas. E porisso toda a arquitetura de *hardware* do robô **JECAII** está pensada para permitir esta modularidade, que deve ser acompanhada por pacotes de *software* convenientes para as adaptações. Como foi dito, nós utilizamos os microprocessadores Z-8ØH para basear as arquiteturas dos subsistemas Escravos, mas quaisquer outros microprocessadores podem ser utilizados, pode-se construir inclusive subsistemas mistos, com diferentes arquiteturas de controladores em função das necessidades de controle específicas de cada junta, pois dependendo do tipo de junta e da função que deve executar, seu controle pode ser mais complexo ou mais simples, e nestes casos uma arquitetura mista e modular pode tornar o sistema global de custo menor. Como existe uma padronização do barramento de comunicação paralela entre o Mestre e os Escravos, basta que o projeto destes Escravos respeite esta padronização. Neste barramento de comunicação existem conectores apropriados para fazer-se a conexão individual de cada Escravo, e portanto, os câmbios nestes controladores podem ser facilmente realizados.

Para realizar a transferência de dados entre o Mestre e os Escravos optamos por utilizar comunicação paralela com palavras digitais de 8bits por tranferência através do formato INTEL INTELLEC 8/MDS para garantir a transferência correta. Utilizamos técnicas de interrupção para que o Mestre possa tranferir dados diretamente para as memórias *RAM's* dos Escravos, ou ler dados da memória destes controladores. O processo pode ser individual para um controlador específico, para mais de um controlador, ou via comunicação *broadcasting* para todos os controladores ao mesmo tempo. Isso é possível porque cada Escravo possui um endereço específico dentro do sistema que é definido através do posicionamento adequado de chaves tipo *Dip Switch* presentes em cada um. Todas as transferências podem ser realizadas através de comandos gerados por programação de alto nível desde o Mestre, apenas respeitando o endereçamento correto definido pelo *hardware*, e a placa de Interface do Mestre/Escravos é quem interpreta e direciona os pedidos de interrupções adequados, e realiza as transferências de dados de forma bidirecional. Desta forma, todos os algoritmos de controle podem ser desenvolvidos e testados no próprio Mestre e enviados convenientemente para os Escravos pelo usuário. E também o usuário pode ter acesso aos dados do processamento dos Escravos para realizar análises de desempenho.

Nos **Anexos A, B, e C** estão comentadas com maiores detalhes as partes que compõe esta arquitetura eletrônica hierárquica que projetamos para o controle eletrônico do **JECAII**.

SEGUNDA PARTE - SOFTWARE

1.5 ☞ ENGENHARIA DE PROGRAMAÇÃO QUE OPERACIONALIZA O JECAII

Um sistema baseado em multiprocessadores jamais pode ser concebido sem levar-se em consideração a interação entre o funcionamento dos componentes eletrônicos a utilizar, e a programação para fazê-los operar de acordo com a aplicação e as potencialidades que se quer dar ao sistema. Um mesmo sistema microprocessado pode ser utilizado para realizar tarefas muito simples, ou então muito complexas, geralmente tanto a parte do circuito físico eletrônico como a programação influem para compor a capacidade final de processamento do sistema, mas é a programação que pode colaborar mais para que o custo final do projeto seja menor.

Ao iniciar um projeto desta natureza, deve-se ter claro quais as necessidades que somente podem ser oferecidas por construção física, por exemplo, freqüência de trabalho, tamanho da memória para armazenamento de dados, número de unidades de entrada e saída de dados para a comunicação entre o sistema e seu exterior, e tipo de geração de sinais de sincronismos para coordenar o funcionamento interno e a tranferência de dados entre seus componentes. Quando estes aspectos estiverem definidos, então é a programação que irá definir a capacidade total do seu processamento. Quando o mesmo sistema possui mais de um microprocessador para processar as informações, existe a necessidade de fazer-se um interfaceamento físico entre eles e uma programação especialmente projetada para isso.

Nós dotamos o *hardware* eletrônico do robô **JECAII** de componentes que permitem uma programação bastante flexível, e que globalmente pode dar uma grande capacidade de processamento para processar dados necessários ao controle de movimentos de um robô. O projeto deste *hardware* com a descrição de suas potencialidades está apresentado em detalhes no **Anexo A**, e neste ítem fazemos a apresentação da engenharia de programação que propusemos para a coordenação global do sistema, para tornar fácil seu manuseio e para otimizar a utilização de todos os seus recursos.

Para que um robô com as características do **JECAII** possa funcionar adequadamente, existe a necessidade de uma programação bastante complexa. Como o sistema em conjunto possui muitos setores que devem funcionar de forma sincronizada para compor os movimentos da estrutura durante a execução de uma tarefa, e dependendo da tarefa esse sincronismo pode variar, é a programação fica responsável pelo seu bom funcionamento.

Existe a necessidade de se transportar para a programação muitos aspectos que são conhecidos sobre a estrutura física do robô. E a forma que se utiliza para isso são os modelos matemáticos que podem ser facilmente programados. Existem dados, ou parâmetros, conhecidos sobre esta estrutura que são invariantes e podem ser armazenados em memória em forma de tabelas, e outros dados que são variantes de acordo com a evolução dos movimentos estruturais e tipos de esforços que a estrutura sofre durante estes movimentos, e para o tratamento desses dados, a programação é bastante mais sofisticada, pois de alguma forma ela deve dar a capacidade de interpretação adequada desses dados variantes, e fornecer soluções para manter a boa movimentação agindo de

forma a controlar o fluxo de energia nos acionadores da estrutura eletro-mecânica. Para esta etapa existem técnicas de controle de processos que também podem ser abordadas matematicamente, programadas e utilizadas para auxiliar o sistema a realizar este controle de fluxo de energia e informações.

A filosofia geral de controlar um processo, é definir quais as variáveis desse processo que serão controladas, então possuir uma forma de supervisionar a sua evolução no tempo, e um modo de injetar e retirar energia do processo de forma que este fluxo de energia possa conduzir cada variável a atingir os valores que se desejar no momento que se desejar. No caso do robô **JECAII**, e de muitos outros robôs, pode ser importantíssimo o controle de suas variáveis de posição, velocidade, aceleração, torque, vibrações, etc..., por este motivo incorporamos à estrutura sensores para medir os valores de cada uma dessas variáveis. Fazemos um procedimento indireto para supervisionar a evolução destas variáveis do elemento terminal da estrutura, que na realidade são as variáveis de maior interesse para a execução de uma tarefa pelo robô. Fazemos isso através de sensores colocados em cada um dos servomecanismos de juntas para medir a evolução individual destas variáveis em cada uma delas, e então compomos a evolução das variáveis do elemento terminal com a utilização dos modelos matemáticos que traduzem o conhecimento da estrutura para a programação.

Para a realização de qualquer tarefa deve existir um programa de coordenação, que é justamente o programa que processa estes modelos matemáticos devidamente traduzidos para a programação, de acordo com as exigências impostas pela tarefa em questão. Este programa de coordenação é dependente do tipo de controle que se quer realizar sobre a estrutura do robô, e para isso também existem várias técnicas que podem ser utilizadas. Normalmente esta coordenação envolve algum tipo de controle de trajetórias, que pode ser ponto-a-ponto ou contínua, e em programas mais sofisticados podem ser utilizadas técnicas para gerar ajustes com intenção de recuperar erros provocados por setores individuais do sistema durante os movimentos. Não é obrigatoriamente necessário que este programa de coordenação de movimentos seja escrito para realizar o procedimento de ação em tempo-real em função da evolução das variáveis, desde que o conhecimento sobre todo o processo de movimentos seja bem conhecido e muito bem traduzido para a programação. Neste caso pode-se fazer a geração dos dados necessários para a coordenação dos movimentos em uma etapa anterior ao processo de movimentar o robô, método este conhecido por geração *off-line* da coordenação dos movimentos. Mas de qualquer forma existe a necessidade de haver um programa que faz a amostragem adequada destes dados gerados anteriormente aos movimentos. Para o robô **JECAII**, a execução deste programa está pensada para ser feita pelo Micro Mestre, não importando se forem utilizados métodos *off-line* ou *on-line* para isso. Ou ainda, isso é realizado pelo nível hierarquicamente superior da arquitetura hierárquica de dois níveis que concebemos.

Os microcontroladores que compõem o nível hierarquicamente inferior da arquitetura fazem o controle de movimentos direto sobre as variáveis de cada servomecanismo de junta, e cada um tem um *hardware* eletrônico que pode processar programas de controle escritos em linguagem *assembler* devidamente inseridas em suas memórias em posições destinadas para isso. Fizemos todo o projeto prevendo que estes programas *assembler* pudessem ser escritos e testados no nível hierarquicamente superior, com auxílio de *softwares* específicos para este fim, para facilitar a programação aos usuários, uma vez que o desenvolvimento de programas *assembler* envolve métodos bastante mais sofisticados que a programação em linguagens de alto nível, e sem a

utilização de ferramentas de simulação, montagem, compilação e *linker*, criar-se-ia uma enorme dificuldade e dispêndio de tempo para sua composição. Assim, quando um usuário deseja desenvolver um programa de controle para o nível de controle direto sobre o servomecanismo de uma junta, ele deve apenas dominar a utilização de algum programa-ferramenta desses. No caso da utilização do robô em um ambiente industrial, talvez não haja a necessidade dos usuários terem que programar neste nível, eles podem apenas selecionar a utilização de programas desenvolvidos anteriormente, por programadores com maior nível de especialização, mas de qualquer forma garantimos que esta possibilidade exista.

Claramente em qualquer aplicação de uma máquina para a realização de uma tarefa, existe primeiramente uma etapa de planejamento, uma de criação, uma de testes, uma de operação prática, e uma de avaliação de desempenho para realimentar os dados na intenção de fazer-se melhoramentos. No entanto, depois que seja atingido um nível satisfatório de desempenho no trabalho da máquina para executar um tipo específico de tarefa, várias dessas etapas não são mais necessárias, e portanto poder-se-ia simplificar mais a programação e o *hardware* eletrônico, para máquinas que têm aplicações específicas. Se uma indústria tiver a necessidade de adquirir várias máquinas, pode-se apenas incrementar uma delas com capacidade para a realização de todas estas etapas, e as demais apenas para realizarem a operação prática baseada nos dados obtidos nas etapas executadas nesta outra etapa para desenvolvimento. Como criamos apenas um protótipo do robô **JECAII**, sua programação está pensada para realizar todas elas.

Lançamos a idéia de criar um pacote de *software* geral de supervisão de todo o sistema do robô, que pode ficar organizado dentro de um subdiretório na memória de massa do Micro Mestre, que pode ser desenvolvido de forma modular, com subprogramas destinados a aplicações específicas que podem ser invocados para processamento em função das necessidades impostas pelos usuários e pelos critérios de projeto do sistema. Deixamos sempre a possibilidade de acréscimo de novos subprogramas que possam vir a ser incorporados ao pacote básico, facilitando sua anexação. O diagrama hierárquico de funções da **figura 1.12** mostra em blocos como estão distribuídas estas funções da programação em nosso sistema.

1.5.1 ☞ HIERARQUIA DA PROGRAMAÇÃO

Para melhor entendimento do pacote de *software* que está sendo gerado para controlar o sistema mecatrônico, é conveniente explicá-lo por partes. Pode-se dividir os sub-programas em dois grupos de acordo com sua aplicação nos níveis hierárquicos que compõem o *hardware* de controle. Assim, os programas desenvolvidos em linguagem de alto nível que são executados pelo Micro Mestre, formam o primeiro grupo, e os programas desenvolvidos em linguagem de baixo nível que são executados pelos Escravos formam o segundo grupo. Desde o plano inicial optou-se por utilizar a *linguagem C* de programação para o desenvolvimento dos programas do primeiro grupo, e as facilidades do pacote *Turbo C*$^{++}$. Para o desenvolvimento dos programas do segundo grupo optou-se por utilizar pacotes de programas que oferecem a possibilidade de montar, compilar, *linkar*, *debugar*, e simular programas escritos em linguagem *assembler* de microprocessadores Z-80.

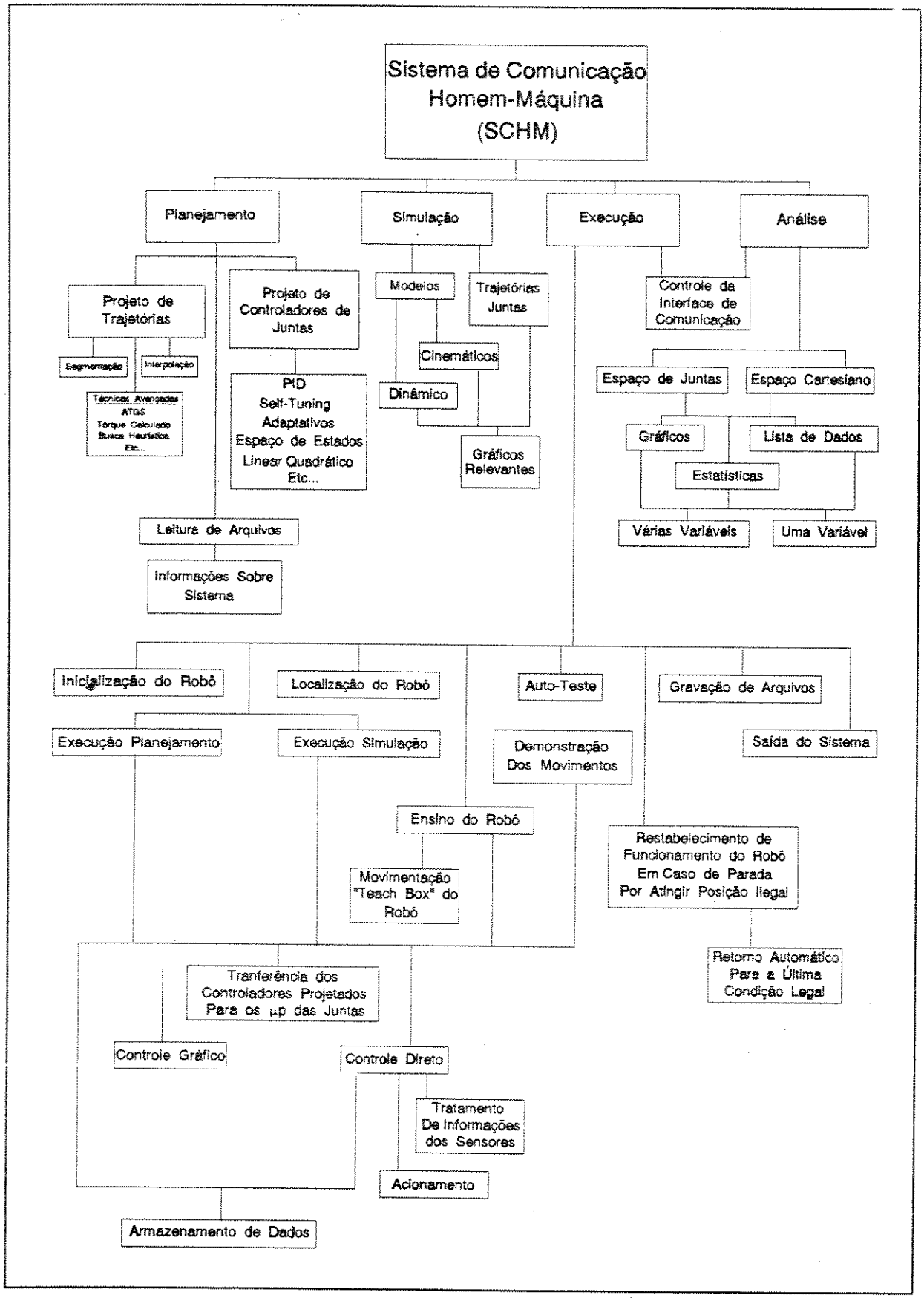


Figura 1.12 - Diagrama Hierárquico de Funções para a Operacionalização do **JECAII**.

♦ *PROGRAMAÇÃO PARA A COMUNICAÇÃO ENTRE O MESTRE E OS ESCRAVOS*

O projeto de *hardware* permite a comunicação entre o Micro Mestre e os Escravos sempre com a gerência do Micro Mestre com possibilidade de transferência de 8 *bits* por instrução tanto no sentido Micro Mestre → Escravos como vice-versa, através de acesso direto aos barramentos de dados e endereços dos microcontroladores de juntas, via pedidos de barramentos e aceitação de pedidos de barramentos. Cada microcontrolador de junta possui um endereço específico dentro do sistema global, e uma vez acessado, o Micro Mestre pode escrever ou ler dados tanto da memória como das unidades de E/S de cada controlador. Portanto, os programas do segundo grupo podem ficar residentes no disco rígido do Micro Mestre, e quando for conveniente ele pode transferí-los para as memórias *RAM's* dos Escravos. Essa possibilidade oferece grande facilidade de programação dos algoritmos de juntas a serem inseridos nos Escravos, e possibilita grande versatilidade durante a execução dos movimentos porque o Micro Mestre pode alterar a programação dos Escravos sempre que for conveniente.

♦ *PROGRAMAS DO SEGUNDO GRUPO*

Os programas desenvolvidos em linguagem *assembler* para serem executados nos Escravos devem estar aptos a gerenciar toda capacidade dos seus *hardwares*. Porisso documentamos todo o *hardware* com indicação de posições de memória *EPROM*, memória *RAM*, e endereços das unidades de E/S. Assim, toda vez que necessitamos acessar quaisquer destas partes, basta verificar qual seu endereço dentro do sistema. Também está prevista a inserção destes dados em um arquivo de Ajuda (*Help*) que poderá ser requisitado para auxílio à programação via Sistema de Comunicação Homem-Máquina (**SCHM**).

Uma vez que um programa executável pelos Escravos estiver na forma ideal, ele pode ser armazenado no Micro Mestre, e os programas do primeiro grupo poderão transferí-lo quando forem requisitados pelo operador, ou pela tarefa em execução.

♦ *PROGRAMAS DO PRIMEIRO GRUPO*

Dentro do pacote de programas do primeiro grupo também é conveniente fazer subdivisões para melhorar o entendimento. Como o objetivo deste trabalho é tornar o sistema didático, e com capacidade para realizar tarefas em ambientes industriais, é fundamental que o SCHM torne o sistema de fácil operação e compreensão por usuários de diferentes níveis de conhecimento sobre assuntos relacionados com a robótica. Para isso todo o desenvolvimento dos programas visa dar uma aparência, na tela do monitor do Micro Mestre, agradável ao operador, com o máximo de explicações *on-line* possível sem causar confusão.

Fazendo uma subdivisão em quatro blocos dentro dos programas do primeiro grupo podemos explicar a intenção de abranger as áreas importantes necessárias ao estudo e controle de um robô. Estes blocos são: Planejamento, Simulações, Execução e Análise (**figura 1.13**).

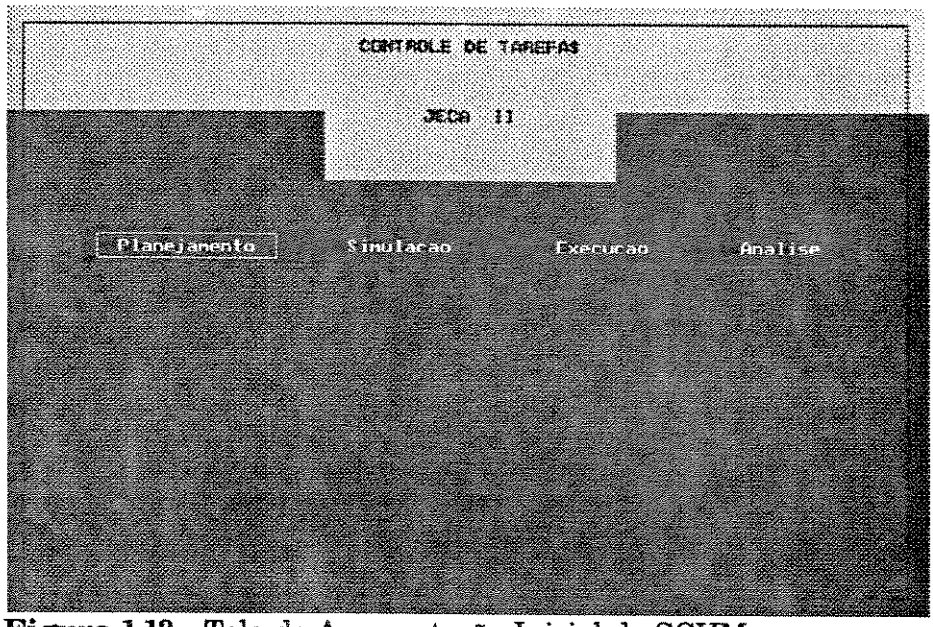


Figura 1.13 - Tela de Apresentação Inicial do SCHM.

No bloco Planejamento prevemos a possibilidade do operador poder selecionar se quer planejar o funcionamento das juntas ou uma tarefa, definir estratégias de controle multivariável, com possibilidade de ler arquivos de situações já planejadas anteriormente e armazenadas no disco rígido, e ainda um meio de obter informações sobre o sistema global do robô (**figura 1.14**).

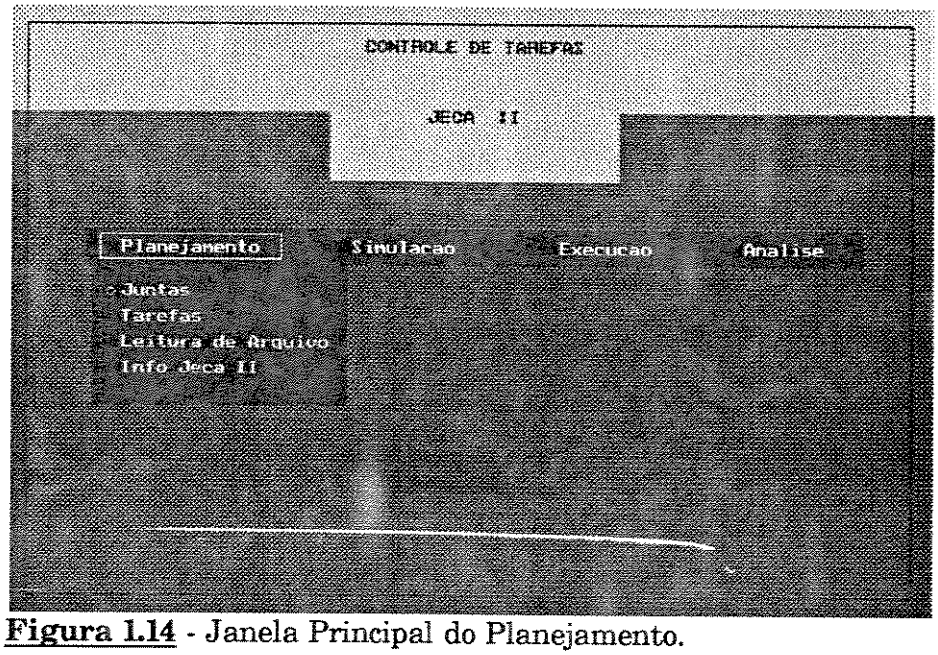


Figura 1.14 - Janela Principal do Planejamento.

Por exemplo, se o operador selecionar o planejamento de juntas, o sistema oferecerá nova possibilidade de selecionar qual a junta que deseja planejar, isto permite que ele possa planejar o funcionamento de cada junta de modo independente das demais. Uma vez que tenha optado por uma determinada junta, o sistema oferecerá a possibilidade de planejamento de algoritmos de controle tipo PD, PID, Controle Linear

Quadrático, etc... previamente estruturados dentro do pacote de programas (**figura 1.15**).

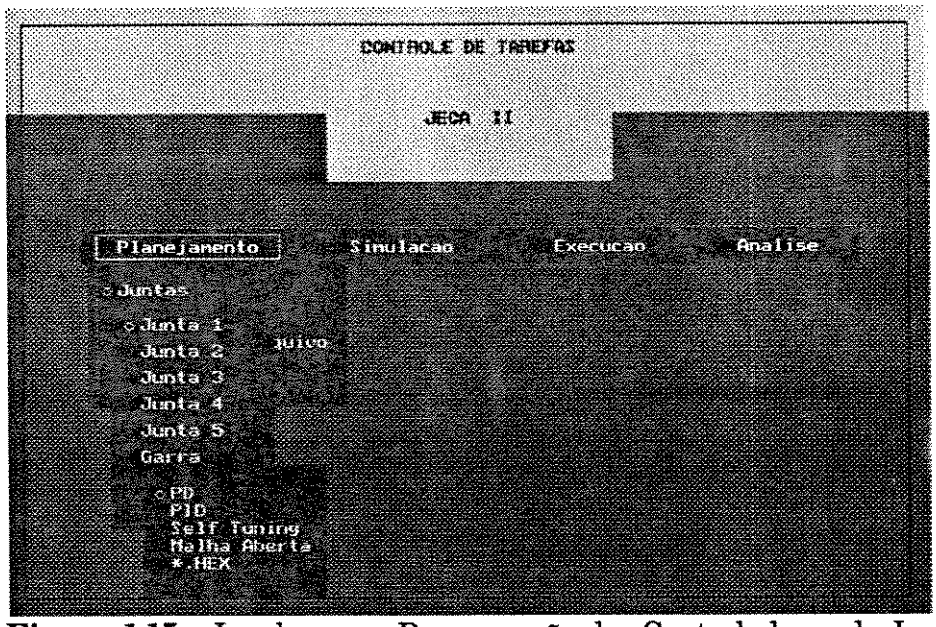


Figura 1.15 - Janelas para Programação dos Controladores de Juntas.

Então ele poderá indicar os valores para os parâmetros destes controladores (**figura 1.16**).

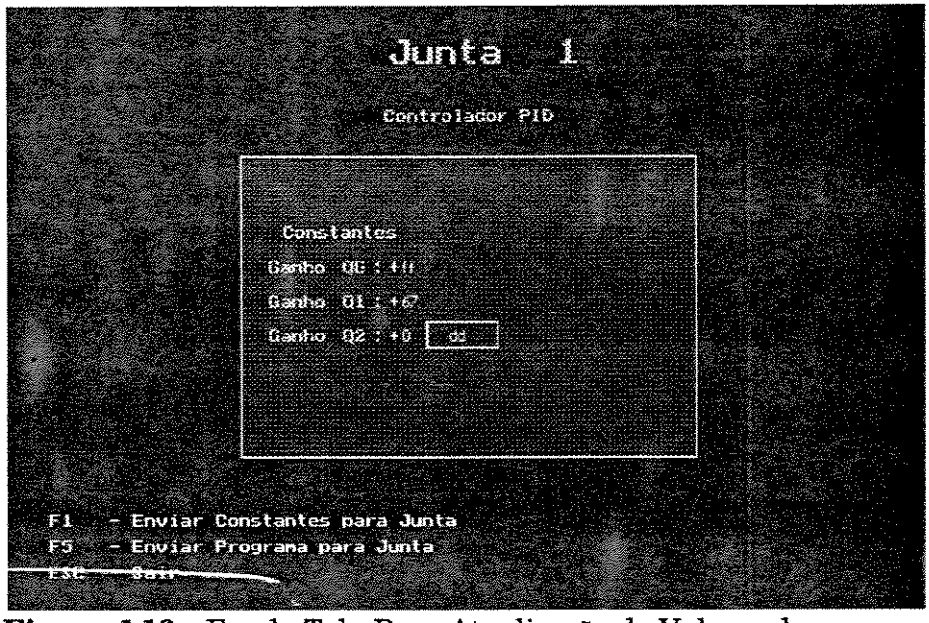


Figura 1.16 - Ex. de Tela Para Atualização de Valores dos Parâmetros dos Controladores de Juntas.

Ou ainda planejar um algoritmo de controle que ele próprio tenha planejado, respeitando o *hardware* do sistema, e armazenado previamente no disco rígido. Se ele optar por fazer um planejamento de algoritmo de controle de junta que tenha desenvolvido, aparecerá um pedido para que digite o nome deste arquivo que está em formato de programação hexadecimal em um subdiretório do disco rígido do Mestre ou em

um *diskette*. Se o operador selecionar o planejamento de tarefas, o sistema oferecerá possibilidades como trajetória, garra, e obstruções do ambiente sendo que dentro do ítem trajetórias terá a possibilidade de escolher se deseja ponto-à-ponto, ou contínua. Dentro do ítem ponto-à-ponto, poderá optar pelo tipo de interpolação, velocidade entre pontos, aceleração entre pontos, número de pontos, etc, que envolvem este tipo de procedimento. Se optar por trajetórias contínuas poderá, por exemplo, optar pela técnica do torque calculado, técnica *ATGS* por seccionamento em subtrajetórias ou busca heurística com inteligência artificial desenvolvida nesta tese, ou qualquer outra que venha a ser incorporada ao pacote de *software*.

Se optar por garra poderá planejar os movimentos que ela deverá realizar como por exemplo, determinar posições de abertura e fechamento, e forças de preensão. E se optar por obstruções do ambiente, poderá definir as restrições do volume de trabalho do robô.

O ítem Leitura de Arquivos é disponível para que o usuário possa recuperar facilmente todos os dados relevantes de um prévio planejamento que foi realizado, e que foram guardados em algum subdiretório do disco rígido do Mestre através de outro recurso oferecido pelo mesmo pacote de *software*. Uma vez selecionado este ítem, aparecerão na tela os nomes dos arquivos disponíveis que tem informações sobre planejamento de funcionamento do robô em um subdiretório padrão, que também pode ser modificado pelo usuário.

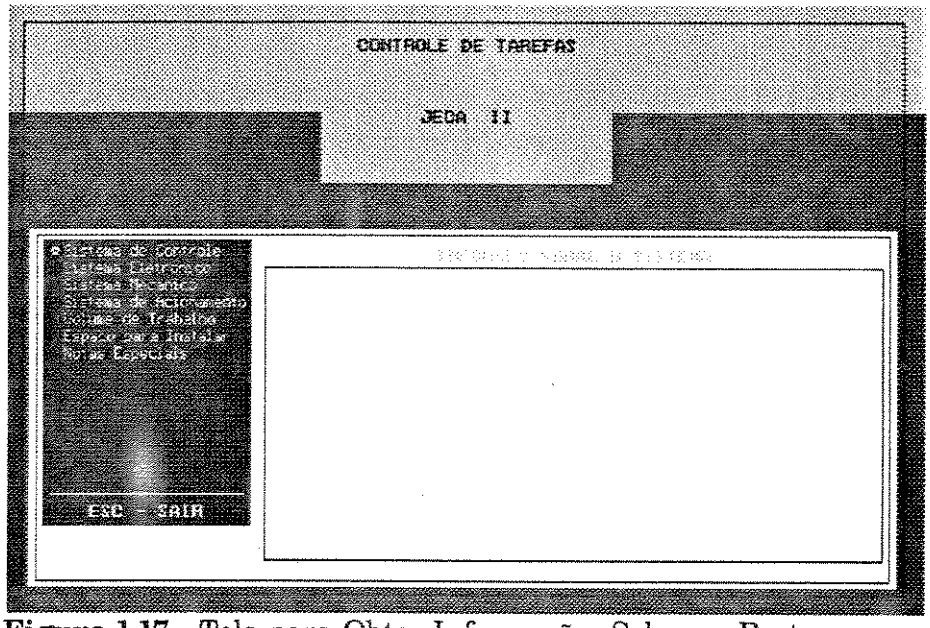


Figura 1.17 - Tela para Obter Informações Sobre as Partes que Compõem o Sistema de Controle do Robô **JECAII**.

No ítem Info JecaII, o usuário pode obter informações sobre as características das partes do sistema, para se acaso tenha alguma dúvida sobre ele (**figura 1.17**). O arquivo que contém estas informações pode ser facilmente alterado, para inserção e retirada de informações, para ser atualizado de acordo com as alterações que o projeto vier a sofrer. Estas informações são por exemplo: Sistema de controle, sistema eletrônico (ver **ANEXO A**), sistema mecânico, sistema de acionamento (Ver **ANEXO B**), volume de trabalho, parâmetros de *Denavit* e *Hartemberg* **[20]**, modelos, espaço para

instalar, e notas especiais.

As possibilidades neste tópico de planejamento são muitas, e pode-se abrir tantos sub-ítens com ramificações quantos se queira. No sub-ítem juntas, por exemplo, cada programação depende dos tipos de algoritmos disponíveis, e dos tipos e formas de entradas de dados que sejam necessários para torná-los corretos e aptos ao controle de cada junta em particular. Sempre que um novo algoritmo for posto para integrar o pacote, ele deve vir acompanhado da parte de SCHM para sua fácil alteração de parâmetros. A estratégia que adotamos para o armazenamento de dados sobre parâmetros variantes dos algoritmos é deixar uma área específica reservada da memória *RAM* de cada Escravo para injetar novos valores. Neste ponto o usuário programador de algoritmos de controle direto, deve prever quantas posições de memória são necessárias para isso, reservá-las, e fazer as alterações na programação de alto nível, para compatibilizar a entrada de dados via teclado, com a respectiva aparição de ítem na tela, e o respectivo envio de novo dado para a posição de memória adequada.

Existem subrotinas prontas que fazem parte do pacote para facilitar esta comunicação que podem ser utilizadas. Basta chamar primeiramente a parte de interpretação de caracteres do teclado, depois fazer a conversão via outra subrotina para o valor hexadecimal correspondente, informar para qual posição de memória o dado deve ser tranferido, e para quais Escravos este dado deve ser enviado.

Também existe a possibilidade de um programador realizar toda a programação independentemente da utilização deste pacote de *software*, apenas tomando conhecimento e respeitando as definições de *hardware* e endereços de acesso para a comunicação entre os setores que compõem o sistema. Mas esta não é nossa intenção, porque obrigaria a cada usuário ser um experto em eletrônica e programação de sistemas microprocessados. Nossa intenção é que o SCHM possa tornar a programação tão fácil quanto possível, e que o usuário não precise consultar manuais de operação toda vez que for programar. A indicação que fazemos é que, sempre que for inserido um novo algoritmo de controle, exista a participação de pelo menos um dos projetistas do sistema para eventuais esclarecimentos de dúvidas.

Citamos aqui que no momento apenas lançamos as idéias de como o pacote de *software* deve ser arquitetado, e quais as idéias básicas que deve conter. Esta programação de todo o pacote ainda está em andamento.

As saídas para o monitor de video controlado pelo Mestre e que compõem o sistema SCHM foram pensadas para apresentação em sub-ítens que compõem janelas apropriadas com o conjunto de informações para escolhas específicas que o sistema admite. No momento a criação dessas janelas está feita por programação especificamente para esta finalidade, com controle de telas gráficas em alta resolução padrão *EGA/VGA* e utilização de 16 cores. Mas esta idéia de apresentação com utilização de janelas já está sendo passada para a filosofia de apresentação e controle em ambiente *Windows*, que é uma tendência mundial para o gerenciamento de *sotwares* em máquinas tipo a que utilizamos como Micro Mestre. A passagem para esta filosofia permite também desenvolver uma versão mais avançada do pacote de programação incluíndo acesso direto a pacotes de programação e simulação (tecnologia *DDE*) desenvolvidos para operarem com sistema operacional *DOS* disponíveis no mercado especializado de *softwares*, que trarão maiores facilidades para desenvolver programas mais complexos de controle do sistema.

Quando o usuário já realizou o planejamento da tarefa que o robô deverá

executar, então ele pode passar para a etapa de simulação (**figura 1.18**), apenas utilizando as setas de controle do teclado. Quando se trata de escolha de ítem dentro de uma janela, existe uma indicação através de círculos pequenos que mostram a escolha atual para seleção, e sempre existe a possibilidade de acesso direto a um determinado ítem através do pressionamento de teclas apropriadas segundo indicações do SCHM.

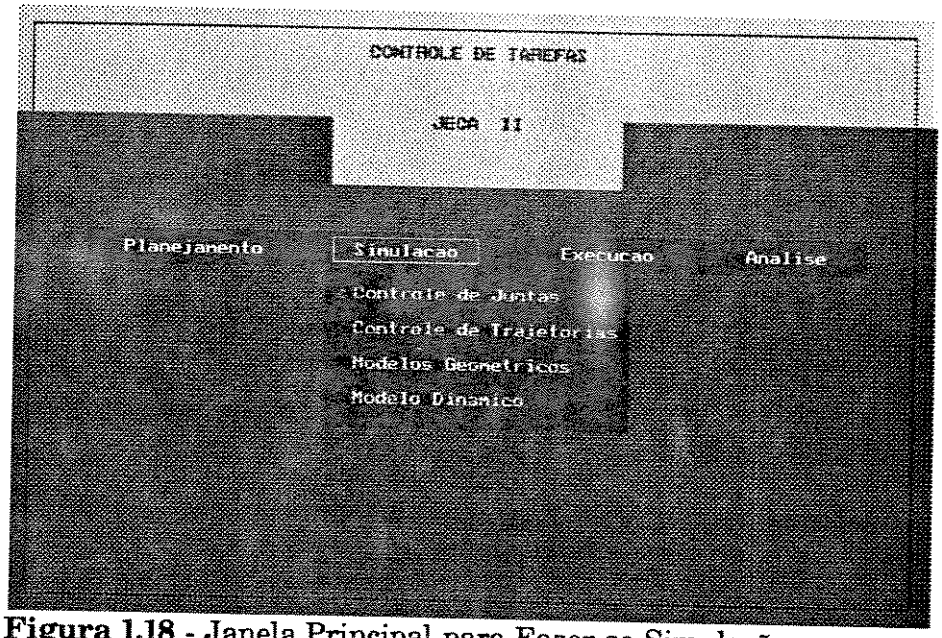


Figura 1.18 - Janela Principal para Fazer-se Simulações.

Para realizar uma simulação, o usuário tem atualmente quatro possibilidades que podem oferecer enorme auxílio para a verificação prévia, sem colocar o robô em funcionamento, se seu planejamento ao ser posto em prática funcionará de acordo com suas expectativas. Estes quatro tipos de simulações são: Controle de uma junta isolada, controle de trajetórias, solução do modelo cinemático, e solução do modelo dinâmico.

Para a simulação do controle de juntas, ele poderá selecionar qual a junta que deseja simular em função do planejamento feito na outra etapa, onde foi selecionado o algoritmo de controle, e indiretamente o movimento que a junta deverá fazer para compor os movimentos físicos necessários para executar a tarefa planejada. Neste caso a simulação é individual para cada junta, supondo que as demais juntas influem na resposta sob a forma de uma perturbação. O usuário poderá também ignorar o planejamento da tarefa feita e simular as respostas de cada junta à um outro conjunto de níveis de referências (**figura 1.19**). Isso serve para conferir se o seu desempenho individual está satisfatório ou deve ser alterado. Aqui pode-se escolher por uma programação padrão que o sistema oferecerá segundo otimização feita pelos projetistas do **JECAII**, ou então definir modelos que é um ítem preparado para o caso da estrutura modular do robô ser alterada, e necessitar-se alterar seus dados, como por exemplo, tipo, reduções, etc... Pode também redefinir-se seus limites de curso, selecionar-se controladores, e redefinir-se o tempo de amostragem utilizado pelo servomecanismo de controle responsável por cada uma. Se for constatado que algo deve ser alterado, após uma simulação através dos gráficos apresentados com auxílio do ítem Analisar, o usuário pode retornar ao respectivo sub-ítem da janela de planejamento e alterar os parâmetros que achar necessário, fazendo esta interação tantas vezes quantas forem necessárias para obter boas respostas. Está previsto

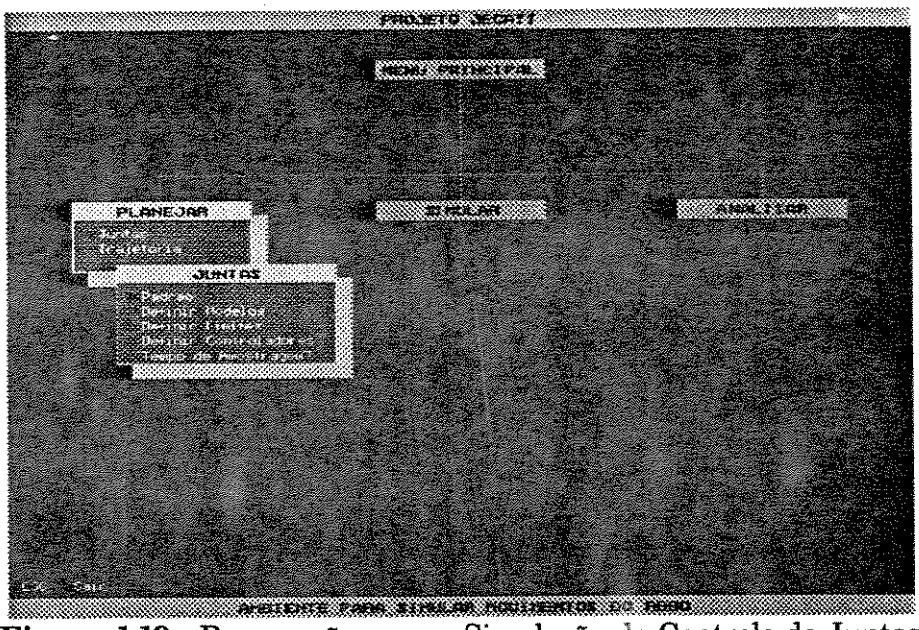


Figura 1.19 - Preparação para a Simulação do Controle de Juntas.

que estas simulações apresentem curvas de respostas temporais para variáveis como posição (**figura 1.20**), velocidade (**figura 1.21**), aceleração (**figura 1.22**), erros de trajetória, etc..., com controle por cursor sobre estas curvas para determinar os tempos e seus respectivos valores, objetivando obter informações precisas dos movimentos, e que possa-se adicionar perturbações nas variáveis para verificação do desempenho dos algoritmos de controle escolhidos. Estes resultados de simulações também poderão ser enviados para impressão como forma de ter-se dados para análises mais eficazes.

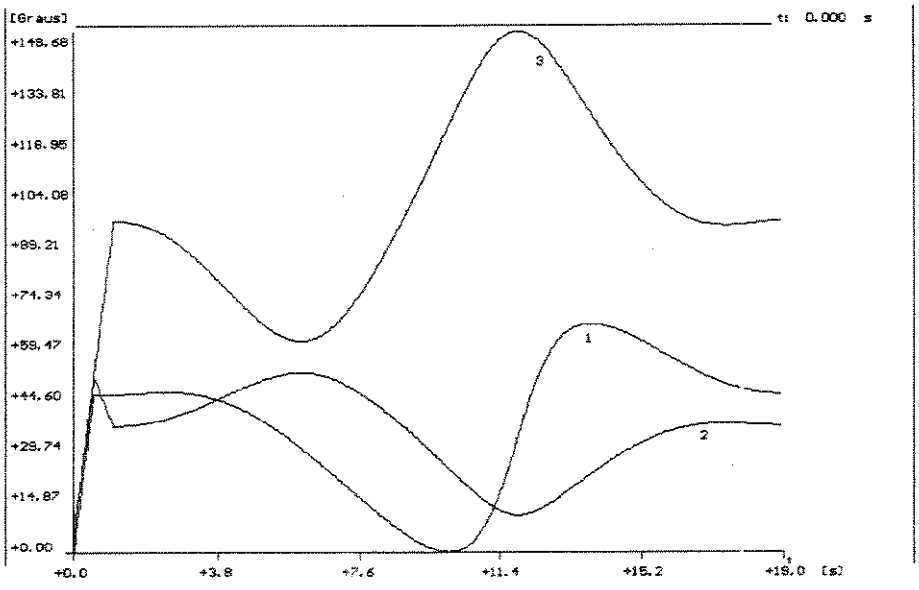


Figura 1.20 - Gráfico de Posições de Juntas.

No sub-ítem de simulação de controle de trajetórias, o usuário poderá verificar como deverão ser os movimentos globais do robô, quando todas as juntas são coordenadamente movidas de acordo com o respectivo planejamento feito na etapa anterior

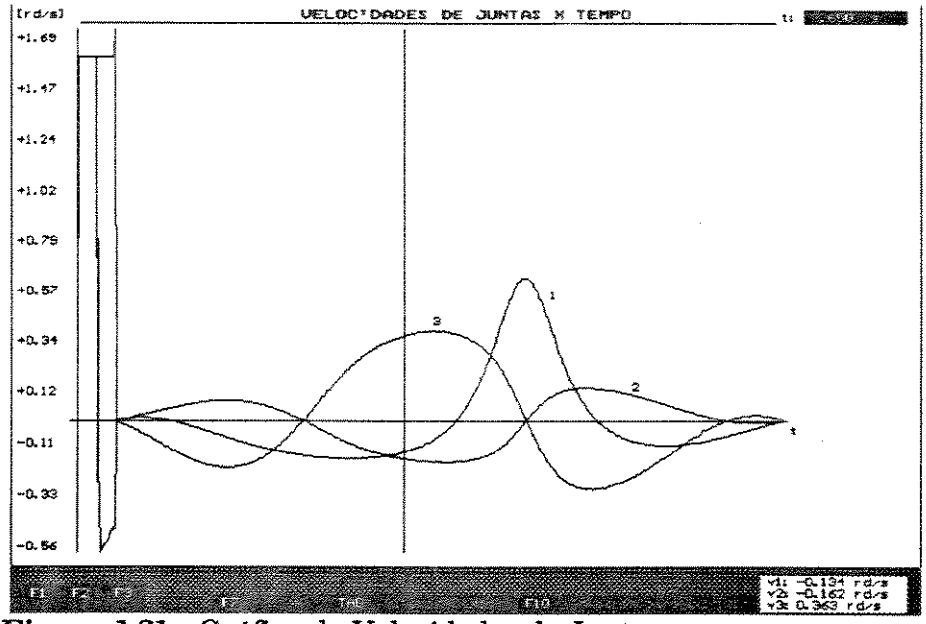


Figura 1.21 - Gráfico de Velocidades de Juntas.

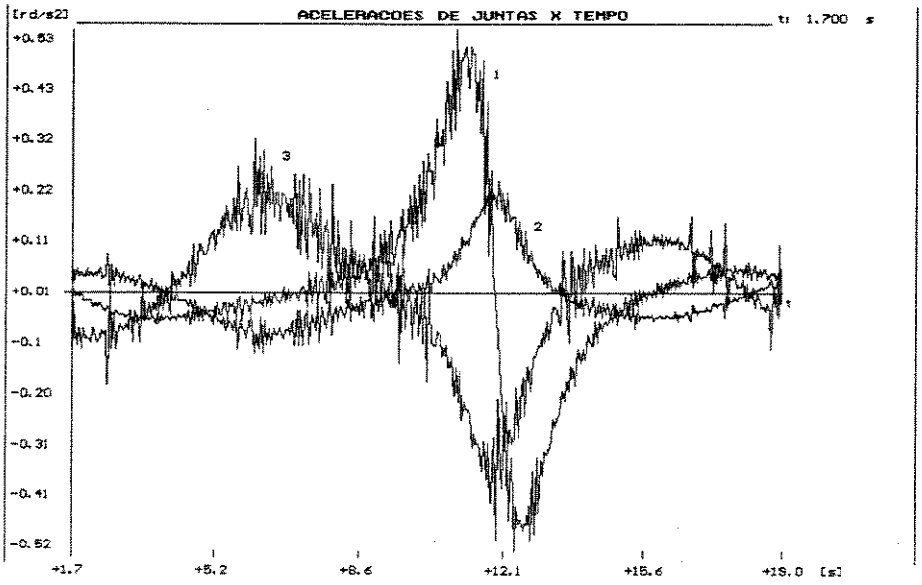


Figura 1.22 - Gráfico de Acelerações de Juntas.

(**figura 1.23**). Este procedimento é mais complexo que o anterior razão pela qual somente juntas individuais são consideradas, mas aqui ele poderá obter informações, por exemplo, sobre o comportamento das variáveis de posição, velocidade e aceleração do elemento terminal do robô, e de abertura e fechamento da garra adaptada a este elemento terminal. Aqui também está previsto que o usuário possa utilizar a geração de outra trajetória qualquer com suas respectivas equações paramétricas, para realizar testes rápidos de comportamento, e são considerados os controladores selecionados na janela de planejamento em cada simulação.

Para se planejar uma trajetória existem vários pontos que devem ser considerados. O primeiro deles é definir o espaço total que contém esta trajetória,

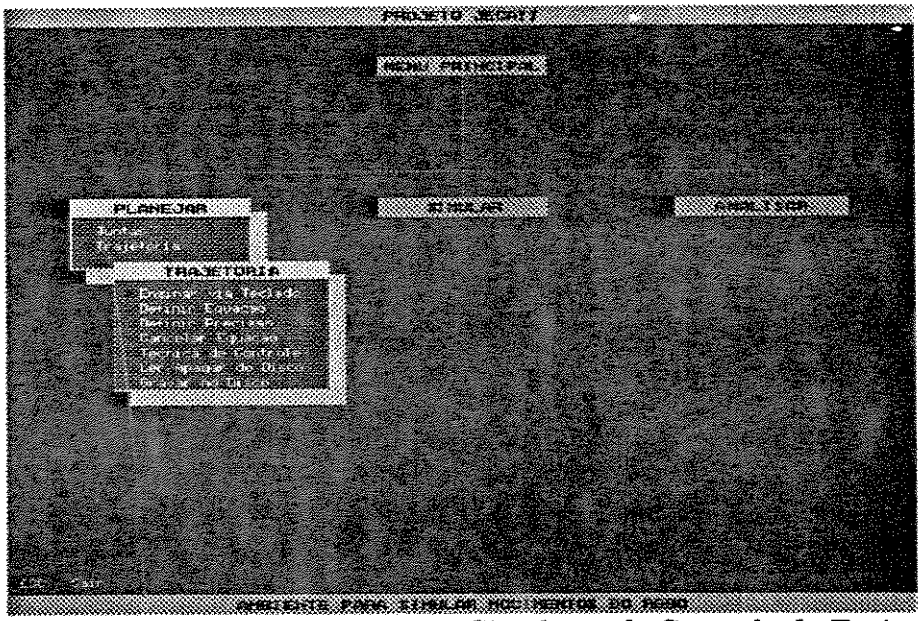


Figura 1.23 - Preparação para a Simulação do Controle de Trajetórias.

verificando a presença de singularidades da estrutura e obstáculos. Normalmente existe um espaço, ou volume de trabalho bem definido, que o robô pode atingir com seu elemento terminal. Então se a trajetória necessária para ser rastreada em função da tarefa a realizar, puder ser definida totalmente dentro desse volume de trabalho, o processo fica mais simples. Se a tarefa, por exemplo, for de usinagem em algum material, basta posicionar este material de modo adequado dentro dessa região. Entretanto, se ela não puder ser definida totalmente dentro do volume de trabalho, será necessário executar o trabalho por partes, subdividindo a trajetória total. Mas podemos imaginar que cada parte dessas compõe uma nova tarefa, portanto podemos abordar o problema do planejamento como uma parte da execução total.

Existem tarefas que exigem um procedimento contínuo de trabalho desde o início até a sua finalização, embora nem todas necessitem isso. Um processo de soldagem, por exemplo, pode exigir algo assim. Neste caso, o único limitante é o volume de trabalho do robô que deve ser adequado. Esse problema somente pode ser resolvido com o projeto mecatrônico de um robô apropriado para a realização de tais tarefas.

Nós trataremos dos casos que as tarefas exijam o rastreamento de trajetórias que podem estar contidas inteiramente dentro do volume de trabalho. E isso não invalida a extrapolação para outros robôs com volumes de trabalhos mais amplos.

Existem enormes quantidades de tarefas que apenas exigem movimentos em planos, ou mesmo em retas. Para estas, o problema de planejamento, geração e controle das trajetórias fica bastante mais simplificado. É evidente que tudo fica em função das equações matemáticas que descrevem a trajetória requisitada por esta tarefa. A priori um computador pode resolver qualquer equação matemática, desde que seja bem programado para fazer isso. O problema maior é que as equações matemáticas mais complexas exigem tempos maiores de processamento para encontrar suas soluções, e isso pode causar dificuldades para que o sistema de controle consiga gerá-las em tempo hábil para que o processo possa ser bem controlado. Por este motivo existem muitos sistemas robóticos que utilizam a técnica de ensinar o robô a rastrear trajetórias. Esse processo normalmente não é muito preciso, mas pode ser satisfatório para resolver uma grande

variedade de problemas. Ao invés de programar as equações que descrevem a trajetória, é dada ao usuário planejador a possibilidade de mover lentamente todas as juntas do robô através de todo o seu volume de trabalho, e assim fazer com que o elemento terminal do robô vá lentamente sendo conduzido sobre o caminho que definirá a trajetória. Juntamente com esse processo, o sistema, por si só, ou por comandos, armazena os dados lidos dos sensores durante os movimentos, em vetores adequados. Então quando é encerrada a etapa de ensino, o sistema já possui definida a trajetória a ser rastreada como um conjunto de pontos, não necessitando resolver equações para gerá-los durante a etapa de funcionamento do robô.

Apesar de poder funcionar muito bem, este método exige que o sistema todo do robô seja muito robusto, com relação a suportar os esforços que as tarefas exigem. Como não existe uma formulação matemática para definir as trajetórias, faltam recursos para o sistema de controle agir de forma compensatória caso hajam desvios, ou acontecimentos imprevistos que possam provocar esforços inconvenientes sobre a estrutura. Outro problema é que dependendo da precisão que se queira para o rastreamento da trajetória, com uma técnica assim é necessário que o sistema de controle eletrônico tenha grande quantidade de memória disponível. Também existe o problema de ensinar a velocidade e a aceleração com as quais os movimentos devem ser realizados. Normalmente os robôs que utilizam estes métodos, já tem definidas previamente as velocidades que farão os movimentos, e somente são armazenados dados sobre as posições das juntas. As acelerações são determinadas em função destas posições a atingir, e das velocidades selecionadas dentro do rol de velocidades possíveis.

Existe uma explicação simples sobre o porque da utilização tão difundida deste método de planejamento de trajetórias; ela está no fato de que o usuário planejador não necessita ter conhecimento de técnicas matemáticas e de controle. Ele apenas deve ter o conhecimento de como proceder de forma repetitiva, e isso para treinamento pessoal é muito mais simples. No entanto, existem tarefas que necessitam muita precisão e flexibilidade, e quer-se que o custo total do planejamento, e do sistema de controle sejam os mais reduzidos possíveis. Portanto, se ele for dotado de técnicas de programação e controle que auxiliem a um programador inserir equações matemáticas de forma fácil, o tempo de planejamento pode ser enormemente reduzido, contribuindo para reduzir muito o seu custo. Tome como exemplo que o robô deve ser programado para rastrear uma elipse contida em um plano, e que o seu perímetro total seja de 1metro. Se for utilizado o método de ensino por movimentação do robô, primeiramente o usuário deverá fazer um desenho desta elipse em uma superfície plana, depois marcar todos os pontos de interesse pelos quais ele está interessado que o robô passe, então posicionar este plano com o desenho de forma precisa dentro do volume de trabalho do robô, e ir fazendo os movimentos, de posição para posição, levando o elemento terminal do robô sobre cada uma delas, e fazendo a respectiva gravação. Pode-se notar que isso é extremamente tedioso, e exige um tempo enorme para a realização, mesmo que o usuário planejador seja muito hábil no seu manuseio, além do fato de ser um método artesanal, incompatível com os recursos tecnológicos disponíveis em pacotes de resolução de modelos e equações. Enquanto que este planejamento poderia ser muito facilitado com a introdução apenas de uma equação matemática muito simples, e de um passo de amostragem. A única exigência é que o usuário programador tenha um conhecimento suficiente de matemática. E normalmente em ambientes onde hoje pode-se utilizar a tecnologia da robótica, existem muitas pessoas com este tipo de conhecimento.

Uma forma de programar equações matemáticas para encontrar suas soluções é escolher uma linguagem de programação, e transferir toda a estrutura da

equação para a programação nestes códigos da linguagem, utilizando-se do conhecimento algébrico para resolvê-las, ou algum pacote de *software* existente para esta finalidade. Só que neste caso, toda vez que se necessite resolver uma nova equação, todo o processo deve ser repetido. Isso é oneroso também, se pensarmos em termos de programação com uma linguagem que deve ser compilada e *linkada* para execução em uma máquina com as características do Micro Mestre. O ideal seria que o engenheiro planejador pudesse escrever uma equação matemática diretamente no teclado do Mestre, assim como a escreveria em uma folha de papel, com a sintaxe usual e universal que se utiliza em desenvolvimentos matemáticos, e que a programação dada a ele interpretasse esta sintaxe, ou ainda via a utilização de pacotes de *software* disponíveis para estas finalidades através de gerenciamento *Windows*, por exemplo: MATLAB, MATEMATICA, etc.

Com esta intenção foi criado um programa interpretador para entrada de equações diretamente por digitação de dados, com uma sintaxe apropriada, no Micro Mestre, com analisador sintático automático, que permite a edição de linhas como até 60 caracteres que podem possuir os operadores matemáticos básicos de adição, diminuição, multiplicação e divisão, operadores trigonométricos como seno, cosseno, tangente, etc..., operadores de exponenciação e logarítmicos, e permite inserção de até 10 níveis de parentesis entre os operadores. Os números que integram estas linhas devem ser escritos em notação de números reais, e existe a possibilidade da inserção de uma variável (k) para ser operada juntamente com cada linha, que pode ser utilizada como parâmetro variante para produzir a discretização temporal da equação (**figura 1.24**).

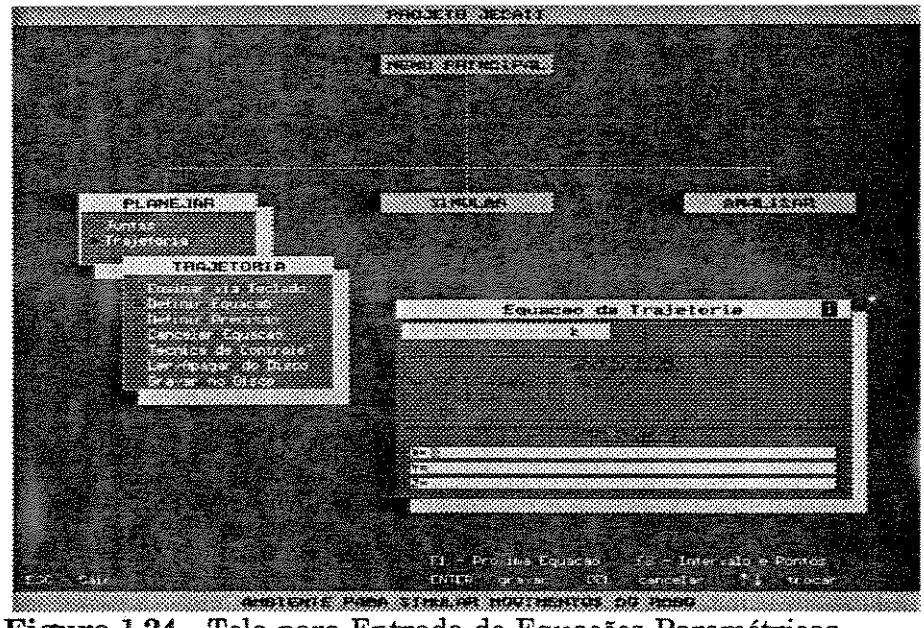


Figura 1.24 - Tela para Entrada de Equações Paramétricas.

Este interpretador oferece a possibilidade de poder-se definir quais os limites de variação para a variável "k", e qual o passo de discretização a utilizar para gerar as respostas da equação (**figura 1.25**). Ou então, qual o número de pontos requisitados entre os limites de variação para "k" para o cálculo automático de qual deve ser o passo de "k" (**figura 1.26**). E para a inserção de mais outros operadores matemáticos que possam interessar, este programa interpretador foi desenvolvido de modo modular, através de funções, que está aberto para a incorporação de mais operadores.

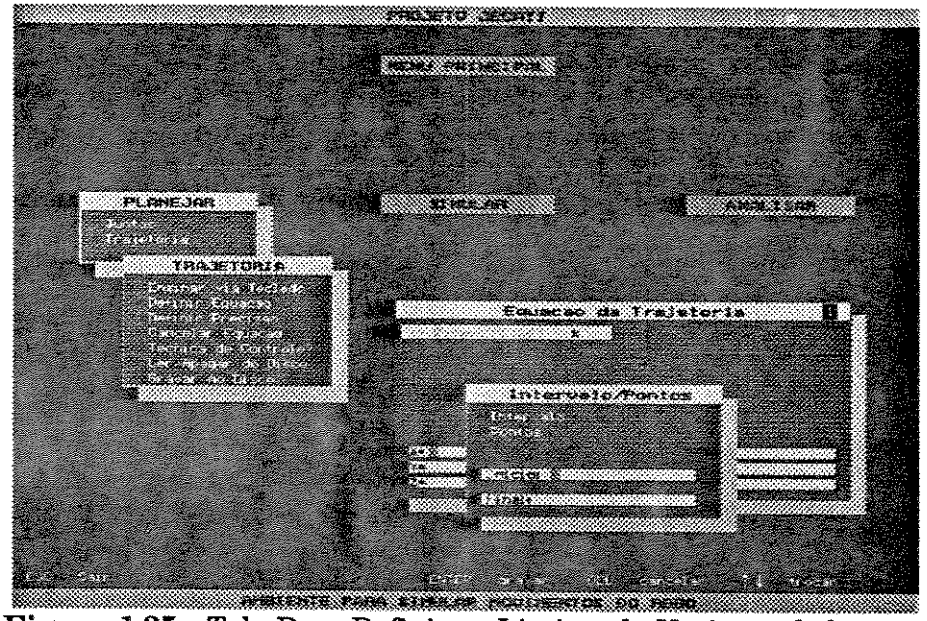


Figura 1.25 - Tela Para Definir os Limites de Variação de k.

Com este mesmo interpretador de equações matemáticas entradas de teclado, anexamos a possibilidade de entrada de três equações com o mesmo parâmetro de variação "*k*" (equações paramétricas), para poder-se definir equações espaciais através de suas equações paramétricas. Isso permite uma excelente flexibilidade para poder-se gerar as trajetórias que o robô deve rastrear. Ainda está previsto que a trajetória possa ser montada por partes, isto é, pode-se definir vários conjuntos de equações paramétricas que definem uma trajetória global. Então o operador planejador pode construir, por exemplo, um triângulo, definindo os seus lados pelas respectivas equações e intervalos de variação do parâmetro "*k*", e informando sobre a seqüência que estas equações devem ser resolvidas.

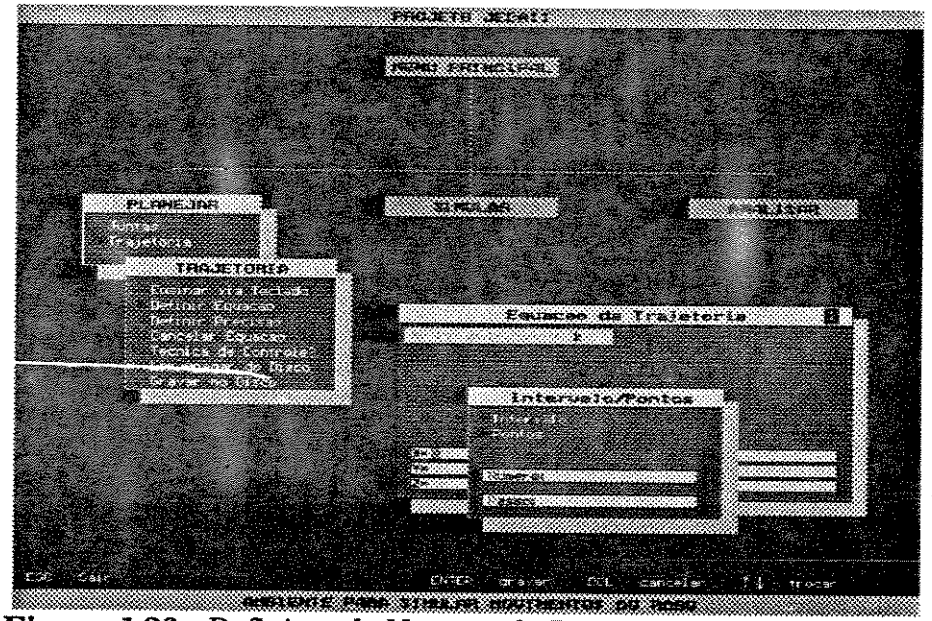


Figura 1.26 - Definição do Número de Pontos Que a Trajetória Conterá, ou o Passo de Discretização.

Quando estas equações forem devidamente digitadas, o usuário poderá dar partida no processo de geração dos pontos que formam a trajetória definida por cada equação; cada ponto é definido por uma trinca (X,Y,Z), fazendo com que o programa gere-os, um a um, de acordo com a variação de "*k*" programada.

No pacote de *software* de supervisão e controle do sistema do robô **JECAII**, este interpretador pode ser utilizado tanto para o planejamento como para a simulação e a execução do rastreamento de uma trajetória. Apenas que para a execução do rastreamento, ele interage com outras funções dadas pela programação, como é explicado mais adiante na apresentação dos ítens que compõem esta respectiva janela de opções oferecidas pelo menu principal do SCHM.

A simulação do controle de rastreamento de uma trajetória, é dependente dos tipos de algoritmos escolhidos, tanto para o nível de supervisão como para o nível de juntas, porisso esta parte do pacote de *software* necessita utilizar os dados programados através da janela de planejamento, e a programação gerencia diversas funções que fazem a representação física e matemática do robô. Esta parte tem maior complexidade de programação do que a parte de execução com a movimentação física do robô, porque todos os dados que seriam obtidos diretamente através dos sensores postos para fazer medidas sobre as variáveis de interesse que evoluem de acordo com seus movimentos, são gerados através do processamento das equações matemáticas que o representam; porisso que este processo normalmente é mais lento, e pode-se tornar ainda mais lento de acordo com os dados que necessitam ser gerados para a apresentação gráfica, em função das exigências da pessoa que esteja realizando as simulações.

Nós criamos a possibilidade de apresentação gráfica em dois modos. O primeiro deles, e mais lento, procede a apresentação da evolução do elemento terminal, de acordo com cada ponto atingido, com controle em alta resolução, com apresentação simulada em perspectiva *3D*, e amostrando saídas numéricas coordenadas com a evolução das variáveis X, Y, e Z que definem o posicionamento do elemento terminal, e dos respectivos valores numéricos da evolução do posicionamento de cada junta. Neste primeiro modo, também são apresentados dados sobre os tempos de simulação e de execução decorridos, supondo que os movimentos fossem reais em função dos períodos de amostragem escolhidos, e programados para as malhas de controle de supervisão das trajetórias (**figura 1.27**).

Estes períodos de amostragem também podem ser programados através da entrada em uma janela oferecida por esta parte de simulação. E os valores adequados para eles são função da capacidade de processamento computacional do sistema eletrônico do robô, e da complexidade do algoritmo escolhido (**figura 1.28**).

Para que se possa realizar os experimentos de simulação levando-se em consideração as influências das forças externas que podem causar desvios de rastreamento, existe a possibilidade de se programar alguns de seus possíveis efeitos, através do acesso às janelas de Perturbações. Na janela principal deste ítem, pode-se selecionar entre três ítens chamados respectivamente por: Perturbações Na Medida das Distâncias, Perturbações Nas Malhas de Posições, e Perturbações Nas Malhas de Velocidades (**figura 1.29**).

As perturbações nas medidas das distâncias referem-se principalmente aos erros causados devido aos instrumentos de medidas, e aos erros de modelagem do robô e

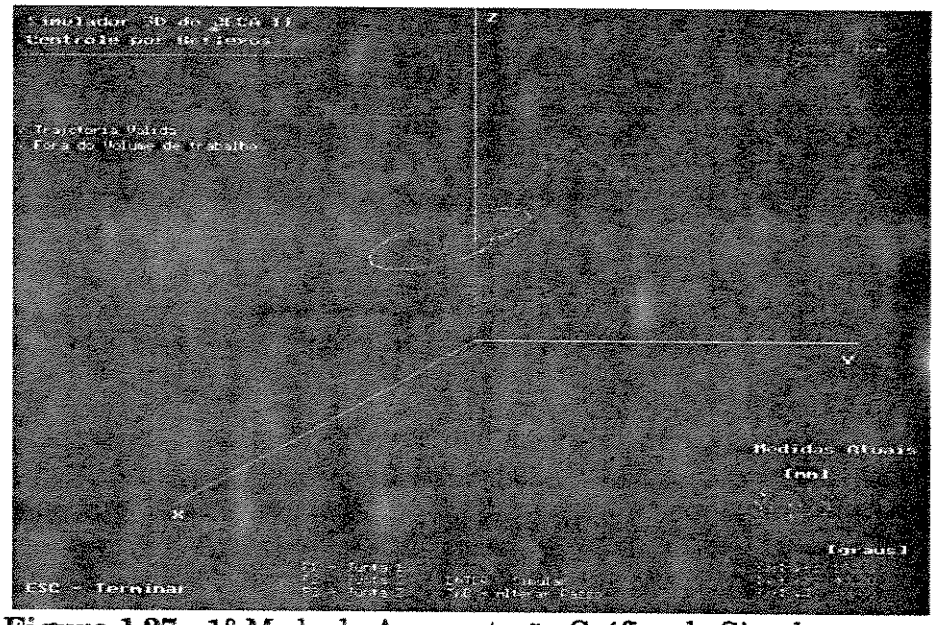


Figura 1.27 - 1º Modo de Apresentação Gráfica de Simulação.

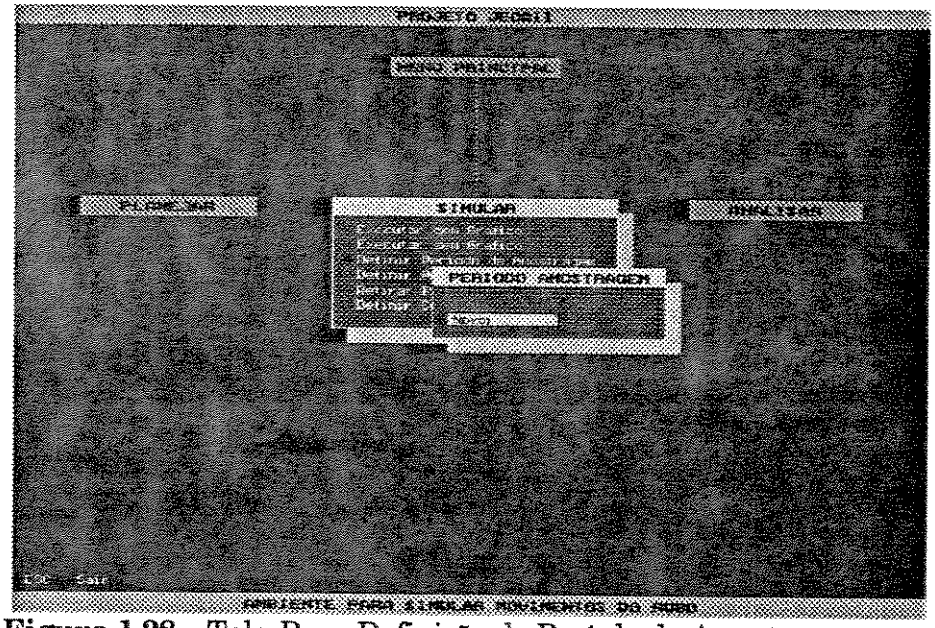


Figura 1.28 - Tela Para Definição do Período de Amostragem da Malha de Controle de Trajetórias.

de seu ambiente de trabalho. Estas medidas de distâncias são relativas ao ponto onde encontra-se posicionado o elemento terminal do robô, e o próximo ponto objetivo requisitado pela trajetória a rastrear. Ela pode ser obtida, por exemplo, através de um sistema de visão eletrônica, com auxílio de câmeras de vídeo, que por sua vez, podem fornecer dados imprecisos devido a problemas de calibração, escalamento, e influências de erros do sistema de digitalização e aquisição de dados. Ou ainda, esta medida pode ser calculada indiretamente através da utilização da leitura dos sensores de posição das juntas do robô, e do seu modelo cinemático direto. E as perturbações nas malhas de posições e de velocidades são relativas a erros ocorridos nos servomecanismos, devido às influências da dinâmica dos movimentos, e dos erros de modelagem, que podem causar maus desempenhos por parte dos seus controladores.

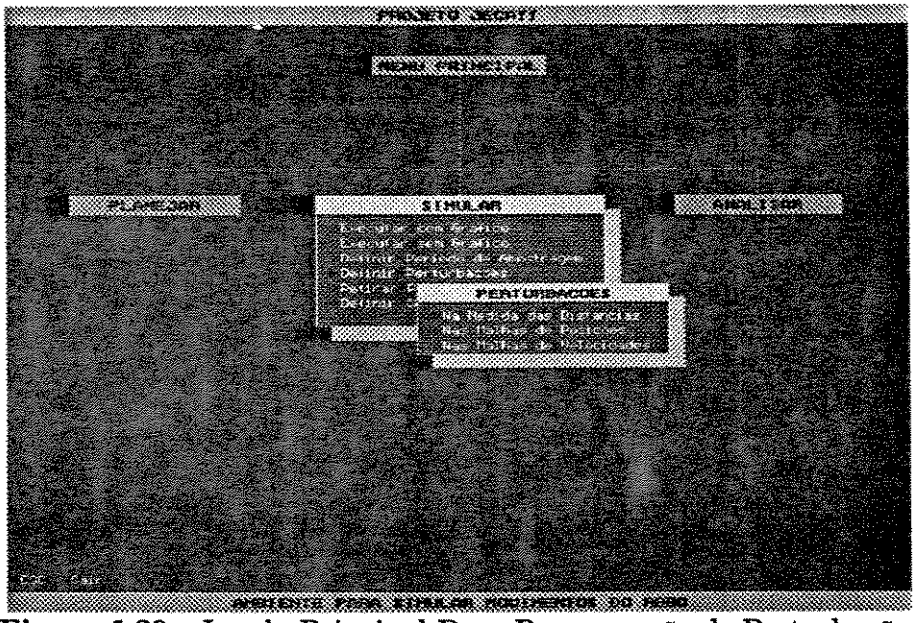


Figura 1.29 - Janela Principal Para Programação de Perturbações.

Uma vez selecionado um desses ítens, o SCHM permite que seja escolhido seu tipo. Atualmente estão a disposição três tipos, ou seja, Perturbações Randômicas, Perturbações Constantes, e Perturbaçôes Periódicas. Nesta janela também é possível fazer-se a verificação de qual é a programação atual das perturbações para simples verificação (**figura 1.30**).

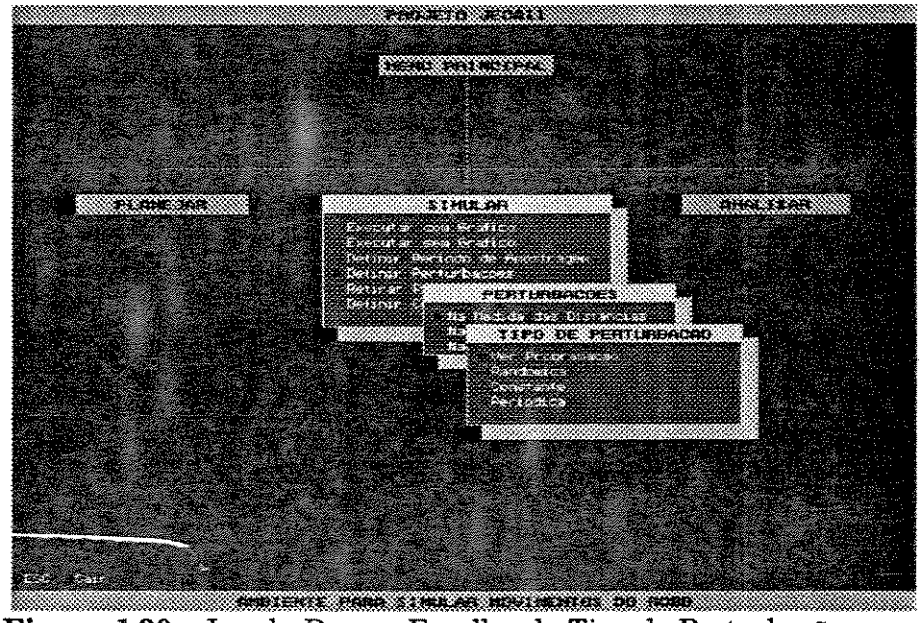


Figura 1.30 - Janela Para a Escolha do Tipo de Perturbação.

Quando é programada alguma perturbação nas malhas de posições ou velocidades, a programação atual do pacote impõe o mesmo tipo para todas as juntas. Isso pode não ser totalmente realista, mas pode dar uma boa aproximação, de qualquer forma, é possível especificá-las com maior detalhamento apenas inserindo mais janelas no SCHM. Cada escolha dessas provoca o direcionamento da programação para a entrada dos dados

respectivos a ela como pode ser visto na **figura 1.31**.

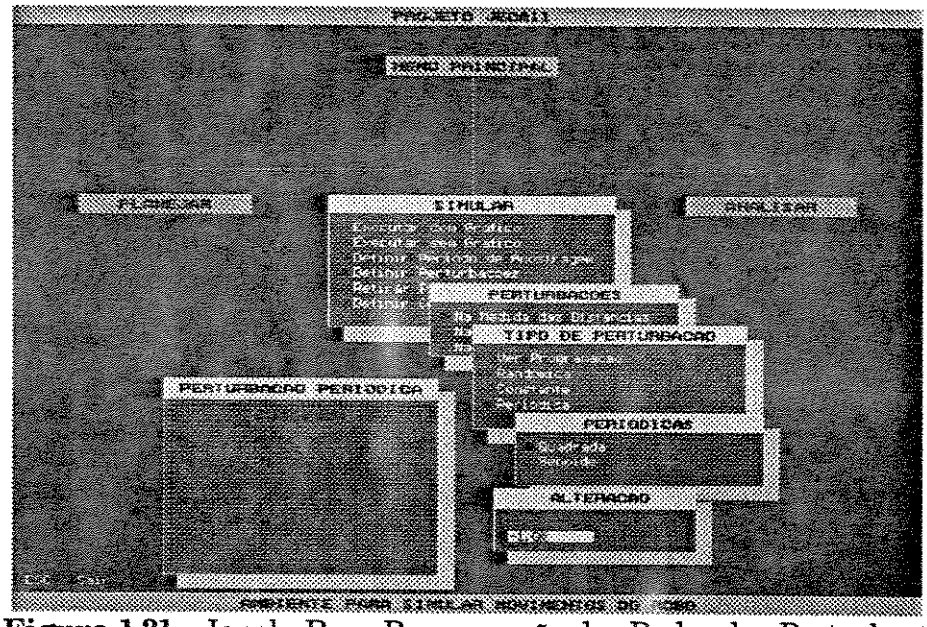


Figura 1.31 - Janela Para Programação dos Dados das Perturbações.

Assim, o programador pode selecionar que tipo de perturbação quer considerar nas simulações, qual sua forma, e o intervalo de duração, que não necessariamente deve começar no início da simulação.

Para que possa-se alterar as perturbações impostas, o SCHM também oferece uma possibilidade, bastando que o programador acesse o ítem apropriado dentro da janela principal de Perturbações.

No segundo modo de simulação do controle de trajetórias, não é apresentada a evolução gráfica apresentada no primeiro modo, e sim, apenas uma indicação que a simulação está evoluindo, e uma indicação de finalização da simulação (**figura 1.32**). Como as funções específicas para o controle da tela gráfica, neste caso, não são utilizadas, existe um ganho enorme de tempo. Os dados de interesse sobre as ocorrências da simulação, no entanto, são armazenados em arquivos especiais no disco rígido do Micro Mestre para fazer-se análises através da seleção da respectiva janela do menu principal oferecida pelo SCHM.

Como estes arquivos que armazenam dados das simulações são padrões e específicos para armazenar dados referentes à última simulação realizada, sempre que o usuário que está realizando as simulações desejar guardar estes dados para análises futuras, ou fazer comparações, ele pode gravá-los de forma automática e organizada no disco rígido do Mestre apenas entrando na janela de execução oferecida pelo menu principal do SCHM, e selecionando o respectivo ítem, dando um nome para este conjunto de dados, na intenção de recuperá-los rapidamente por ocasião de uma necessidade futura. Como são gerados muitos dados que dependem principalmente do tempo de simulação necessário para simular uma trajetória, também é necessária uma quantidade relativamente grande de memória para sua armazenagem. Nós não limitamos o tamanho que estes arquivos podem ter, mas lembramos que deve-se ter sempre em consideração este fato para que não ocorram problemas de falta de memória durante o processamento

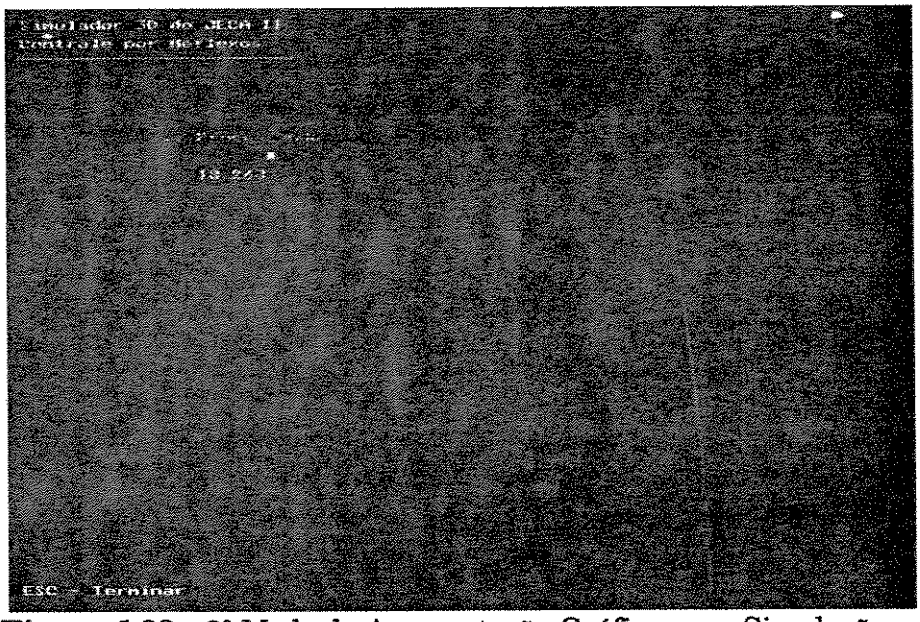


Figura 1.32 - 2º Modo de Apresentação Gráfica para Simulação.

da simulação que poderiam fazer o sistema ter seu processamento travado.

A gravação dos dados nesses arquivos é feita com o controle do próprio pacote de *software*; ao iniciar uma simulação, os arquivos padrão são abertos para escrita, e este processo de escrita é seqüêncial de acordo com a evolução da simulação. Quando a simulação acaba, é inserido um caracter de controle que indica o final de cada tabela de dados gerada. E este caracter serve para ser usado como sinalizador pela parte do pacote de *software* responsável pelo auxílio às análises. Esta não limitação do tamanho dos arquivos que armazenam os dados das simulações gera a necessidade da criação de funções especiais para sua interpretação; estas são expostas na apresentação do funcionamento da facilidade oferecida pelo pacote para as análises de resultados.

Ainda com respeito ao primeiro modo de simulação, onde é apresentada a evolução dos movimentos do robô, através do controle gráfico da tela do monitor, o pacote permite que a simulação destes movimentos seja feita passo-à-passo, para que o usuário possa acompanhar a evolução numérica das variáveis apresentadas na tela. Isso serve para acelerar o processo de ação/correção na programação, porque em caso de um movimento estar sendo conduzido mal, o usuário pode interromper imediatamente a simulação em qualquer ponto, e retornar a outras janelas de apresentação do SCHM para fazer alterações de acordo com o que foi constatado, e com seus critérios para resolver os problemas.

Dentro da janela de simulação, o usuário pode recorrer ao ítem Modelos Geométricos (cinemáticos), que oferece a possibilidade de fazer-se a simulação utilizando a matemática que define a geometria do **JECAII**. Fazendo a escolha, o SCHM apresenta uma tela própria para a simulação, dividida em duas partes, que funcionam sincronizadas (**figura 1.33**). Uma que permite a entrada de dados referentes aos ângulos das juntas do robô, e com utilização da cinemática direta, calcula as posições e orientações cartesianas do seu elemento terminal, apresentando-as na outra parte da tela. E vice-versa, o usuário poderá optar pelas entradas de dados que definem as posições e orientações cartesianas

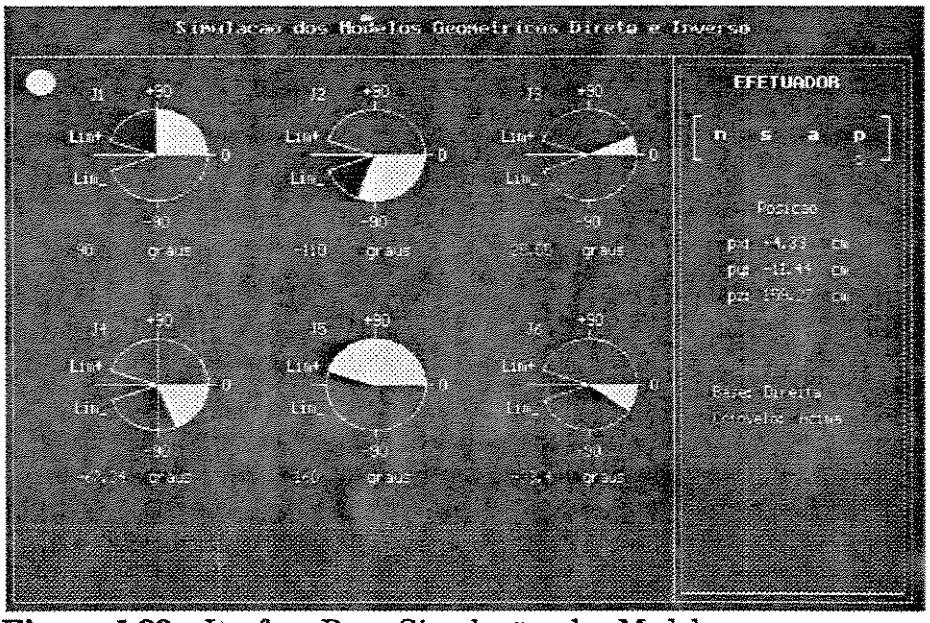


Figura 1.33 - Iterface Para Simulações dos Modelos Cinemáticos - Modo Direto.

e com utilização da geometria inversa do **JECAII** programada, são apresentadas na respectiva parte da tela, as posições de juntas respectivas. Como a geometria do **JECAII** permite atingir uma determinada posição espacial através de quatro configurações distintas, nesta última opção, existe a possibilidade de programação de duas variáveis auxiliares chamadas por Base e Cotovelo, que fazem esta seleção para a configuração desejada. A seleção entre a entrada de dados para a geometria direta ou inversa é feita através de uma tecla de controle, e a cada toque é feita a troca de opção intermitente (a parte da tela ativa é marcada com um círculo vermelho). As entradas de dados numéricas são sempre interpretadas como números reais (**figura 1.34**).

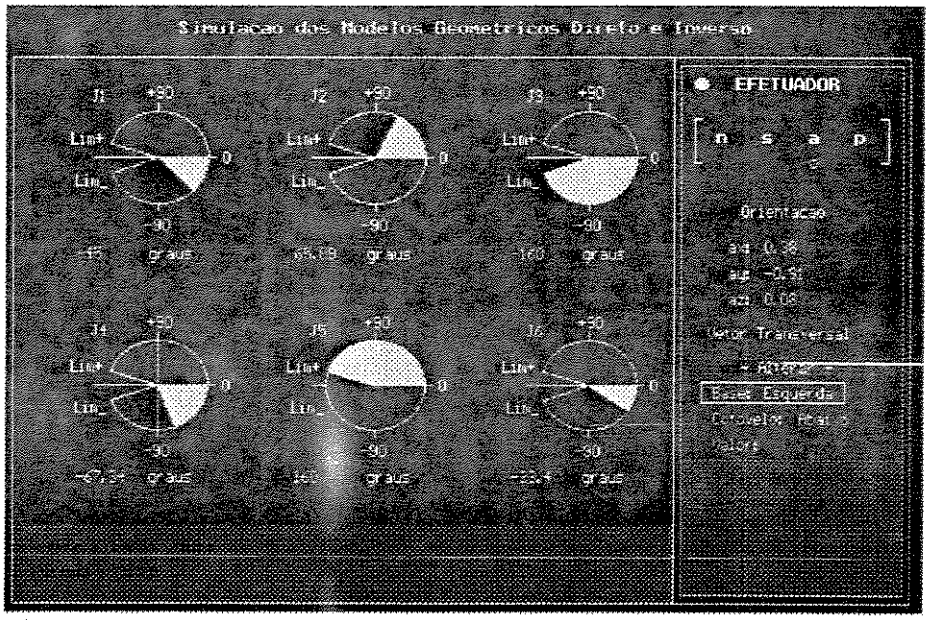


Figura 1.34 - Interface Para a Simulação dos Modelos Cinemáticos - Modo Inverso.

Na escolha pela cinemática direta, a alteração do ângulo de cada junta para nova simulação, é feita através da sua respectiva seleção e da entrada de um novo número equivalente ao ângulo de posição desejado para a junta em questão. Este método de entrada de dados permite edição. Quando a entrada está correta, basta o pressionamento de uma tecla de controle, e automaticamente o valor é inserido e amostrado na tela gráfica, e numericamente na parte da tela reservada para isso. Quando todas as alterações individuais de cada junta tenham sido realizadas, então o usuário deve fazer uma confirmação para que o simulador calcule e apresente as devidas saídas na outra parte da tela que indica as posições e orientações do elemento terminal equivalentes. As saídas que indicam as orientações do elemento terminal são normalizadas, e são apresentadas através de vetores, segundo a convenção internacional utilizada em robótica: **a**, **s**, e **n**, onde:

a (approach) : Vetor de ataque da garra, aponta na direção de seus dedos.
s (sliding) : Vetor transversal à direção de ataque da garra.
n (normal) : Vetor normal ao plano formado por **a** e **s**.

Nestes modelos cinemáticos direto e inverso programados, estão anexadas todas as limitações físicas apresentadas pelo **JECAII**. E existem saídas de informações diretas para a tela do monitor que indicam quando as entradas de dados programados pelo usuário não podem ser satisfeitas, ou seja aqueles casos onde são feitas tentativas que resultam em pontos do elemento terminal fora do volume de trabalho do robô.

Para retornar ao menu principal do SCHM, ou mais especificamente para a janela de simulação, basta que o usuário pressione uma tecla de controle. Para que ele sempre tenha as informações sobre quais as teclas ativas em cada tela, aptas a operação, estas telas gráficas apresentam estas respectivas informações em posições estratégicas que podem ser facilmente visualizadas.

Para a simulação do modelo dinâmico do **JECAII** também reservamos um ítem dentro desta janela de simulação, que tem a finalidade de realizar o cálculo deste seu respectivo modelo matemático. A intenção aqui é que o usuário possa obter dados sobre as forças e torques que sua estrutura sofre em função de suas características e das cargas impostas em seu elemento terminal. Esta parte da programação ainda não foi concluída, devido a que o levantamento completo de todos os dados referentes aos parâmetros estruturais do robô dependem do término completo de sua construção, porém a apção atual que fizemos é através do desenvolvimento matemático através da modelagem de *Lagrange*, fazendo o balanceamento das energias cinéticas e potenciais que aparecem durante os movimentos sobre a estrutura articulada do robô. Podemos assim obter os torques necessários em cada junta para suportar uma determinada situação de posição, velocidade e aceleração no elemento terminal do robô. A saída gráfica para esta simulação está pensada para seguir o mesmo padrão adotado para a simulação dos modelos cinemáticos. Porém, com outras facilidades como poder-se programar perturbações (não linearidades nas transmissões dos movimentos das juntas, efeitos produzidos pelos métodos e freqüências de acionamento, e a identificação ponderada dos efeitos das distintas forças que a estrutura sofre, como o efeito das forças gravitacionais, centrípetas e de *Coriolis*.

Como, neste caso, é necessário fazer simulações utilizando dados de como se o robô estivesse em movimento, estas simulações somente fazem sentido utilizando-se os dados planejados tanto para a trajetória como para os controladores, porisso o funcionamento desta parte da simulação é função do planejamento realizado

anteriormente. E durante o processo de simulação também devem ser gerados arquivos que componham tabelas de dados para posterior análise, com apresentações gráficas da evolução das variáveis envolvidas e das suas respectivas evoluções numéricas, através do controle pela janela de análise do menu principal do SCHM.

As ações tomadas pelo usuário nas janelas principais e derivadas de planejamento e simulação não executam nenhuma ação de movimento do robô, conforme apresentação feita anteriormente. Para fazer o **JECAII** agir, ou mover-se, sempre é necessário acessar a janela de Execução do SCHM (**figura 1.35**). Compusemos esta janela de nove ítems, cada um contendo procedimentos fundamentais para que o sistema ofereça ao usuário a possibilidade de poder ativar o funcionamento físico do robô, e dar e obter algumas informações essenciais sobre o funcionamento da sua eletrônica.

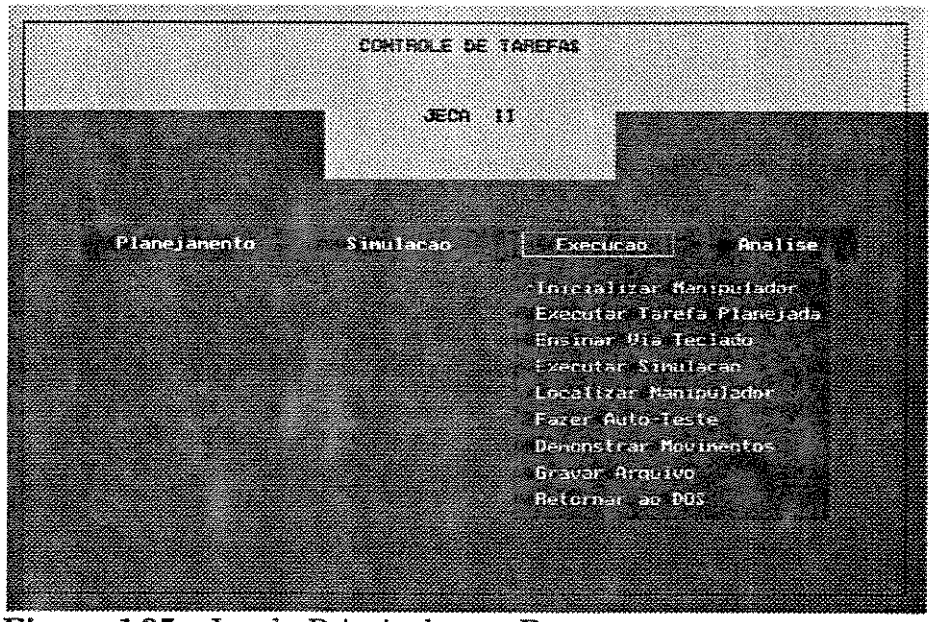


Figura 1.35 - Janela Principal para Execuções.

No ítem Inicializar Manipulador, o usuário pode ativar o programa que prepara o robô para começar a realização de uma tarefa. Este procedimento é também executado automaticamente sempre que o sistema é energizado. Primeiramente o Mestre bloqueia o processamento de todos os Escravos através de um pedido em modo *broadcasting* de seus barramentos, e garantindo que cada circuito de acionamento de junta receba a palavra digital que faz o seu respectivo motor ficar parado (parado, neste momento, não significa desenergizado), e então envia uma palavra digital de comando para cada Escravo processar uma rotina de inicialização que está programada em suas memórias *EPROM*, e libera imediatamente seus barramentos passando a verificar periodicamente uma posição de suas memórias *RAM* reservada como bandeira para sinalizar a finalização deste procedimento. Denominamos de bandeiras aquelas posições de memória *RAM* dos Escravos destinadas a conter valores lógicos que servem como indicadores de estado. Neste caso da execução da rotina de inicialização, o próprio Mestre durante o pedido de barramento inicial, inicializa a bandeira de cada Escravo, e estas bandeiras assumem apenas dois valores lógicos (equivalente a um Sim e um Não, ou em valores hexadecimais FFh e 00h), um Não equivale a informação que o procedimento de inicialização da junta ainda não foi concluído, e um Sim, que a junta já está na posição de inicialização, ou em sua posição zero.

É conveniente lembrar que os sensores *Encoders* que utilizamos no projeto são do tipo incremental, portanto toda vez que o sistema for desenergizado e a informação do posicionamento do robô for perdida, existe a necessidade de recuperá-la. Isso é feito com o auxílio dos sensores de fim de curso instalados estrategicamente para definir os finais de cursos de movimento de cada junta. Por convenção, estes programas de inicialização das juntas, quando processados, ativam os seus motores para girarem em sentido anti-horário, em velocidades baixas, o suficiente para que não haja perigo de danificação da estrutura, quando cada motor for novamente parado pelo próprio programa, ao ser identificado o sinal eletrônico que indica posicionamento da junta sobre seu fim de curso correspondente. Portanto, durante este procedimento cada Escravo fica constantemente verificando este sinal que está presente em um *bit* de uma de suas portas de entrada digital. Uma vez identificado este sinal, imediatamente o programa faz o respectivo motor parar, e envia um comando para zerar os contadores que indicam o número digital equivalente ao posicionamento da junta respectivamente a este final de curso, e injeta o valor FFh na bandeira de indicação, dizendo Sim ao Mestre com respeito ao procedimento de inicialização da junta já ter sido concluído. O mestre por sua vez, durante este tempo, fica supervisionando estas posições de memória bandeiras dos Escravos, com acessos periódicos à suas memórias, e o programa de inicialização selecionado é posto em execução por ocasião de cada energização do sistema, ou pela opção através da janela de execução do menu principal de execução; este procedimento só termina quando todos os Escravos tenham concluído suas tarefas de inicialização expostas. Neste momento o sistema terá obtido novamente a informação do posicionamento do robô, e então pode-se colocá-lo para realizar movimentos com a informação precisa do seu posicionamento.

Existe uma diferença fundamental entre os procedimentos de inicialização do robô via energização e via seleção pelo ítem de menu de execução, porque evidentemente no caso de energização os Escravos devem primeiramente proceder a execução de seus programas monitores, que fazem a devida programação dos componentes que compõem seus circuitos digitais, e preparam cada servomecanismo para ser posto em operação, e só então é que o Mestre dispara a execução da inicialização do robô. Cada Escravo também possui outra posição de memória bandeira que serve para indicar se já foi executada, ou não, a programação dos componentes que compõem seu *hardware*, e se está apto a funcionar como Escravo, que o Mestre pode verificar a qualquer momento de necessidade.

Esta posição inicial que o robô assume logo após ter sido executado este procedimento de inicialização exposto, é uma posição não muito confortável para iniciar a execução de uma tarefa, porque como é uma posição limite do seu volume de trabalho, ele fica limitado a obrigatoriamente mover primeiramente suas juntas em sentido horário. Para evitar isso, escolhemos uma nova posição inicial que coloca cada junta em sua posição mediana com respeito aos seus limites de curso. Estas posições medianas são bem conhecidas, porisso para executar tal procedimento adicional, o Mestre primeiramente tranfere um programa de controle padrão para cada Escravo, que pode ser tipo PID, envia os dados equivalentes a estas posições como níveis de referências para estes servomecanismos e habilita cada um a proceder o controle, e a posicionar o robô adequadamente. Para que isso não provoque movimentos muito bruscos da estrutura, na realidade os níveis de referências enviados pelo Mestre são gradativamente enviados de forma adequada até atingir-se o nível de referência que corresponde a posição mediana de cada junta.

Certamente existem ainda muitas outras facilidades que podem ser

incorporadas para estes procedimentos de inicialização, realizando maiores quantidades de testes, reservando mais posições de memórias bandeiras para indicar condições que ajudem a encontrar mais facilmente onde encontra-se o robô, em caso de seleção deste procedimento por ítem de menu, e que podem tornar o processo mais rápido. A intenção prioritária neste ponto da pesquisa, entretanto, é a proposta de que o sistema permita que o usuário possa facilmente recuperar o conhecimento do posicionamento do robô, em caso de perda dessa informação (**figura 1.36**), e que o robô possa ser posto para estar pronto para ser operado totalmente de forma automática ao ser energizado. Ainda, o mesma SCHM, durante estes procedimentos de inicialização fica informando na tela gráfica sobre o que está sendo realizado para manter o usuário informado, e indica sobre eventuais problemas que possam impedir o posicionamento do robô em sua posição de inicialização com êxito.

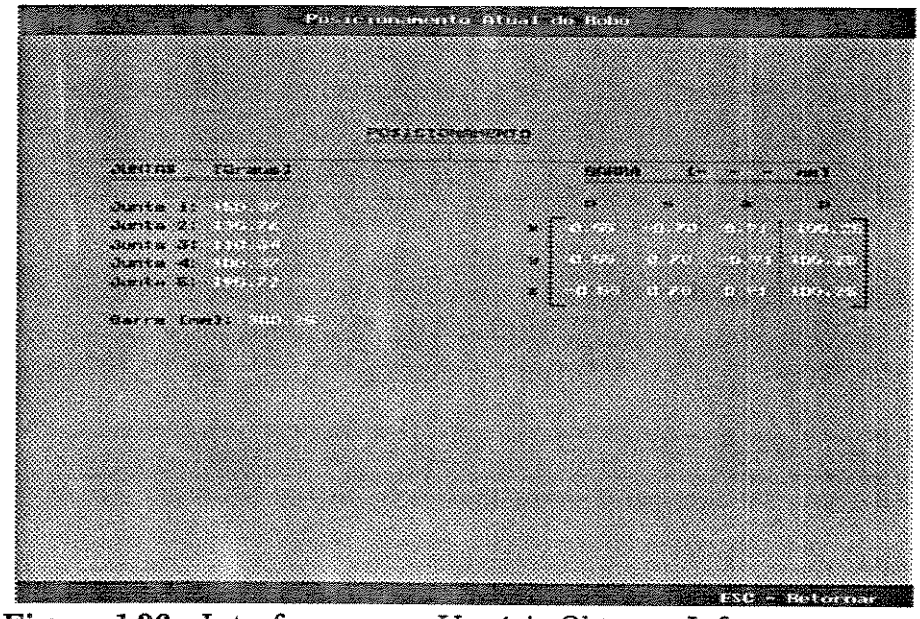


Figura 1.36 - Interface para o Usuário Obter as Informações do Posicionamento do Robô.

Outro ítem da janela de execução é chamado de Executar Tarefa Planejada. Quando ele é selecionado, o sistema faz a utilização de todos os dados programados na janela de planejamento para atuar sobre a estrutura do robô, e fazê-lo mover-se. Há sempre uma etapa inicial, onde são tranferidos todos os algoritmos de controle selecionados para os Escravos, ainda com o robô parado. Essa tranferência é feita seqüêncialmente para o caso de serem escolhidos algoritmos distintos para cada um. São feitas também as inicializações das posições de memória bandeiras, que serão utilizadas pelo Mestre para supervisionar os movimentos. Feito isso, começa o processo da geração dos níveis de referências que juntos formam a trajetória, tudo de acordo com o algoritmo elegido, e de acordo com a tarefa programada.

Como cada algoritmo de controle dos movimentos tem características particulares, o modo que será conduzido o controle dependerá do algoritmo, e é feito com que o Mestre proceda o controle a nível supervisório de acordo com a programação definida, e já residente no seu dísco rígido. Estes mesmos programas já têm intruções que fazem as devidas aberturas de arquivos para guardar dados do processamento e a evolução dos valores assumidos pelas variáveis de interesse, tanto a nível de trajetórias

como a nível de juntas. Tudo isso é definido pelo programador de alto nível, e adequado às condições oferecidas pelo pacote de *software* global, que talvez seja a tarefa mais difícil de programação.

Também neste ítem durante a evolução dos movimentos são apresentadas informações atualizadas na tela gráfica do monitor de vídeo, para que o usuário possa acompanhar com detalhes o desempenho do robô. E é dada a ele a possibilidade de fazer o robô parar de mover-se no instante que desejar, com o pressionamento de teclas do teclado de interface do Mestre.

Nossa idéia também é fazer com que antes de iniciar a execução de uma tarefa planejada, o sistema SCHM interaja com o usuário fazendo perguntas sobre o que deve ser guardado em memória para posterior análise, e em função disso, com auxílio da programação previamente estabelecida, e controle por sinalizadores bandeiras, atuar de modo correto, fazendo a abertura de arquivos e os armazenamentos de dados necessários. Este ainda é assunto de pesquisa, porque como é um problema de tempo real, qualquer acréscimo de comandos na programação implica num aumento de tempo necessário para processá-lo, porém acreditamos que algoritmos eficientes de controle como o algoritmo de busca heurística proposto nesta tese, por terem características de processamento muito rápido possam oferecer tranqüilamente tais possibilidades.

Para o desenvolvimento destes programas é muito importante que seja previamente feito um plano das variáveis globais a serem utilizadas, fazer uma escolha de nomes sugestivos para facilitar sua depuração, também fazer um plano da memória total que é necessária para fazer todos os armazenamentos de dados, subdividir de forma adequada as partes dos programas, definindo funções que podem ser eventualmente chamadas para executar procedimentos específicos como, por exemplo, abertura e fechamento de arquivos no disco rígido, interpretação de cadeias de caracteres entrados por teclado, preparação de telas gráficas, cálculos matemáticos, etc... Isso facilita enormemente a manutenção do *software*, e não cria a necessidade de produção de arquivos executáveis muito extensos, que não é uma característica muito apreciada pelos programadores, além de prejudicar a modularidade do pacote. É interessante lembrar que um bom pacote de *software* é composto de muitas partes bem definidas, como uma parte executável principal, arquivos de saídas e entradas (*drivers*) para o controle específico de periféricos, bibliotecas de dados, arquivos executáveis com coordenação do arquivo executável principal para apoio e aplicação específica dentro do pacote como, por exemplo, preparar dados para saídas gráficas, fazer cálculos de equações, etc... Estes arquivos recebem dados e instruções do arquivo executável principal, tratam estas informações, podendo inclusive atuar sobre o *hardware* do sistema, e após devolvem o controle ao executável principal.

Na escolha do ítem Ensinar Via Teclado, o usuário pode fazer a programação da trajetória que o robô irá rastrear com o auxílio visual dos próprios movimentos do robô, e de uma tela gráfica que é constantemente atualizada através das condições apresentadas pelo *hardware*, e das tomadas de decisões feitas por ele, através do teclado do Mestre. Nossa intenção é também construir uma interface apropriada para entrada de dados com teclas especialmente projetadas para esta finalidade, com extensão por cabo, e com comunicação serial com o Mestre para que o operador possa estar próximo ao robô enquanto realiza este tipo de programação. No momento, entretanto, estamos utilizando apenas o teclado padrão para isso, através de suas teclas de setas, de funções, de entrada, e algumas alfa numéricas para seleção de sub-ítens de menu, que permitem fazer esta programação, e lançar a idéia.

Uma vez selecionado este ítem, primeiramente aparece uma tela de advertência, propositalmente colocada para alertar o programador sobre se o robô já está inicializado ou não, porque cada vez que ele for programado por este método, é fundamental que o sistema tenha atualizado de modo preciso as coordenadas de posição de cada junta, para a programação ser realizada com sucesso (**figura 1.37**).

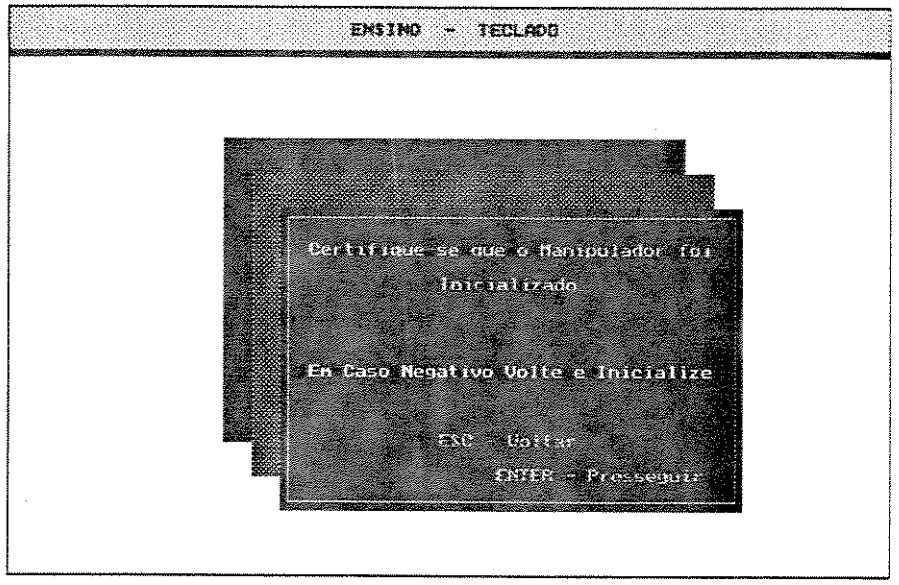


Figura 1.37 - Advertência para Ensino Via Teclado.

Neste momento, se o programador tiver dúvidas, ele pode retornar ao respectivo ítem de menu de execução, selecionando-o dentro da janela de execução, e retornando em seguida novamente para este ponto, ou então, se estiver seguro que o robô está bem inicializado, apenas deve pressionar uma tecla de controle para que o SCHM apresente a tela básica sobre o ensino do robô via teclado (**figura 1.38**).

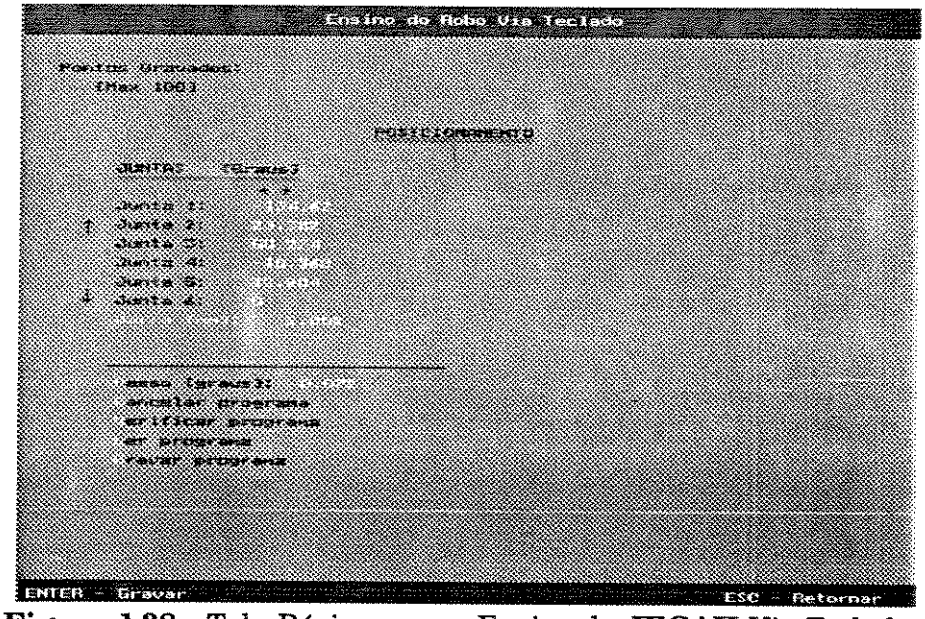


Figura 1.38 - Tela Básica para o Ensino do **JECAII** Via Teclado.

Nela o programador pode selecionar o ítem Passo, e escolher o passo adequado, em graus, que cada movimento de junta será realizado. Estes passos já estão previamente definidos interiormente pela programação, em função das precisões oferecidas pelos sensores *encoders* e pelo *hardware* de seus circuitos de interpretação, e são múltiplos de sua maior precisão. Para fazer as escolhas adequadas, basta utilizar as setas de controle até encontrar a definição requerida. Isso pode inclusive ser feito a cada novo movimento se o programador assim desejar. As alterações são feitas imediatamente após o pressionamento das teclas de controle, e cada pressionamento desses provocará o movimento de um passo, com a amplitude programada, na respectiva junta que aparece em destaque na coluna da tela gráfica destinada para esta indicação. Cada movimento apenas é feito com uma junta por vez. Como a junta da garra possui curso diferente das juntas rotacionais, e seu movimento é de translação, as modificações em seus passos também seguem uma proporção diferente, mas isso é feito automaticamente pela programação. Nesta mesma parte dos procedimentos, o programador pode escolher sobre qual a junta que deverá ser movida no próximo pressionamento das setas para a direita e para a esquerda, fazendo a utilização das teclas de para cima e para baixo, que provocam um deslocamento respectivo da junta em destaque (*SCROLL*) a cada respectivo pressionamento, sendo que ele é circular. Este destaque é simplesmente apresentado com o ítem da junta sendo posto em cor distinta das demais (vermelho → destaque, branco → normal). Quando a junta e o passo estão devidamente definidos, cada pressionamento de seta para a direita ou para a esquerda provoca um movimento do robô , e ao lado do respectivo ítem em destaque, é feita a atualização da posição atual atingida pelo último movimento, baseada na leitura dos sensores e da respectiva conversão para as coordenadas angulares. Assim, o programador poderá conduzir o robô para qualquer ponto que faz parte do seu volume de trabalho, e com a precisão que for necessária, dentro da máxima precisão oferecida pela sua estrutura eletro-mecânica. Note que estes movimentos exigem o controle constante sobre cada servomecanismo de junta, porisso é necessário que o sistema deixe ativos todos os Escravos, cada um com sua programação adequada, durante os procedimentos de ensino via teclado. Um novo passo é convertido em uma nova referência para os respectivos Escravos, que realizam os rastreamentos particulares delas sobre seus servomecanismos.

Atualmente foi deixada a possibilidade de gravação em memória de até 100 pontos, que formarão a trajetória por onde o robô deverá passar quando for requisitado para fazer o rastreamento de acordo com o que lhe foi ensinado. Mas, facilmente, este número de pontos pode ser aumentado, apenas modificando alguns dados na programação, e aumentando a quantidade de memória destinada ao armazenamento. O armazenamento é feito num arquivo especial, aberto no disco rígido, que contém uma matriz com seis linhas, cada linha destinada a armazenar os dados de uma junta, e tantas colunas quantos forem os pontos reservados para armazenagem da trajetória (atualmente 100 colunas). Quando o robô estiver adequadamente posicionado sobre um ponto que formará parte da trajetória, o programador pode gravá-lo através de um simples pressionamento de tecla. Neste momento a matriz de armazenagem é atualizada com a inserção dos dados indicados pelos sensores de posição devidamente convertidos, e nas respectivas linhas da matriz reservadas para cada junta, e coluna indicada pelo índice que vetora o próximo ponto a ser programado. Este índice também é atualizado a cada novo ponto gravado, e seu valor atualizado fica apresentado no canto superior esquerdo da tela gráfica.

Pode acontecer que o programador cometa algum erro, e proceda a gravação de um ponto errado, ou mesmo que decida retirar algum ponto gravado. Para estes casos, colocamos uma outra possibilidade, que pode ser selecionada por teclado, e permite fazer-se a verificação ponto-à-ponto da programação, e a retirada de pontos programados. Isso

pode ser feito através do acesso ao ítem Verificar Programa. Feita esta escolha, imediatamente aparece um quadro na tela gráfica apresentando o ponto atual (último ponto programado), com os respectivos dados de posicionamento de cada junta, e a composição da matriz que define o posicionamento e a orientação do elemento terminal para a atual configuração das juntas apresentada (**figura 1.39**).

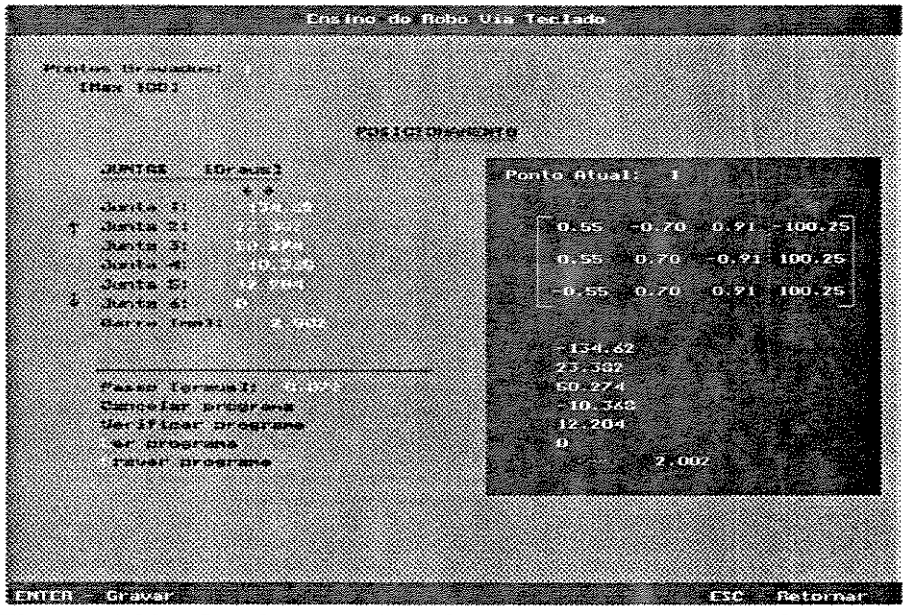


Figura 1.39 - Quadro para Verificação da Programação feita pelo Ensino do Robô Via Teclado.

Dentro deste quadro de verificação do programa, o programador pode fazer uma varredura para cima ou para baixo de pesquisa sobre todos os pontos programados, sendo que o sistema faz a atualização imediata do quadro mostrado. Se o programador quer retirar um determinado ponto programado, basta que ele faça a pesquisa citada até que o quadro de visualização apresente o ponto a ser retirado das posições de memória que ocupa dentro da matriz de programação. Se este for um ponto intermediário, a programa de controle realiza a atualização da matriz fazendo um deslocamento para a esquerda de todos os pontos seguintes programados, para preencher esta coluna da matriz que foi retirada, de forma que a seqüência de pontos restantes permaneça a mesma, somente sem o ponto que foi retirado.

Para retornar à tela de programação é só pressionar uma tecla de controle, e o quadro de verificação da programação é retirado, habilitando-se novamente o modo de Ensino. Ainda como nota, no modo de verificação, o robô é movido sendo reposicionado para o ponto apresentado no quadro, e a cada ponto que for retirado, ele é posto sobre o ponto imediatamente anterior programado, isso para que o operador possa também fazer a visualização das configurações físicas que o robô adota em cada ponto programado. Porisso também, a pesquisa sobre a programação é feita de ponto para ponto programado e de modo seqüêncial. Quando o programador abandona o modo de verificação da programação, o programa de controle reposiciona o robô sobre o último ponto programado, para que o programador possa seguir com o ensino.

Ainda na tela gráfica de interface para o ensino do robô via teclado, existe a possibilidade de poder-se apagar toda a programação realizada. Esta escolha,

primeiramente coloca um sinal de advertência sobre se o operador está seguro do que quer, com a intenção de dar a possibilidade de recorrência, se acaso o pressionamento da respectiva tecla de controle foi acidental, e o desejo de apagamento da programação não foi intencional, neste caso, basta pressionar outra tecla de controle, e tudo retorna a normalidade. Em caso da escolha ser intencional, basta fazer uma confirmação após a escolha, e o programa de controle do ensino apagará toda a matriz de programação, procedendo uma reinicialização deste processo, preparando para a entrada de um novo primeiro ponto a ser programado, e assim por diante, conforme explicamos. Existem duas outras possibilidades que são: Ler ou Gravar no disco rígido um arquivo que contém os dados sobre uma programação de ensino realizada. Se a escolha for ler um programa do disco rígido, primeiramente aparece outra advertência, se acaso o programador está seguro do que quer fazer, e programar os pontos da trajetória diretamente através de um arquivo de entrada, perdendo todos os dados que estavam sendo gravados, isso apenas para evitar que aconteçam erros por distração, e pressionamento de teclas por engano. Neste caso, um pressionamento de tecla de controle faz retornar a normalidade, e uma confirmação coloca em ação a abertura de duas janelas na tela gráfica, uma que contém os nomes dos arquivos com programação de ensino disponíveis no disco rígido, à direita da tela, e outra que mostra o subdiretório atual do disco que foi utilizado para esta pesquisa de arquivos disponíveis, que têm extensão apropriada (*.PTO) definida internamente pela programação do pacote de *software* (**figura 1.40**).

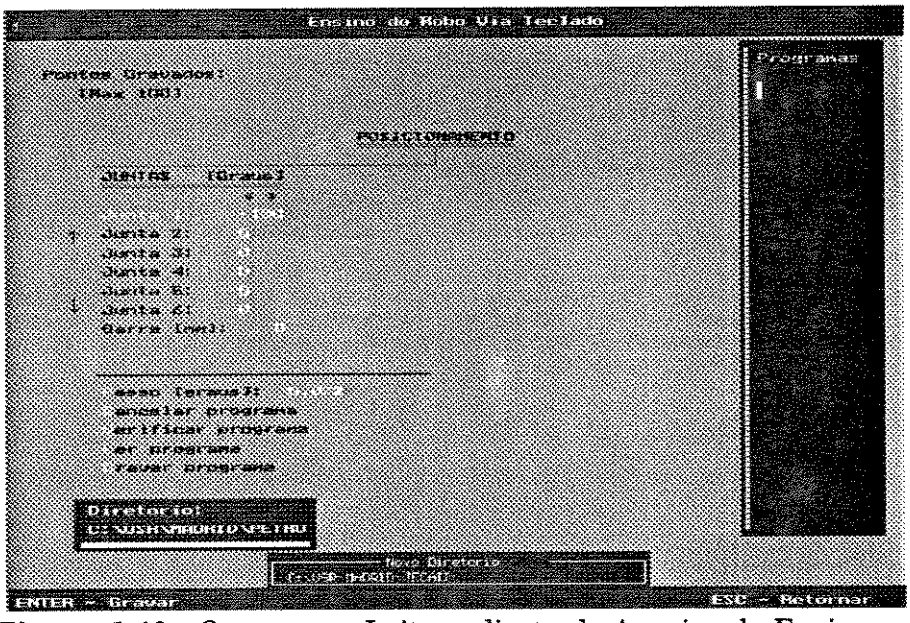


Figura 1.40 - Opção para Leitura direta de Arquivo de Ensino Residente em Memória.

Neste caso, se o programador quiser alterar o subdiretório de pesquisa destes arquivos, poderá fazê-lo selecionando o ítem Diretório, o que provocará a aparição de uma nova janela preparada para apresentar a entrada da seqüência de caracteres que irá definir este novo subdiretório para pesquisa. Esta janela apresenta a possibilidade de definir que o subdiretório seja qualquer um disponível no disco rígido, ou mesmo dos *drivers* dos *diskettes*. Quando a cadeia de caracteres estiver formada e correta, um pressionamento de tecla de controle faz a atualização automática para o respectivo subdiretório, atualizando as janelas de Diretório e de Programas com as informações coerentes. Se acaso, o programador cometer erros na digitação da entrada dos caracteres

que formam a cadeia que define o novo subdiretório, então ele pode recorrer à utilização das teclas de controle para reeditar, e se acaso o subdiretório requisitado não existe, ou não seja possível de ser acessado, aparecem comentários de advertência na tela gráfica explicando a devida impossibilidade.

Se tudo estiver correto, o programador pode escolher qual arquivo quer ler com o auxílio das teclas de controle que fazem a seleção de forma circular, aparecendo na tela com uma barra branca sobre o respectivo arquivo, que aparece na janela de Programas, e que atualmente está sendo selecionado. Se for dado o pressionamento de outra tecla de controle, o arquivo que estiver aparecendo como selecionado é automaticamente carregado, formando a matriz de programação de ensino, e a programação faz o retorno para a devida tela básica de ensino que permite verificar, avaliar e modificar aquilo que está programado.

Para gravar um novo arquivo de dados sobre o ensino, basta também selecionar o respectivo ítem oferecido pela tela de interface, que faz a pergunta sobre a cadeia de caracteres que definirá onde o arquivo deve ser gravado, e com qual nome. Se for dada a entrada somente do nome, então o programa de controle de ensino procederá a gravação deste arquivo no subdiretório padrão reservado para conter arquivos com nomes *.PTO, caso contrário destinará o arquivo para o subdiretório formado pela cadeia de caracteres entrados pelo teclado. Tudo isso após um pressionamento de tecla de controle, sendo que os nomes dos arquivos escolhidos não necessitam ter extensão, porque ela é injetada pelo próprio programa de controle do ensino.

Quando o programador seleciona o ítem de Ensino Via Teclado no menu principal de execução, o sistema automaticamente prepara-se para que a execução da tarefa planejada seja feita baseada neste ensino, de modo que um novo pedido de movimentação do robô por esta escolha fará um pedido sobre qual deve ser o arquivo de ensino que deverá ser utilizado. Porisso, a cada vez que o programador quiser abandonar os procedimentos por ensino via teclado, ele deverá retornar pelo menos uma vez para as janelas de planejamento ou simulação, que preparam as execuções para outros respectivos modos.

O modo de execução da programação ensinada por teclado, depende de fazer-se interpolações entre cada ponto, porisso, o algoritmo programado para isso é executado baseado em algum procedimento desses. Como existem muitas formas de fazer interpolações, pretende-se deixar a possibilidade de escolha do tipo através de seleção de ítem por mais uma janela que deixará a disposição todos os tipos programados; Ex. interpolações lineares, quadráticas, cúbicas, espirais (cotóides), por polinômios de ordens mais elevadas, com as devidas programações de parâmetros e definições de velocidades e acelerações entre tramos da trajetória a ser rastreada. Como este não foi um ponto diretamente desenvolvido por esta pesquisa de tese, apenas deixamos preparado o projeto de programação para a sua fácil inserção. Assunto de pesquisa este que ainda deverá ser estendido e devidamente preparado para que o robô tenha desempenhos satisfatórios quando programado por este método de ensino para realizar tarefas úteis.

Para o retorno desta tela de ensino do robô via teclado, basta o pressionamento de uma tecla apropriada quando estiver sendo apresentada sua tela básica de visualização. Ainda deverão ser incorporadas outras facilidades principalmente com respeito a informações de ajuda e de advertências para promover a programação rápida, e com menos erros provocados por escolhas indevidas de procedimentos, etc... E também intencionamos promover o ensino através da utilização sincronizada da cinemática inversa

do robô para poder-se programá-lo com movimentos coordenados das juntas fazendo deslocamentos por passos diretamente em coordenadas cartesianas, para por exemplo, movê-lo em combinações ponderadas de X, Y, e Z para facilitar a programação de movimentos retilíneos neste espaço, e de outras curvas, como círculos, parábolas, etc... que são mais freqüentemente utilizadas, apresentando uma tela gráfica variante desta que apresentamos, adequada ao ensino para esta opção de movimentos.

O ítem Executar Simulação foi colocado especificamente para executar o que o seu próprio nome informa. O procedimento executado é muito parecido com o que foi descrito nos comentários feitos para a janela de simulação do menu principal do SCHM, apenas que, ao invés de se utilizarem dados gerados através da modelagem matemática, são utilizados os dados, devidamente convertidos, dos próprios sensores instalados estrategicamente na estrutura. E os dados gerados para o controle da execução de uma tarefa, ao invés de apenas serem apresentados na tela gráfica, agora são enviados para os devidos Escravos, e mais especificamente para as suas portas de saídas que interagem com cada sistema de acionamento de junta, para fazê-las moverem-se sob esta coordenação. Aqui funciona todo o processo de comunicação de dados entre os componentes do *hardware* explicado nos **Anexos A e C**. O sistema pode ser programado para fazer o armazenamento de dados sobre a evolução temporal das variáveis de interesse para possibilitar a realização das devidas análises posteriores sobre o desempenho atingido na execução de cada tarefa.

O ítem Localizar o Manipulador foi colocado com a intenção do usuário poder conferir o seu posicionamento dentro do volume de trabalho. Nesta saída gráfica são apresentadas informações sobre o posicionamento de cada junta relativamente às suas coordenadas de referência, e a respectiva matriz que informa sobre o posicionamento e a orientação do seu elemento terminal (**figura 1.41**). Esta mesma saída gráfica pode ser requisitada em outros momentos da utilização da programação, e de acordo com as necessidades pode-se fazer esta representação através da utilização dos ângulos de *Euler*.

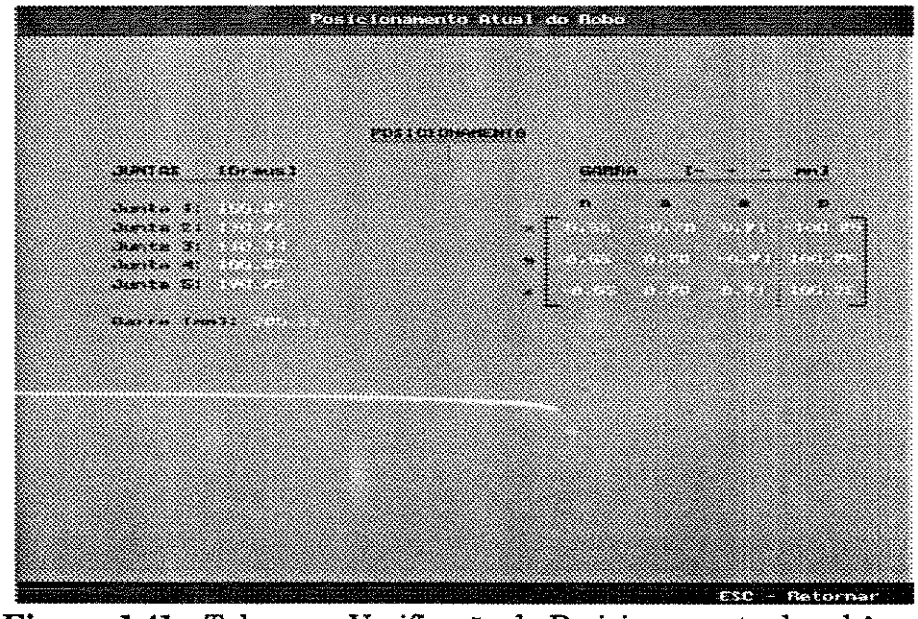


Figura 1.41 - Tela para Verificação do Posicionamento do robô.

Na escolha de fazer Auto Teste, o usuário poderá verificar o estado de

funcionamento de toda a parte eletrônica do sistema de controle do **JECAII**. Aqui ele poderá verificar as condições das memórias do Mestre e de cada Escravo necessárias ao bom funcionamento, com teste exaustivo sobre cada posição. Verificar se tudo está correto com o sistema de acionamento de cada Escravo, verificar se o sistema de comunicação está em ordem, verificar se o sistema de aquisição e interpretação dos dados dos sensores está funcionamndo corretamente. Verificar as condições de carga do sistema de baterias, e verificar as condições de configuração de *hardware* do Mestre. Cada opção poderá ser escolhida com auxílio de uma janela que apresenta os respectivos ítens, e são apresentados comentários sobre eventuais problemas encontrados durante os procedimentos de testes, e sobre o funcionamento correto do ítem selecionado. Se algum dos ítens apresentar algum comentário indicando mal funcionamento, o robô deve ser desenergizado e deverá ser realizada a manutenção na respectiva parte afetada, para que o sistema global possa funcionar adequadamente. Este procedimento de Auto Teste também é invocado pela supervisão da programação cada vez que o robô for energizado, procedendo-se uma verificação automática ítem-à-ítem, e em caso de ser encontrado algum problema é apresentada, na tela gráfica do monitor de vídeo, a respectiva mensagem de aviso sobre os setores com problemas, e sobre os tipos de problemas encontrados.

Colocamos um ítem para Demonstrar Movimentos, para fazer apresentações das potencialidades do sistema de controle do robô. Aqui o usuário terá a possibilidade de selecionar algum entre os programas escritos para procederem a execução de movimentos sobre a estrutura que permitem a constatação visual de como o robô pode ser posto para funcionar, que mostram a precisão, suavidade, velocidade, etc... com que os seus movimentos podem ser realizados, e podem dar uma idéia realista, e colaborar para a formação de opinião sobre sua capacidade de realizar tarefas de ensino e industriais, projetados com finalidades específicas de demostração para explorar cada ítem. Esta janela pode abrigar quantos ítens se queira, apenas com a inserção do respectivo comentário em uma linha de um arquivo que contém estes dados, e do seu respectivo alocamento dentro do pacote de *software*.

Ainda na janela de execução o usuário poderá fazer a gravação no disco rígido dos dados atuais em utilização, em cada momento que estiver utilizando o pacote de programação e controle do **JECAII**, apenas selecionando o ítem Gravar Arquivo desta mesma janela, que fará um pedido semelhante ao descrito no ítem de Ensinar Via Teclado, com abertura de uma janela apropriada a apresentação da cadeia de caracteres que formam o subdiretório e o nome do arquivo a ser gravado, com as mesmas possibilidades de edição, e apresentação de erros, e que cria arquivos que podem ser relidos através do ítem de Leitura de Arquivo da janela principal de Planejamento do menu principal do SCHM, para sua futura utilização.

E o último ítem da janela de execução deve ser selecionado toda vez que o usuário desejar abandonar a utilização deste pacote, este ítem tem o nome de Retornar ao DOS, indicando que o controle de operação do Micro Mestre será passado diretamente para os comandos do sistema operacional DOS. Mas ainda antes da saída é apresentado um comentário de advertência sobre se o usuário está seguro de querer abandonar o pacote. Em caso afirmativo, basta confirmar, e o pacote imediatamente será abandonado, em caso contrário, o simples pressionamento de qualquer outra tecla fará o cancelamento da saída. Este recurso é para que o usuário não seja obrigado a abandoná-lo em função de algum engano de digitação.

Quando o sistema tiver sido posto para fazer alguma simulação ou execução

de uma tarefa, ou se o usuário tiver feito a leitura de algum arquivo para entrar diretamente com dados gravados sobre alguma simulação ou execução feita anteriormente, e gravada no disco rígido, ele poderá optar pelo ítem Análise oferecido no menu principal do SCHM, e então poder recorrer a verificação de Gráficos, Listar Dados, e pedir Estatísticas. Se entrar no ítem Gráficos, a interface oferecerá oito alternativas, ou seja, apresentação de gráficos de Posições XYZ, Velocidades XYZ, Acelerações XYZ, como saídas calculadas das evoluções destas respectivas variáveis do elemento terminal no espaço cartesiano; Gráficos de Posições de Juntas, Velocidades de Juntas, e Acelerações de Juntas, apresentando a evolução destas respectivas variáveis em seus espaços de trabalho; Gráficos de distâncias e erros respectivos à evolução do posicionamento do elemento terminal em relação a cada coordenada de passagem definida pelo planejemento da trajetória a rastrear.

Para os gráficos que apresentam as posições XYZ, existe uma nova possibilidade de escolha com quatro modos de apresentação. Um sobre a Evolução da Trajetória de Posição XYZ, em modo 3D semelhante ao comentado no ítem sobre as simulações conforme mostra a **figura 1.42**. E outros três, cada um apresentando uma das projeções: XY (**figura 1.43**), XZ (**figura 1.44**), e YZ (**figura 1.45**).

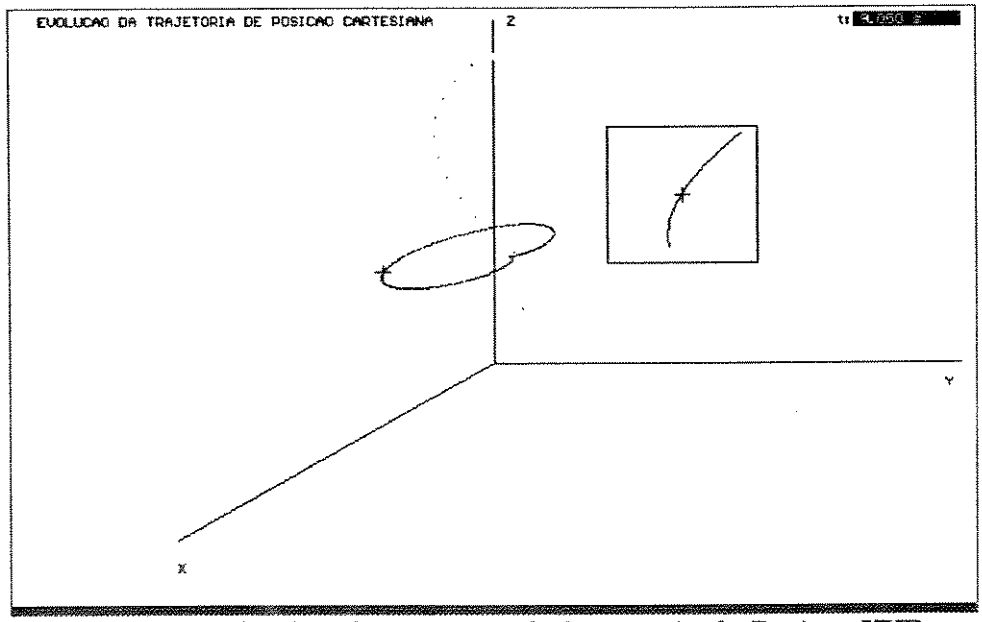


Figura 1.42 - Gráfico de Evolução de Trajetória de Posição XYZ. Modo 3D.

Cada modo desses pode ser selecionado através de teclas de funções, procedendo sua imediata apresentação na tela gráfica. Ainda estas telas gráficas apresentam o recurso da movimentação de um cursor com utilização das setas de controle, pelas quais, o usuário pode fazer este cursor mover-se sobre os pontos da trajetória digitalizados, e apresentados em cada tela. E para que seja possível obter informações mais detalhadas sobre os reais acontecimentos durante as simulações e execuções, incorporamos um método de amplificação (*ZOOM*) que pode ser invocado com o pressionamento de outra tecla de função. Com isso, aparece na mesma tela, uma janela quadrada, que pode ser deslocada para qualquer posicionamento dentro desta tela, e movida segundo o passo indicado na parte central inferior da tela, e alterado de modo circular, em passos múltiplos de 5, até 125 passos.

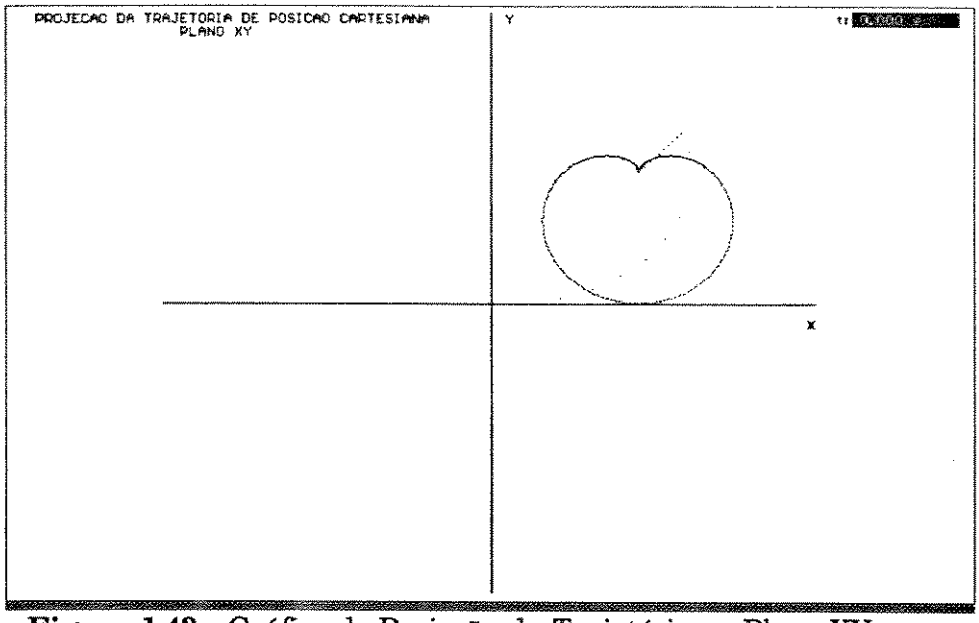


Figura 1.43 - Gráfico de Projeção da Trajetória no Plano XY.

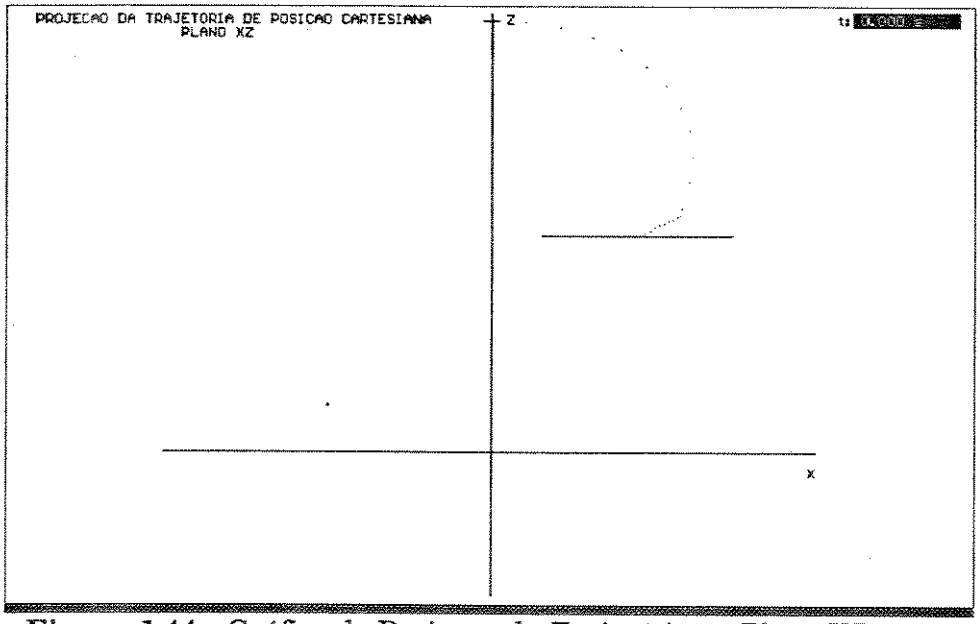


Figura 1.44 - Gráfico de Projeção da Trajetória no Plano XZ.

Isso foi permitido para que a mesma janela de amplificação não ofusque o gráfico apresentado pela evolução da trajetória, ou que possa ser colocado próximo a uma região de interesse para análise. Dentro da janela de amplificação, aparecem amplificados por um fator *4X* os dez pontos programados anteriores, e os dez posteriores à posição onde se encontra o cursor; isso permite uma visualização muitíssimo mais precisa daquela que é apresentada na tela ampla, que fica limitada pela definição possível através da placa de controle do monitor de video, que no caso é de 650 X 480 *pixels*.

Para ainda melhorar o aspecto de visualização, dentro desta janela de amplificação fazemos a interpolação linear entre os pontos digitalizados, que deve ser bem interpretada pelo usuário, porque entre os pontos não necessariamente os movimentos são retilíneos. A intenção aqui é apenas a visualização, e dar uma idéia razoável do

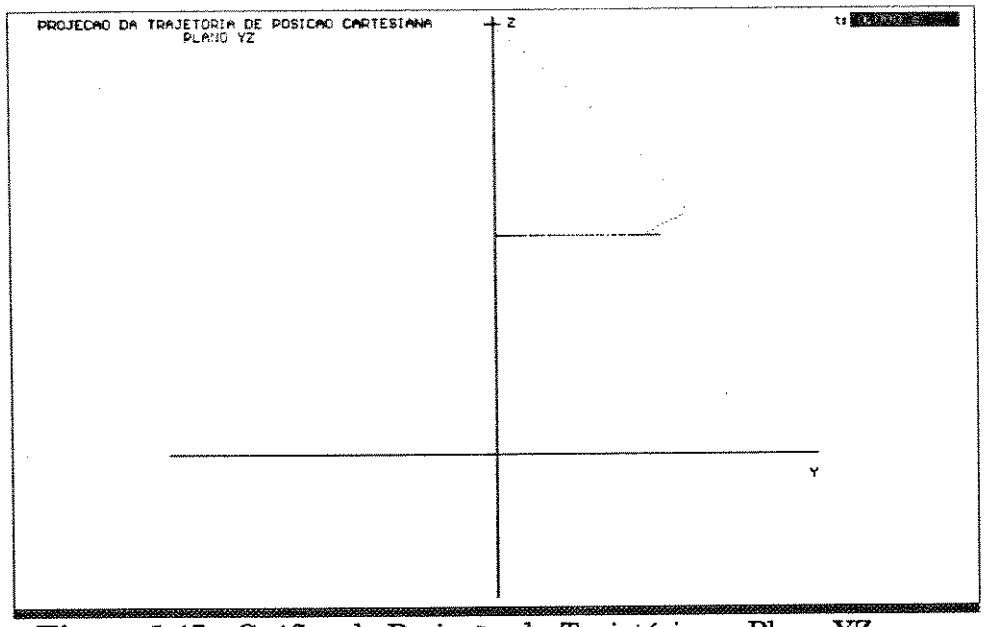


Figura 1.45 - Gráfico de Projeção da Trajetória no Plano YZ.

movimento ocorrido.

Como é um fator interessante para a análise, que um ponto marcado em uma projeção possa ser facilmente verificado em outra das projeções, nós fizemos com que as apresentações destes respectivos gráficos dos posicionamentos cartesianos estejam sincronizadas pelo posicionamento do cursor; ao ser elegida a apresentação de uma outra projeção, inclusive o sistema de apresentação da janela de amplificação também é sincronizado da mesma forma. Outro dado importante foi colocado no canto superior direito da tela gráfica, que informa sobre o tempo decorrido, em que o determinado ponto acessado pelo cursor, foi atingido pelo processo de simulação ou execução da tarefa.

Nos sub-ítens para pedido de apresentação dos gráficos de velocidades e acelerações cartesianas, foram reservadas apenas uma tela gráfica para cada um, que podem apresentar até quatro curvas. Ou seja, velocidade na coordenada X do elemento terminal, na coordenada Y, e na coordenada Z, e ainda a curva da velocidade tangencial calculada como $Vt = \sqrt{vx^2 + vy^2 + vz^2}$ Ver **figura 1.46**. O mesmo vale para as acelerações cartesianas. Existem vários recursos incorporados a estas apresentações gráficas para melhorar os procedimentos de análise sobre elas. O primeiro deles é a anexação de um cursor, colocado como uma linha vertical de largura 1 *pixel* movido pelas setas de controle, que fazem os respectivos deslocamentos deste cursor sobre as curvas, de acordo com um passo que é apresentado no centro inferior da tela gráfica, e que pode ser alterado de modo circular em múltiplos de 5 até 125 vezes. A posição que o cursor assume sobre as curvas cria a atualização das respectivas informações sobre suas amplitudes, na região destinada para isso, no canto inferior direito da tela gráfica, e são apresentados 500 pontos para visualização, de forma que se a trajetória descrita possui mais de 500 pontos digitalizados e gravados no disco rígido; Se o cursor for levado para acima destes 500 pontos, então ocorre um giro horizontal na apresentação gráfica, colocando novos 500 pontos em função da região onde se encontra o cursor; o mesmo vale para se acaso houver o retorno do cursor para a esquerda, uma vez que ele tenha sido levado para posições superiores, com um giro horizontal inverso. Juntamente com cada movimento feito pelo cursor é feita também a atualização do respectivo tempo da

ocorrência de tal situação, no canto superior direito; esta atualização é baseada no dado programado para a simulação ou execução da trajetória referente ao período de amostragem da respectiva malha de controle.

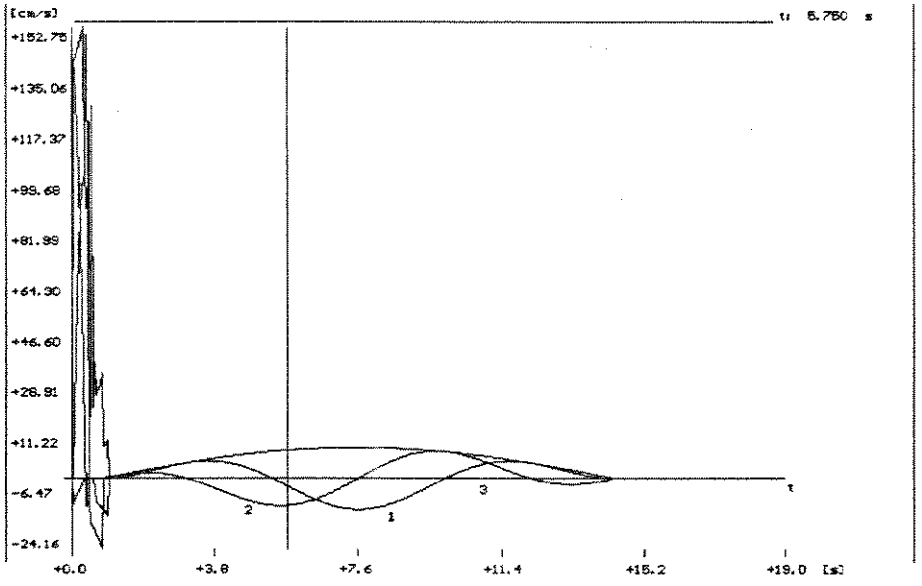


Figura 1.46 - Gráfico de Velocidades Cartesianas.

Através das teclas de função, pode-se ativar ou desativar a aparição gráfica de cada uma das respectivas curvas apresentadas. Isso serve para melhorar a definição, ou a comparação mais detalhada entre elas, e também para preparação de impressão. Também é possível chamar-se uma janela de amplificação, que permite a amplificação de *5X* em torno da posição indicada pelo cursor, isso permite visualizar com maior precisão as variações rápidas ocorridas durante a evolução das variáveis. Esta janela de amplificação também sempre se desloca juntamente com o cursor.

Outro recurso é a apresentação de uma determinada região de pontos utilizando o recurso de compressão/expansão da tela, invocado pelo pressionamento de outra tecla de função, que pergunta primeiramente qual é o ponto inicial para apresentação; então o usuário deve deslocar o cursor para este ponto de interesse, e confirmar; em seguida, o sistema pergunta para o usuário qual é o ponto final para apresentação; então novamente ele deve deslocar o cursor até a posição final desejada, e confirmar. Depois disso, automaticamente a tela é atualizada fazendo a devida compressão ou expansão para que todo o intervalo desejado seja apresentado. Esse procedimento pode ser muito importante para produzir-se amplificações maiores entre os pontos, para detetar, por exemplo, freqüências mais elevadas dos sinais, e também para permitir a visualização de regiões maiores de pontos numa mesma tela.

Ainda são permitidas alterações no formato de escala de amplitude de apresentação das curvas, isso pode ser em outra escala, em função dos valores máximos e mínimos que elas assumem, através da linha de zero no centro da tela; e impressão de valores máximos em função da sua relação de aspecto, ou amplitude máxima e mínima definida pelo usuário. Outra função prepara a tela fazendo uma conversão de colorido para preto e branco, interessante para fazer-se a impressão nesta forma, que é característica de muitas impressoras. Uma vez feita esta escolha, primeiramente o sistema pergunta ao usuário se ele deseja colocar indicadores sobre as curvas, para que a apresentação delas

em preto e branco, não cause confusão, principalmente quando serão imprimidas mais de uma; neste caso, ao ser respondido com um Sim, aparecem números no canto superior direito da tela, que podem ser deslocados com o auxílio das setas de controle, para qualquer posição que faz parte do espaço para apresentação das curvas, e cujos passos de deslocamento são função do mesmo passo que é apresentado na parte inferior central da tela gráfica, e que pode ser alterado circularmente; assim pode-se movê-los de forma rápida e de forma lenta para o posicionamento mais adequado, a critério do usuário.

Uma vez que este procedimento é ativado, o Mestre somente devolve o controle ao usuário quando tiver concluído por completo a preparação da tela. Porisso existe também uma informação de advertência antes de ser iniciado, perguntando se o usuário está seguro do que quer realizar, para que não perca tempo devido a algum engano de digitação.

As precisões numéricas apresentadas nestas telas gráficas, não são muito rigorosas, apenas por questão de espaço para as suas apresentações; porisso, se o usuário necessitar dos valores mais precisos, deverá selecionar o ítem do menu principal de Analisar referente a Listar Dados, que apresenta os mesmos ítens que a janela de gráficos, porém, com saídas em forma de listas numéricas com maior precisão, de acordo com os números reais que foram arquivados durante as simulações ou execuções do rastreamento de trajetórias.

Os gráficos de posições, velocidades e acelerações de juntas, conforme exemplos apresentados anteriormente nas **figuras 1.20, 1.21, 1.22**, possuem cada um, um ítem particular para suas seleções, neste sub-ítem de Gráficos; todos eles têm as mesmas possibilidades para o melhoramento das análises que foram apresentados para os ítens de velocidades e acelerações cartesianas.

Para os ítens de apresentação de gráficos de distâncias e erros, função da evolução dos movimentos do elemento terminal do robô, com respeito aos pontos que definem a trajetória a rastrear, também incorporamos as mesmas possibilidades dadas aos outros ítens de gráficos, apenas que estas saídas são mais simples, porque apresentam uma única curva por tela (**figura 1.47**).

Nestes gráficos também é apresentada a informação sobre a média, que é um fator muito relevante para a análise destes resultados.

Lembramos que em todos os gráficos é feita uma interpolação linear entre os pontos digitalizados, com a intenção de melhorar a visualização, que deve ser levada em consideração para uma análise mais minusciosa dos desempenhos do robô.

No gráfico de erros, o usuário pode comprovar se os erros com que cada ponto é atingido pelo robô estão dentro da precisão imposta pela programação definida no planejamento da trajetória, porque este gráfico possui auto-escalamento, e o valor indicado como limite na tela gráfica, mostra também o limite alcançado pelas aproximações do elemento terminal do robô a cada ponto.

E dentro do ítem Estatísticas, o usuário poderá recorrer a cálculos automáticos de valores médios atingidos pelas curvas, desvios padrões, determinar mínimos e máximos, e outras funções para análises estatísticas sobre as curvas de evolução das variáveis; isso é feito primeiramente com a seleção da curva de interesse,

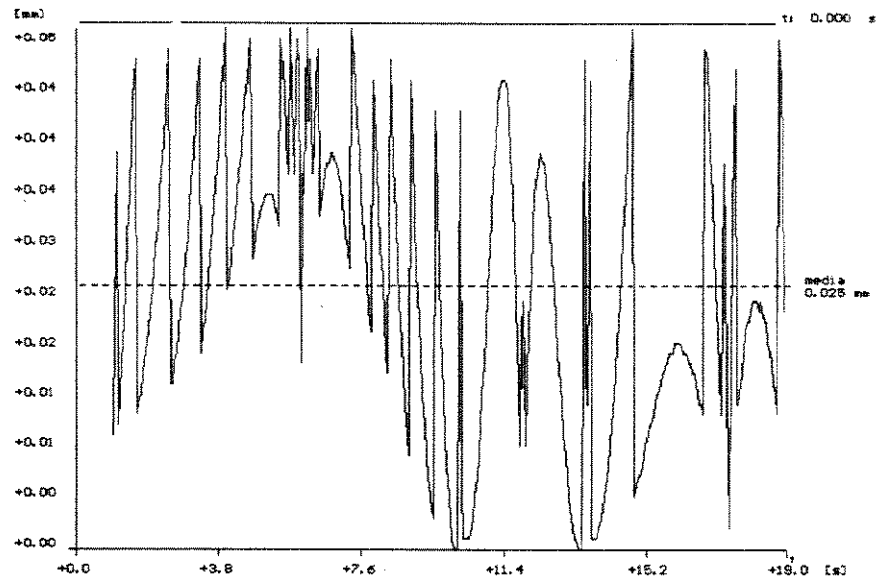


Figura 1.47 - Gráfico dos Erros de Rastreamento de Trajetória.

com a entrada através de janela apropriada, e então pedido de ítem, que procede a execução da respectiva função que compõe o pacote e encontra os valores requisitados. Aqui também pode-se acrescentar quantas funções se queira, apenas fazendo sua anexação ao pacote, e respeitando os critérios de entrada e saída de dados, com acréscimo da informação de sua existência, no arquivo que faz a indicação gráfica na tela, e faz o controle das execuções das funções.

CAPÍTULO II

ESTADO DA ARTE EM PLANEJAMENTO E CONTROLE DE TRAJETÓRIAS

Neste Capítulo fazemos alguns comentários sobre métodos já utilizados para este tipo de planejamento e controle, e propostos por outros pesquisadores que encontramos na literatura especializada, para situarmos com maior abrangência os problemas, e demonstrarmos a importância de se obterem soluções eficazes para que os robôs possam ser utilizados cada vez com melhores desempenhos em seus ambientes de trabalho.

O problema de controle de trajetórias é talvez o problema mais importante do controle da execução de uma tarefa por um robô, cujo objetivo é fazer com que sua estrutura mecânica, rastreie uma ou mais curvas que definem espacialmente e de forma contínua os pontos pelos quais seu elemento terminal deve passar, com requisitada precisão, velocidade e suavidade. Como é um problema de rastreamento, pode haver a necessidade da realização de controle em tempo-real dos movimentos para que o procedimento obtenha sucesso.

Para avaliar o desempenho dos robôs, pode-se determinar algumas de suas características importantes realizando-se observações, medições, e análises de seus comportamentos baseadas em suas posturas, onde postura é definida como a combinação da posição e da orientação da sua estrutura física. Para esta finalidade pode-se considerar os seguintes aspectos **[53]**:

- *Exatidão de Postura*: Expressa o desvio entre a postura de comando (provocada pela ação do sistema de controle), e a postura desejada (definida pela trajetória requisitada para rastreio), numa mesma direção.

- *Repetibilidade de Postura*: Expressa a concordância entre as posições e orientações da postura desejada, depois de *n* movimentos repetidos com a mesma postura de comando numa mesma direção.

- *Exatidão de Postura Multi-Direcional*: Expressa o desvio entre as diferentes posturas atingidas, quando são tentadas as mesmas posturas de comando por *n* vezes, em três direções perpendiculares.

- *Exatidão de Distância*: Expressa o desvio no posicionamento e na orientação entre a distância desejada e a distância de comando.

- *Repetibilidade de Distância*: Expressa a concordância entre as diversas distâncias atingidas através das mesmas distâncias de comando, após *n* repetições na mesma direção.

- *Tempo de Estabilização de Postura*: É o período de tempo decorrido entre o instante que o robô atinge a postura desejada, e o instante no qual o movimento oscilatório, próximo a esta postura, fique dentro de um limite especificado.

- *Máximo Desvio de Postura (Overshoot)*: É o máximo desvio entre a postura desejada, desde o primeiro momento que é atingida, e a postura mais distante provocada por oscilações, até a estabilização.

- *Variação da Exatidão de Postura (Drift)*: Variações próximas à postura desejada, durante um tempo especificado.

- *Exatidão e Repetibilidade do Caminho*:

  - A exatidão do caminho é o máximo desvio obtido, em posição e orientação, durante todo o percurso da trajetória.
  - A repetibilidade do caminho expressa a concordância entre a trajetória desejada e as trajetórias realizadas, através das mesmas trajetórias de comando, após *n* repetições.

- *Máximo e Mínimo Desvios de Esquina*: O máximo e o mínimo desvios de esquina podem ser medidos quando a trajetória possui ângulos retos no intervalo em que é definida, e neste caso, eles equivalem, respectivamente, à máxima e à mínima distâncias que o elemento terminal do robô atinge, relativo a cada ponto de vértice formado.

- *Exatidão e Repetibilidade da Velocidade do Percurso*:

  - A Exatidão da velocidade do percurso é o erro percentual entre a velocidade de comando, e a velocidade requisitada pela trajetória de rastreio, após seu completo rastreamento.
  - A repetibilidade da velocidade de percurso é a concordância entre a velocidade desejada, e as velocidades executadas durante o rastreamento da trajetória, após *n* execuções, e sob as mesmas condições.

- *Flutuação da Velocidade de Percurso*: É o máximo desvio na velocidade ocorrido durante um rastreamento completo da trajetória.

- *Mínimo Tempo de Posicionamento*: É o tempo decorrido entre a partida e a chegada ao estado estacionário, quando o elemento terminal do robô percorre uma determinada distância, através de um caminho predeterminado.

- *Capacidade de Movimentação com Carga*: É o máximo deslocamento unitário que o elemento terminal do robô pode realizar, carregado com uma carga especificada, sobre uma trajetória especificada.

Para que um robô passe por esta avaliação com bons resultados, existe a necessidade de que seu sistema de controle tenha um desempenho muito satisfatório, e isso não é fácil de se conseguir. É extremamente complicado propor técnicas gerais de solução que sirvam para controlar diversas estruturas mecânicas com características dinâmicas e acoplamentos não-lineares distintos.

Ao fazer-se o modelamento matemático para abstrair as características cinemáticas e dinâmicas de um robô, ou manipulador mecânico, que possui vários graus de liberdade, depara-se com equações dinâmicas não-lineares, e que possuem parâmetros variantes no tempo **[17][27][61][73]**, ou seja, não é fácil encontrar suas soluções, sem necessariamente, ter-se que utilizar métodos numéricos trabalhosos; esse fato gera conflito com o outro, da necessidade de solução em tempo-real. E todo o problema fica mais agravado porque, mesmo para movimentos à baixas velocidades, as tomadas de decisão que devem partir do sistema de controle, em função do movimento requisitado, e do comportamento em cada instante da estrutura, ao realizá-lo, devem ser suficientemente rápidas em função da capacidade de reação inercial dos elementos acionadores da estrutura **[30]**. Por exemplo: motores elétricos podem ter capacidade de reação de inércia mecânica em tempos menores que 100ms, inclusive, dependendo da tecnologia empregada, ainda em tempos muito menores que este. Uma solução para este problema é utilizar controladores com grande capacidade de processamento, mas que, na prática, podem tornar-se inviáveis por encarecimento demasiado do custo do sistema global. E mesmo que se consiga justificar a utilização de um sistema computacional de grande porte, ainda assim, é lógico pensar que deve-se aproveitar ao máximo seus recursos para deixá-lo o mais hábil e versátil possível.

Os métodos propostos até o momento para encontrar soluções a este problema podem ser geralmente enquadrados em dois grandes grupos de abordagem: Métodos de solução via espaço de juntas; Métodos de solução via espaço cartesiano.

As abordagens via espaço de juntas visam obter seqüências polinomiais adequadas que garantam movimentos suaves sobre as trajetórias das variáveis de juntas buscando carga computacional menor e promovendo movimentos sem variações bruscas de acelerações e velocidades. É adequada a utilização de polinômios de baixa ordem **[44]**. Geralmente a trajetória é dividida em diversos segmentos menores, e cada segmento menor substituído por um polinômio de baixa ordem.

Com esta idéia, um subproblema é encontrar quais os polinômios que fazem com que a trajetória seja rastreada com sucesso **[43]**. Evidentemente o trabalho de

particionar a trajetória adequadamente e depois para cada sub-segmento encontrado escolher um polinômio adequado, resulta tedioso. Imagine se para um pequeno trecho de trajetória for necessário dividí-lo em 1000 ou 2000 partes por imposição das precisões exigidas. Também deve-se ter em consideração que na realidade é o elemento terminal quem deve seguir uma trajetória, portanto de alguma maneira deve-se resolver necessariamente o modelo cinemático inverso da estrutura; este é um agravante muito forte, porque resolver as equações inversas envolvidas requer um tempo computacional bastante expressivo, mesmo para robôs com geometria considerada simples **[2][14][84][85]**. Aqui consideramos sempre o argumento para controle em tempo-real, porque, evidentemente estes cálculos podem ser realizados previamente (*OFF-LINE*), e amostrados depois durante a etapa de rastreamento **[10][32][33]**. Mas, neste caso, existe o problema que o sistema de controle atuará mal na presença de perturbações, e perturbações ocorrem sempre, porque existem parâmetros do robô que variam de acordo com a posição e a velocidade das juntas.

A maioria das aplicações destes métodos via espaço de juntas mostra que pode ser possível planejar a trajetória próximo do tempo-real, porque é possível sistematizar o procedimento de encontrar os polinômios adequados para os movimentos em função das restrições de velocidades e acelerações no início e final de cada subsegmento **[27]**.

Consideremos o problema de mover o elemento terminal de uma posição inicial até uma posição desejada num certo acréscimo de tempo. Utilizando a cinemática inversa, pode-se calcular o conjunto de ângulos de junta que corresponde à posição e orientação desejadas. A posição inicial é representada na forma de um conjunto de ângulos de junta. O que se deseja encontrar é uma função para cada junta, cujo valor em Ti (tempo inicial) é a posição inicial da junta, e cujo valor em Tf (tempo final) é a posição onde a junta deve ser posicionada. Existem muitas funções suaves que podem ser usadas para interpolar o valor da junta.

Fazendo um simples movimento suave, no mínimo quatro restrições são evidentes. Duas restrições nos valores das funções vêm da seleção dos valores inicial e final. Em adição, duas restrições mais são derivadas da imposição para que esta função seja contínua em velocidade. Estas quatro restrições podem ser satisfeitas por um polinômio de no mínimo terceira ordem. E como um polinômio cúbico tem quatro coeficientes, pode-se calcular seus valores adequados para satisfazer a todas elas, sendo que cada conjunto destas restrições serve para a especificação de um polinômio cúbico particular. Isso é válido para quando os movimentos têm uma determinada duração e um ponto final a ser atingido (meta). Em geral, queremos que a trajetória esteja em concordância com outros pontos intermediários especificados.

Se o robô deve rastrear outros pontos intermediários, então também pode-se encontrar soluções utilizando os polinômios cúbicos. Usualmente, quer-se realizar a passagem por estes pontos intermediários sem paradas, e neste caso, há a necessidade de generalizar o caminho no qual convenientemente escolhemos os polinômios cúbicos para as restrições da trajetória. No caso de um único ponto meta, as velocidades inicial e final do movimento são zero. Porém, para a passagem por outros pontos intermediários, deve-se converter um conjunto de ângulos de juntas desejados, por aplicação do modelo cinemático inverso, depois considerar-se o problema do cálculo dos polinômios que conectam os valores destes pontos intermediários para cada junta, juntamente com um caminho suave. Se existirem restrições devido a requisições de determinadas velocidades de juntas nos pontos

intermediários, então pode-se determinar os polinômios conforme comentado, mas considerando como restrições as velocidades de passagem exigidas.

Pode-se calcular um polinômio cúbico que conecta quaisquer posições inicial e final com quaisquer velocidades inicial e final. Existem diversos caminhos para especificar as velocidades desejadas nos pontos intermediários, como por exemplo, especificá-las em termos de velocidades cartesianas lineares e angulares do elemento terminal para cada ponto desses, ou fazer com que o sistema de geração encontre estas velocidades com aplicação de algum critério heurístico no espaço cartesiano ou no espaço de juntas, ou ainda que o sistema de geração encontre estas velocidades para causar acelerações contínuas nestes pontos. As velocidades cartesianas nos pontos intermediários podem ser encontradas com auxílio do Jacobiano Inverso do robô, avaliado em cada ponto intermediário. E quando o robô estiver em um ponto de singularidade para um particular ponto desses, então não existe a liberdade para a escolha de uma velocidade arbitrária.

Outro procedimento possível de realizar é primeiramente dividir a trajetória a ser rastreada em um número fixo de segmentos, então para cada segmento, transformar a configuração cartesiana de cada fim, e determinar começo e final de cada posição angular de junta. Depois determinar o tempo requerido para atravessar cada segmento, dividir este tempo em intervalos iguais de acordo com a freqüência de amostragem a ser utilizada pelo sistema de controle, e para cada junta determinar a distância angular a ser percorrida durante cada intervalo de tempo obtido. Então uma vez iniciado o movimento, no n-ésimo tempo de amostragem, o servocontrolador da i-ésima junta deve receber o nível de ajuste final determinado para o segmento em questão.

No caso em que a trajetória só exige a aproximação a determinados pontos, e não a passagem obrigatória por eles, o movimento entre duas posições pode ser executado com velocidade constante. Sem dúvida, isto pode ocasionar descontinuidades na velocidade de passagem de um ponto para outro. Mas também para este caso é possível encontrar soluções através da utilização calculada de movimentos acelerados-constantes-acelerados entre as transições, com tempos de acelerações fixos.

Os cálculos computacionais são executados com base nas funções de trajetória que devem ser atualizadas a cada intervalo de amostragem, e quatro pontos devem ser observados: Primeiro, o conjunto de níveis de ajustes para estas trajetórias devem ser muito bem calculado em modos não iterativos: Segundo, as posições intermediárias entre pontos de passagem exigidos pela tarefa devem ser encontradas e especificadas deterministicamente: Terceiro, a continuidade das posições e suas duas primeiras derivadas devem ser suaves: Quarto, deve-se minimizar as ocorrências de movimentos oscilatórios ao máximo entre intervalos consecutivos de rastreamento para não intensificar os efeitos dinâmicos de acoplamento na estrutura em movimento, que podem tornar o método falho.

Como pode-se ver, existe a necessidade de resolver-se a equação cinemática inversa para cada ponto de passagem exigido pela trajetória imposta, mas entre estes pontos de passagem, o problema é resolvido a nível de juntas. Se uma trajetória de junta usa 3 polinômios (3p), então são necessários 3(p+1) coeficientes para especificar as condições iniciais e finais dos segmentos (em posição, velocidade e aceleração), podendo-se então garantir a continuidade destas variáveis nos pontos extremos de cada um; métodos que utilizam dois polinômios quádricos e um cúbico (4-3-4), dois cúbicos e um quíntuplo (3-5-3), ou cinco cúbicos (3-3-3-3-3), podem também dar solução eficazes a este tipo de

problema [27][43].

Existe uma série de outras variações destes métodos adaptados para rastreamentos de trajetórias específicas, como: Segmentos de retas, de circunferências, etc.., incluindo a solução do problema de continuidade das variáveis nos pontos de transições entre segmentos, com interpolação de outras curvas intermediárias, como quadráticas, espirais, clotóides, etc.[10][33][44][45][51]. Uma deficiência importante nestes métodos é que eles exigem que os controladores de juntas garantam robustez às perturbações, isto é, estes controladores devem atuar de forma a que cada junta persiga exatamente os níveis de ajustes impostos a elas compensando as perturbações que atuam sobre elas, fato que complica enormemente suas utilizações onde as exigências para rastreamento das trajetórias são mais severas, ou onde estas perturbações são de maior intensidade.

Os métodos de solução do problema via espaço cartesiano geralmente propõem fazer o planejamento e o rastreamento das trajetórias em dois passos [17][61]. Primeiro, geração ou seleção do conjunto de pontos ou interpolação de pontos em coordenadas cartesianas de acordo com alguma lei ao longo desta trajetória, e segundo, uma especificação da classe de funções para unir estes pontos de acordo com algum critério. Os cálculos de geração e otimização das trajetórias que geram os sinais de controle para os atuadores de juntas são baseados no erro entre a posição cartesiana atual do robô e cada posição objetivo imposta pela trajetória, e portanto de alguma forma supõe-se que seja realizada uma correspondente conversão da solução encontrada para o espaço de juntas [44], isto é, há necessidade de encontrar de alguma forma as soluções dos modelos cinemáticos inversos para cada ponto de solução, e isso, como já foi citado, exije intensa carga computacional, porisso também existem soluções propostas que fazem integrações entre as técnicas via espaço cartesiano e via espaço de juntas, para otimizar ao máximo as necessidades de cálculos, visando aplicação em tempo-real [15]. A abordagem via espaço cartesiano é mais adequada que a via espaço de juntas para aquelas aplicações onde os eixos de movimentos de juntas não são ortogonais, e não se pode subdividir os problemas de posicionamento e orientação, dificultando a obtenção de funções de interpolação a nível de juntas para resolver o problema de orientação do elemento terminal [37]. No espaço cartesiano, a obtenção destas soluções é feita naturalmente, apenas considerando as matrizes de transformações homogêneas que contém as transformações de translação e rotação elementares entre as juntas [26][27][67].

As soluções encontradas via utilização exclusiva dos modelos cinemáticos, permitem soluções razoáveis desde que os efeitos de perturbações produzidos pela dinâmica não-linear dos manipuladores não sejam muito significativos [60][61], o que pode-se conseguir com um projeto mecânico adequado. Para os casos onde os efeitos dinâmicos não podem ser desprezados, o problema fica muito mais complicado, e é principalmente por esta razão que os maiores esforços dedicados atualmente na busca de soluções do problema de controle de trajetórias estão voltados a propor técnicas que considerem estes efeitos dinâmicos, a fim de que o sistema se comporte adequadamente [19][31][42][52][71][78][87]. Entretanto, se os modelos cinemáticos possuem bastante complexidade, os modelos dinâmicos são ainda muito mais complexos, e proporcionalmente necessita-se maior capacidade computacional para solucioná-los principalmente em tempo-real [39].

A implementação de algoritmos de controle linear é atrativa por causa do

comparativo esforço computacional **[58]**, entretanto devido às grandes não-linearidades dos sistemas robóticos somente pode-se utilizar algoritmos de controle linear para áreas limitadas do volume de trabalho. Uma lei de controle linear multi-juntas pode ser projetada para conseguir-se o desvio mínimo quadrático da trajetória desejada durante o seu rastreamento, e no caso de um ponto fixo a ser buscado um controlador assim pode ter parâmetros constantes **[40]**. No caso de existirem acoplamentos fortes no espaço de trabalho pode-se tentar projetar primeiramente um comparador dinâmico para desacoplamento linear, e baseados nisso desenvolver uma lei de controle ótimo para juntas individuais, por exemplo, com o método de alocação de pólos. Se o sistema tiver ordem muito elevada pode-se tentar fazer sua decomposição em partes através dos métodos de resposta em freqüência e baseados nestes resultados projetar controladores lineares organizados em níveis hierárquicos **[30]**.

Pode-se utilizar técnicas de controle por chaveamento no espaço juntas para melhorar o desempenho dos servocontroladores, por exemplo, a modulação por largura de pulsos (*PWM*) **[9][13][49]**, a modulação por controle de fase (*PLL*) **[50]**, o controle *bang-bang* **[65]**, etc..., através da escolha apropriada de parâmetros de chaveamento, conseguindo-se estabilidade e obtendo linearizações do robô com quase-desacoplamento. Embora este conceito de controle seja relativamente simples, existem dificuldades de implementação para os servomecanismos atuarem em freqüências elevadas de chaveamento, que podem causar problemas na construção física do sistema **[48][71]**.

Outro caminho atraente para resolver os problemas das incertezas de modelagem e das variações de parâmetros, é aplicar o controle adaptativo. Neste caso existem duas abordagens principais, ou seja, o controle adaptativo com modelo de referência **[1][18][69]**, e o controle adaptativo self-tuning **[37][73]**. No primeiro, é reservado para cada junta um controlador de posição. E cada malha de controle de junta é descrita por um modelo linear simples, de ordem adequada, que é processado completamente paralelo ao processo. O processo e o modelo são conduzidos por uma determinada trajetória, em posição, velocidade e aceleração. Dependendo das diferenças entre as saídas do processo e do modelo, os parâmetros do controlador são mudados adaptando o sistema real para o desejado do modelo. No segundo, o robô é descrito por uma equação discreta. Pela aplicação de algum método de identificação *ON-LINE*, os parâmetros atuais do modelo são estimados recursivamente considerando perturbações. E os parâmetros ótimos para a lei de controle são determinados com base na atualização do modelo linear. Estas abordagens têm atrativos comparadas aos outros métodos apresentados, porém ainda não são convincentes pelas avaliações experimentais publicadas no passado. Aparentemente o alto esforço computacional exigido e as questões de incertezas de instabilidade são os fatores que limitam estas implementações **[40]**.

Também existem esforços para o desenvolvimento de sistemas de controle robusto e novos projetos mecatrônicos mais eficazes. Os controladores robustos têm estrutura relativamente simples que promove altas qualidades de performance, e toleram as incertezas de modelagem sem perdas significativas no desempenho. Neste sentido sua grande maioria refere-se à conhecidas leis de controle linear e não-linear. Num primeiro passo é derivada uma relação funcional entre as incertezas de modelagem e as perdas de performance ou a estabilidade, e depois estas perdas são minimizadas pela escolha adequada de parâmetros fixos, pela introdução de uma malha adicional de controle de estabilização **[40][58][72][73]**.

As técnicas de controle preditivo também podem ser aplicados para controlar

os robôs e conseguir-se boas performances. Neste caso, primeiramente utiliza-se um modelo dinâmico interno do robô para realizar a simulação *ON-LINE* da resposta do sistema, incluíndo perturbações. Então uma trajetória de referência faz a transição suave da saída atual do processo para a trajetória desejada com a escolha de um horizonte de previsão, depois uma lei de controle ótimo força a saída do processo de modo que ela permaneça tão próxima quanto possível da trajetória de referência **[40]**. Para isso, as necessidades de memória e de tempo de amostragem são muito moderadas, e hoje já existem essas técnicas implementadas para o controle de robôs industriais, tanto para o controle ponto-à-ponto, como para o rastreamento de trajetórias contínuas, ex. robôs KUKA.

Em muitos trabalhos publicados sobre este assunto, pode-se verificar que, se o problema não fosse o tempo necessário para a realização dos cálculos, poder-se-ia projetar sistemas de controle que fizessem os robôs terem desempenhos praticamente perfeitos na realização de tarefas de extrema complexidade, e mesmo sob fortes restrições. A tendência natural é propor-se métodos que consideram critérios de otimização **[59][68]** adequados a cada aplicação, como por exemplo: Critérios de tempo mínimo **[40][70]** e minimização/maximixação de gradientes de funções **[12][24][45][51][70][67]**, critérios de mínimo gasto de energia, utilização de técnicas de controle não-linear **[65][72][79][83] [86]**, e ainda, técnicas de inteligência artificial que visam tornar o sistema de controle com capacidade de "aprender" com base em dados obtidos através de movimentos realizados pela estrutura **[11][28][39][52]**.

No caso prático importante que uma trajetória desejada é fixada para ser rastreada muitas vezes, a aplicação de leis baseadas em algoritmos de aprendizagem está tornando-se mais e mais atrativa. Aqui, durante poucos testes com o processamento do controlador, por exemplo, um algoritmo baseado em leis de lógica nebulosa (*fuzzy*) **[41]**, ele pode melhorar seu próprio desempenho através das experiências passadas. Os controladores por aprendizagem, ao contrário dos controladores adaptativos, têm por premissa que devem ir tornando-se melhores, sucessivamente, de execução para execução, e de forma que as condições não variem drasticamente entre elas. Mas o problema é justamente este: O tempo disponível para a realização dos cálculos é muito limitado. Então, para utilização prática, passa-se a optar por técnicas sub-ótimas, relaxando a rigorosidade exigida para a solução ótima, mas atentando para soluções dentro de uma margem de erro limitada e aceitável **[4][5][6][34][35]**.

Existem outras técnicas que se afastam um pouco do esquema tradicional de geração e controle de rastreamento de trajetórias que propõem resolver o problema de geração e rastreamento de modo conjunto e em tempo-real, resolvendo as equações das trajetórias a serem seguidas num mesmo algoritmo de controle que envolve todo o sistema do robô. Por exemplo: a técnica *ATGS* (*Autonomous Trajectory Generating Servomechanism*) **[8][29][47][48][80][81][82]**. Pode-se adaptá-la para que resolva o problema de rastreamento com critérios de otimização para que a trajetória seja seguida estavelmente e com baixa taxa de erro. Uma característica particular desta técnica, é que seus algoritmos são muito simples se comparados com as técnicas anteriores, e porisso perfeitamente realizável em tempo-real **[49]**. As características principais são que existe a exigência do conhecimento das equações analíticas que descrevem a trajetória na forma fechada, o robô deve ser movido inicialmente para uma posição que faz parte da trajetória definida, e devem ser inseridos critérios de parada, porque a recursividade do método é infinita. O controle é realizado para que o robô procure estar sempre sobre a curva definida por tal trajetória. Esta estratégia funciona muito bem para movimentos realizados em planos, considerando o controle de velocidade **[29][47][82]**, em movimentos

no espaço, as soluções ficam mais complicadas, porque para a técnica funcionar bem necessita-se encontrar a direção do vetor tangente em cada instante de movimento; no espaço, como a priori quer-se determinar este vetor através de um único ponto, resulta que existem infinitos vetores tangentes que passam por ele, e portanto, necessita-se efetuar mais cálculos, para fazer aproximações etc.. Outra idéia que inclusive é parte integrante desta tese (ver **Capítulo III**), é fazer seccionamentos adequados no espaço a ser executada uma tarefa, para que o robô seja controlado por um método variante desta técnica, mantendo o que ela possui de eficiente para os movimentos no plano.

O estudo aprofundado desta técnica *ATGS* nos conduziu primeiramente a idéia de fazermos seccionamentos adequados nas trajetórias em sub-planos, para fazermos a transferência das suas potencialidades nas aplicações de trajetórias definidas no espaço tridimensional, e o refinamento desta idéia, traduzido para a realização de seccionamentos infinitesimais das trajetórias, juntamente com o estudo de técnicas de vanguarda da inteligência artificial, nos permitiu propor a técnica de busca heurística em tempo-real que estamos apresentando, e que pode resolver muitos dos problemas comentados neste capítulo, e de modo conjunto.

CAPÍTULO III

PLANEJADOR E GERENCIADOR DE TRAJETÓRIAS POR INTELIGÊNCIA ARTIFICIAL

"Duas Abordagens Estudadas Para o Robô JECAII"

3.1 ☞ INTELIGÊNCIA ARTIFICIAL E TRAJETÓRIAS DE ROBÔS

O objetivo principal quando se deseja controlar as trajetórias de robôs é sugerir métodos cujos procedimentos façam com que os erros de percursso durante os movimentos sejam os mínimos possíveis (dentro de uma precisão estipulada para cada determinada tarefa) mesmo que durante estes movimentos o robô sofra perturbações que forçam a desvios indesejados.

Como é simples concluir, esta tarefa não é nada fácil, principalmente quando os métodos sugeridos são baseados em técnicas de controle fixo, isto é, pré-estabelecido, e sem capacidade de adaptação. Esses métodos só funcionam bem quando se conhece exatamente tudo o que ocorrerá durante os movimentos do robô, e ainda que a técnica utilizada e a sua estrutura sejam suficientemente robustas. Basta que verifiquemos na literatura técnica sobre o assunto para constatarmos que a complexidade envolvida quando são utilizados métodos assim é muito grande (veja **Capítulo II**). Geralmente é necessário que se conheça com grande precisão os modelos matemáticos que descrevem a estrutura física do robô. E sabemos que por mais que os modelos sejam complexos e representem de forma quase perfeita estes robôs, sempre existem erros e aproximações feitas, principalmente para descrever as não-linearidades que são intrínsecas às estruturas. Além do mais, muitos parâmetros dos robôs são variantes no tempo, de forma que os controladores, ou os métodos de controle fixos podem deixar a desejar.

Posto isso, a idéia é propor métodos que permitam aos controladores tornarem os robôs capazes de se adaptarem a essas variações, e assim conseguir melhor desempenho ao realizarem tarefas. Também existem muitos trabalhos publicados que propõem métodos para controladores adaptativos, mas ocorre que são métodos ainda de

maior complexidade, e muitas vezes extremamente difíceis de serem implementados para que controlem sistemas robóticos reais.

A pergunta a fazer é: — Será que não se pode propor métodos que permitam controlar os robôs de forma simples, e com a mesma capacidade de adaptação?

Buscando responder a esta pergunta, e explicando a evolução da pesquisa que nos permitiu sugerir a técnica de controle que utiliza conceitos de inteligência artificial que estamos propondo, com uma argumentação para respondê-la, primeiramente estudamos os princípios de funcionamento da técnica de controle ótimo de trajetórias de robôs chamada *ATGS* **[29]**, que gerou parte do conhecimento apresentado na tese de Mestrado **[48]** por nós publicada. Ela possui procedimentos distintos das técnicas tradicionais de resolver este tipo de problema que propõem encontrar-se soluções baseadas diretamente no espaço de juntas ou no espaço cartesiano. A técnica *ATGS* faz uma fusão destes dois espaços em uma única malha de controle que procede a geração e o controle das trajetórias ao mesmo tempo, de forma fechada, e com a possibilidade de anexação de critérios para a recuperação automática do rastreamento em caso de desvios provocados por perturbações. A sua filosofia é muito simples se aplicada para resolver os referidos problemas, quando as trajetórias que compõem as tarefas são definidas em espaços planos. Através de experimentos que realizamos, ela demonstrou-se muito eficiente **[47][49][80][82]**. O seu fator mais atrativo é que ela permite a realização do controle do rastreamento das trajetórias em tempo-real, por necessitar de algoritmos simples e de fácil programação, e o controle simultâneo das posições e velocidades do elemento terminal dos robôs. A seguir apresentamos o seu formalismo matemático para a sua aplicação a problemas de controle de trajetórias definidas em superfícies planas.

ABORDAGEM POR MÉTODOS CLÁSSICOS DE CONTROLE ÓTIMO
ATGS DE DIMENSÃO DOIS

3.2 ☞ TÉCNICA *ATGS* PARA RASTREAR TRAJETÓRIAS NO PLANO

ATGS é uma técnica de controle ótimo, de dois níveis hierárquicos que baseia-se nas medidas dos sensores de posição das juntas do robô para determinar a posição atual do seu elemento terminal, em função desta, gera referências de velocidades para o nível de controle de juntas, forçando o elemento terminal a descrever a trajetória especificada. Para garantir robustez no rastreamento, ela contém a equação que descreve a dita trajetória, a velocidade tangencial para o movimento do elemento terminal, e um critério de recuperação de trajetória baseado em otimização de funcional dentro da malha de controle. A **figura 3.1** ilustra a malha hierárquica de controle global desta técnica.

Quando esta técnica foi lançada **[29]**, foi proposto que os controladores de juntas ficassem em malha aberta de posição e fechada de velocidade, e foram obtidos bons resultados que indicam que ela pode ser eficaz para muitas aplicações. Mas quando os efeitos dinâmicos provocam variações acentuadas nos parâmetros do robô, o controle torna-se menos eficiente.

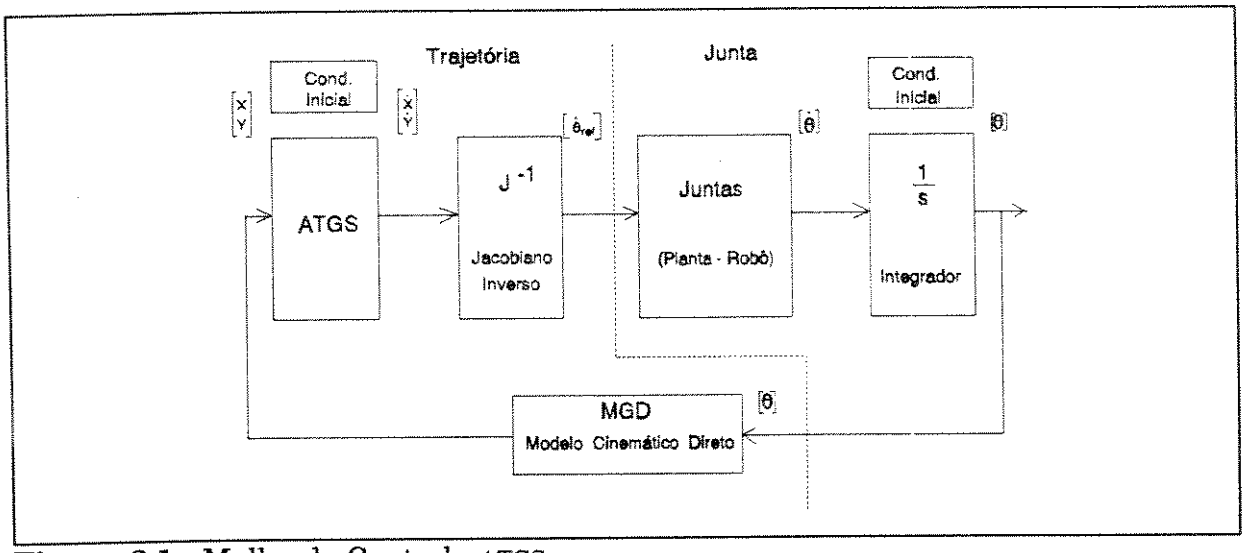


Figura 3.1 - Malha de Controle *ATGS*.

Para manter esta estrutura de controle robusta, mesmo quando o robô for submetido à perturbações maiores, inserimos os servomecanismos de controle de velocidade de juntas, com ação de controle PD e PI, e obtivemos resultados muito satisfatórios para realizar o controle de posição e velocidade de robôs em trajetórias planas **[47][49][80][82]**. Para a aplicação em planos, o modelo *ATGS* segue o processo: Primeiramente definimos a equação da trajetória a ser rastreada através de sua forma fechada, compreendida entre um ponto inicial (P_i) e um ponto final (P_f), veja a **figura 3.2.**

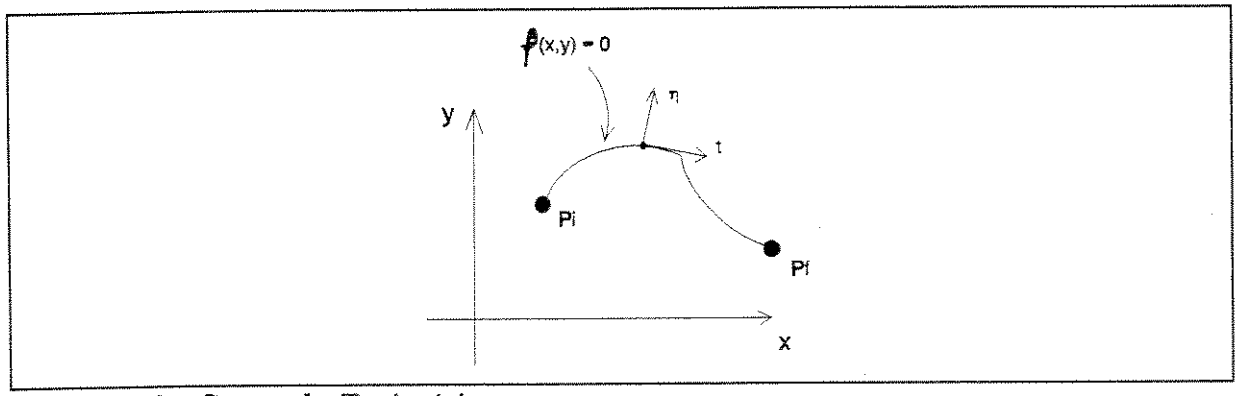


Figura 3.2 - Curva da Trajetória para *ATGS*.

Definimos dois eixos cartesianos cujas origens são dadas pela intersecção do eixo Z com o plano XY, e que forma o sistema de referência inercial do robô. O vetor η é o vetor normal à trajetória definida por f(x,y), e é dado por:

$$\eta = \frac{\nabla f}{|\nabla f|} \qquad \eta = \frac{\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]^T}{|\nabla f|} \rightarrow \eta = \frac{[fx, fy]^T}{|\nabla f|} \qquad \textbf{(3.1)}$$

$$\text{com } |\nabla f| = \sqrt{fx^2 + fy^2}$$

O vetor t é o vetor tangente à trajetória definida por f(x,y), e pode ser obtido através de uma rotação de -90° do vetor η. Ou seja:

$$\vec{r} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} . \nabla \quad \rightarrow \quad \vec{r} = \frac{[fy, fx]^T}{|\nabla f|} \tag{3.2}$$

Supondo que o robô avança sobre a trajetória na direção do vetor $\vec{r}$ com velocidade v, podemos escrever as seguintes equações diferenciais:

$$\begin{aligned} \frac{\partial x}{\partial t} &= v.\frac{fy}{|\nabla f|} \\ \frac{\partial y}{\partial t} &= -v.\frac{fx}{|\nabla f|} \end{aligned} \quad ou \quad \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v.\frac{fy}{|\nabla f|} \\ v.\frac{fx}{|\nabla f|} \end{bmatrix} \tag{3.3}$$

A solução dessas equações diferenciais pode ser obtida através do sistema mostrado na **figura 3.3.**

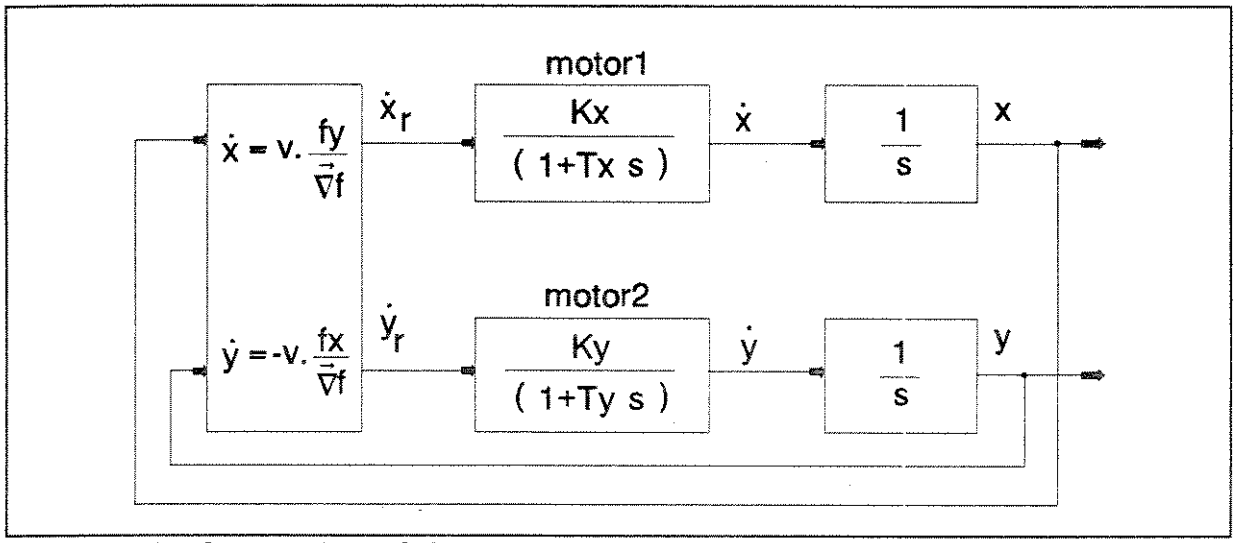


Figura 3.3 - Sistema Para Solucionar Equações Diferenciais *ATGS*.

Definindo um funcional que indica a amplitude do desvio da trajetória como:

$$J = \frac{f^2(x,y)}{2} \tag{3.4}$$

O controle deve atuar no sentido de minimizar este funcional.

O gradiente de J é dado por:

$$[-ffx \ , \ -ffy]^T = \left[\frac{\partial J}{\partial x} \ , \ \frac{\partial J}{\partial y}\right]^T \tag{3.5}$$

Então podemos definir o vetor velocidade de recuperação:

$$\vec{v}_r = \left[-\frac{K.ffx}{|\nabla f|^2} \ , \ -\frac{K.ffy}{|\nabla f|^2}\right] \quad , \quad com \quad K > 0 \tag{3.6}$$

Onde K é o ganho de recuperação.

Adicionando $\vec{v}_r$ na **equação 3.3** podemos obter a lei de controle com função de recuperação da trajetória:

$$\dot{x} = v.\frac{fy}{|\nabla f|} - \frac{K.ffx}{|\nabla f|^2}$$
$$\dot{y} = -v.\frac{fx}{|\nabla f|} - \frac{K.ffy}{|\nabla f|^2} \qquad \textbf{(3.7)}$$

A equação da trajetória é definida em função do tipo de movimento que se quer para o robô durante esta parte dela. Pode ser uma reta, ou um arco de circunferência, ou ainda outra equação mais sofisticada. E a troca do tipo de equação a utilizar pode ser interessante quando o movimento deve contornar obstáculos.

Como exemplo, para uma equação de reta, a lei de controle com função de recuperação pode ser equacionada assim:

$$f = v - ax - b \qquad \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = -\frac{1}{1+a^2}\cdot\begin{bmatrix}K(v-ax-b) & -v.\sqrt{1+a^2}\\ v.\sqrt{1+a^2} & K(v-ax-b)\end{bmatrix}\cdot\begin{bmatrix}-a\\ 1\end{bmatrix} \qquad \textbf{(3.8)}$$
$$\nabla f = [-a\ ,\ 1]^T$$

Considerando que o movimento no plano XY seguirá uma reta que contém o ponto $P_i = P(x_0,y_0)$ e o ponto $P_f = (x_1,y_1)$, ainda devemos determinar os coeficientes angular e linear desta reta, para substituirmos na **equação 3.8**.

Para que o *ATGS* atue de forma correta é necessário que o elemento terminal seja colocado em P_i. Através do modelo cinemático inverso pode-se determinar os ângulos correspondentes em que cada junta do robô deve ser posicionada, então deve-se adicionar um algoritmo inicial a esta técnica de controle, que faz o primeiro posicionamento do elemento terminal nesta posição. Ao iniciar o movimento o elemento terminal está com velocidade tangencial zero, e como foi especificada uma velocidade tangencial (v), deve haver uma aceleração durante os instantes iniciais. Optamos por controlar esta aceleração inicial para que o movimento seja mais suave. Fornecemos referências de velocidades de tal forma que o elemento terminal é movido com aceleração constante. Podemos, por exemplo, definir que a velocidade atingirá seu valor nominal no 8º período de amostragem a partir do início do movimento. Assim o valor de cada incremento é de $v/8$ (observe a **figura 3.4**).

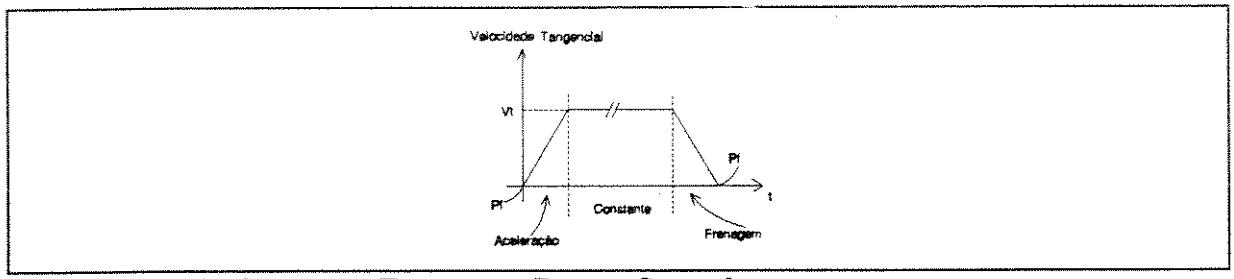


Figura 3.4 - Aceleração e Frenagem Para o Controle por *ATGS*.

Da mesma forma, para finalizar o movimento necessita-se um procedimento de desaceleração. Utilizamos como critério a distância que o elemento terminal se encontra do ponto final P_f. Quando esta distância for menor que um padrão, forçamos um movimento com desaceleração constante até que ele pare. Este processo de desaceleração segue os seguintes passos:

- Verifica-se quanto vale a velocidade tangencial.
- Determina-se qual é o tempo mínimo necessário para que fazer o robô parar,levando-se em consideração a máxima desaceleração possível.
- Determina-se o número de iterações necessárias para parar.
- Gera-se as referências de velocidade para cada iteração até o elemento terminal atingir velocidade zero.

A máxima velocidade tangencial possível é função da capacidade dada pela construção física do robô, e da capacidade computacional do *hardware* do sistema de controle utilizado para processar a malha *ATGS*.

Suponhamos como exemplo, que o período da malha *ATGS* seja de 30ms, e que é requisitada uma velocidade tangencial de 3cm/s para o elemento terminal. Entre duas amostragens sucessivas o elemento terminal poderá percorrer 0,9mm em direção não correta devido a uma eventual perturbação, porisso que dependendo da tarefa a ser executada, esta possibilidade de erro pode afetar o desempenho desta técnica.

Prosseguimos estudando e tentando sugerir uma técnica variante dela para poder aplicá-la aos problemas definidos no espaço tridimensional, com a intenção de manter as suas características de eficiência nos espaços planos. Como foi visto, para que ela funcione corretamente, existe a exigência do conhecimento das equações analíticas que descrevem a trajetória na forma fechada. Isso provoca a necessidade de levar-se o robô inicialmente para uma posição que faça parte da trajetória definida, e exige a inserção de critérios de parada, porque a recursividade do método é infinita.

O controle é realizado para que o robô procure estar sempre sobre a curva definida por tal trajetória. Para movimentos no espaço, as soluções ficam mais complicadas, porque para que ela funcione bem é necessário encontrar a direção do vetor tangente em cada instante do movimento, e no espaço (dimensão três), como a priori quer-se determinar este vetor através de um único ponto, resulta que existem infinitos vetores tangentes que passam por ele, portanto, necessita-se, no mínimo, realizar-se mais cálculos para fazer aproximações etc... A idéia que surgiu então, foi a de fazer seccionamentos em planos no espaço de trabalho do robô, adequando-os para que contenham todos os intervalos que formam as trajetórias a serem rastreadas na execução das tarefas.

3.3 ☞ TÉCNICA PARA RASTREAR TRAJETÓRIAS NO ESPAÇO 3D COM SECCIONAMENTOS PLANARES DO VOLUME DE TRABALHO

O estágio atual das pesquisas ainda não nos permitiu fazer uma generalização desta técnica para qualquer tipo de robô, porisso fazemos esta apresentação fixando-nos na solução de problemas para estrutura semelhantes a que possui o robô **JECAII**.

Podemos definir um vetor P, relativo à origem do sistema de coordenadas (X_0,Y_0,Z_0), que vai até o ponto onde os eixos de movimento das últimas juntas (orientação) se interceptam. Ele corresponde ao vetor de posicionamento do robô; este vetor sempre é definido pelo posicionamento das suas três primeiras juntas. Então visamos a aplicação desta idéia para este sub-problema, que pode ser complementada para a solução completa de posicionamento e orientação, juntamente com uma proposta específica para o problema da orientação, se supuzermos que existe desacoplamento, ou ainda quem sabe por outra variante deste método a ser proposta.

Como o **JECAII** possui uma junta rotacional na base, que permite um movimento de cintura, fazendo analogia com o corpo humano, a solução para o movimento desta junta fica bastante simples. Podemos sempre imaginar que o movimento será ponto-à-ponto, então de um ponto para o outro, compor a trajetória no espaço; podemos facilmente calcular qual deve ser a projeção do ponto atual, e do ponto seguinte, ao plano XY. Para isso basta que ignoremos neste momento a componente Z destes pontos. Então podemos utilizar a lei dos cossenos para encontrar o ângulo (teta1) entre as retas formadas pela origem e estes pontos. Se definirmos estes pontos como $P1=(x_0,y_0)$ e $P2=(x_1,y_1)$ temos:

$$\theta_1 = \cos^{-1}\left[\frac{(x_0^2 + y_0^2 + x_1^2 + y_1^2) - ((x_1 - x_0)^2 + (y_1 - y_0)^2)}{2 \cdot \sqrt{x_0^2 + y_0^2} \cdot \sqrt{x_1^2 + y_1^2}}\right] \qquad \textbf{(3.9)}$$

Uma vez que a base tenha sido movida de acordo com o método anterior, o ponto P do robô estará contido no plano vertical que contém a origem e o ponto final (x_1,y_1) a ser atingido neste movimento. Chamamos este plano de Plano UV.

Assim a trajetória de posicionamento restante fica muito simplificada, e será executada pelos 2º e 3º graus de liberdade, cujos eixos de estrutura já estão no mesmo plano vertical citado. Então utilizamos o mesmo raciocínio apresentado anteriormente para o controle de trajetórias em superfícies planas com a técnica *ATGS*, apenas fazendo a mudança das variáveis x e y para u e v, respectivamente.

Observando o plano UV mostrado na **figura 3.5**, e que o movimento neste plano seguirá uma reta que contém o ponto $P(u_0,v_0) = P(x,y,z)$, e o ponto $Pf = (u_1,v_1)$, devemos determinar os coeficientes angular e linear desta reta para substituirmos na **equação 3.8** adaptada para as variáveis u e v. Então:

$$a = \frac{Z_1 - Z_0'}{\sqrt{x_1^2 + y_1^2} - \sqrt{x_0'^2 + y_0'^2}}$$

$$b = 0$$

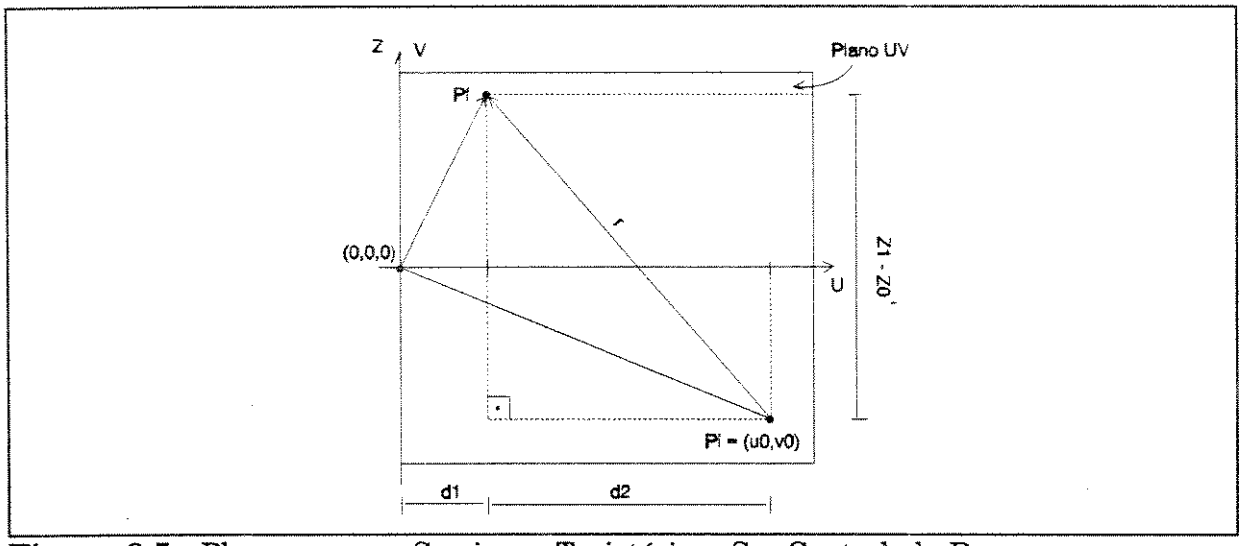


Figura 3.5 - Plano UV para Seccionar Trajetória a Ser Controlada Por *ATGS*.

Pela **figura 3.5** pode-se verificar que:

$$\Delta u = d1 - d2$$

$$\Delta u = \sqrt{u_1^2 + u_1'^2} - \sqrt{u_0^2 + u_0'^2}$$

$$\Delta v = Z_1 - Z_0'$$

Este procedimento pode funcionar muito bem para fazer o robô rastrear uma trajetória ponto-à-ponto. Primeiramente posicionamos seu elemento terminal sobre o ponto inicial P_i, então giramos sua cintura até que o seu elemento terminal atinja um ponto que está contido no plano vertical que contém o ponto de origem do seu sistema de coordenadas e o ponto final P_f, e a partir deste ponto aplicamos a técnica *ATGS* para fazer o elemento terminal rastrear a reta que passa por ele e pelo ponto P_f contida neste plano vertical.

Pode-se notar que há um ponto de descontinuidade na curva de velocidade justamente no momento que o elemento terminal atinge o plano vertical citado e inicia-se o controle por *ATGS*. Esta característica pode não ser interessante para muitas aplicações, sendo que a descontinuidade fica mais acentuada proporcionalmente à amplitude dos movimentos da cintura do robô entre o ponto Pi e o ponto intermediário que está contido no plano vertical. Levando-se em consideração este fato, procuramos fazer um seccionamento melhor da trajetória no volume de trabalho necessário para rastrea-la. Concluímos que deveríamos produzir um algoritmo que particionasse a trajetória em tramos pequenos, que no limite tendessem a zero. Ou seja, que fizesse o robô ser movido por passos infinitesimais. O estudo aprofundado da técnica *ATGS* nos conduziu primeiramente a idéia de fazermos esses seccionamentos adequados nas trajetórias em sub-planos, para conseguirmos realizar a transferência das suas potencialidades para aplicações no rastreamento de trajetórias definidas no espaço tridimensional, e o refinamento desta idéia, traduzido para a realização de seccionamentos infinitesimais das trajetórias, juntamente com o estudo de técnicas de vanguarda da inteligência artificial nos permitiu propor a técnica de busca heurística em tempo-real que estamos apresentando, e que pode resolver muitos dos problemas citados de modo conjunto.

3.4 ☞ TÉCNICA DE BUSCA HEURÍSTICA PARA O PLANEJAMENTO E CONTROLE DE TRAJETÓRIAS DE ROBÔS NO ESPAÇO 3D.

3.4.1 ☞ Inteligência Artificial, Reflexões e filosofia

Verificando alguns trabalhos recentes que propõem métodos que permitem que os robôs "aprendam" a partir das ocorrências anteriores em seus movimentos, pode-se constatar, que são aparentemente muito simples, se comparados com os que versam sobre idéias, conforme apresentamos no **Capítulo II**. Estes métodos são colocados na literatura em uma linha de pesquisa chamada Controle por Aprendizagem (*Learning Control*) **[28] [52]**. Existe uma linha de pesquisa onde são propostos métodos que não necessitam grande parte da modelagem matemática dos robôs para funcionarem, e que através de procedimentos iterativos são capazes de ir melhorando o desempenho do robô ao realizar uma tarefa, a cada vez que ela é realizada da mesma forma novamente **[4][5][6][34][35]**, isto é, o sistema de controle vai aprendendo a realizá-la melhor com o passar do tempo. É uma espécie de análise que vai sendo feita, em função de cada sinal de controle injetado no processo, e de sua respectiva resposta. A planta do robô é vista como uma caixa preta, e a análise é feita em função dos erros entre a trajetória desejada e a trajetória executada em cada período de amostragem. As equações matemáticas a serem resolvidas durante a atividade de controle são bastante simples, e podem permitir o controle em tempo-real, mesmo que o sistema de controle não tenha grande capacidade computacional.

Entretanto existem dois problemas fundamentais nestes métodos, um é que eles só se tornam eficientes para tarefas repetitivas, e outro que dependendo da tarefa, ou trajetória desejada para o robô, ele pode demorar muito tempo para aprender, ou até mesmo não conseguir aprender (os métodos podem não convergir para certos casos). Estão também propostas algumas estruturas auxiliares de controle para que estes métodos sejam mais eficazes, como utilizar realimentações com controladores tipo proporcional + integral + derivativo de estrutura variável **[11]**.

Como visto estes métodos são uma tentativa de se colocar "inteligência" nos sistemas de controle dos robôs, uma vez que são métodos que tentam permitir que os robôs "aprendam" a partir de experiências passadas. Atualmente muitos estudos em robótica, com abordagens por inteligência artificial, estão voltados para o nível de controle direto de atuação sobre o robô, mas acreditamos que poder-se-á obter aplicações interessantes para os níveis de supervisão desses sistemas, onde por exemplo, eles fossem capazes de se "organizarem", avaliarem a aprendizagem conseguida, e utilizá-la para executarem outras tarefas distintas daquelas que já realizaram. Por exemplo, quando pensamos em um pessoa aprendendo, e agindo como no ato de escrever, uma vez que aprenda a escrever seu nome, ela o saberá fazer novamente mesmo que com um lápis ou uma caneta diferente daquela que aprendeu pela primeira vez, e mesmo que o papel ou

a superfície onde estiver escrevendo sejam diferentes. Muitas características podem ser diferentes (peso, atritos, formas, etc...) mas mesmo assim esta pessoa o fará. É claro que poderão acontecer pequenos desvios na forma de escrever cada letra, mas provavelmente não inutilizarão a tarefa.

É exatamente essa idéia que nos faz pensar que seja importante sugerir métodos de aprendizagem a um nível mais elevado do que aquele que só trata exclusivamente do problema dos servomecanismos, sem é claro, descartar esta possibilidade, porque acreditamos que ela certamente deverá fazer parte da estrutura global de métodos de controle inteligente de trajetórias e tarefas.

Fazendo uma reflexão de como nós mesmos agimos quando vamos fazer algum movimento, por exemplo, quando queremos deslocar um objeto de um lugar para outro, segundo uma determinada trajetória, tentamos abstrair as ações básicas que fazemos para tentar transferí-las para um método automático que porventura uma máquina também possa fazer. Acreditamos que a princípio há um breve instante onde fazemos uma análise rápida de como começar a movimentar-nos (certamente em direção ao lugar onde está este objeto); é claro que temos o sistema da visão que nos auxilia muito nessa tarefa, mas por hora, vale acreditar que conhecemos as distâncias através de medidas matemáticas. Também sabemos o lugar e a posição onde nos encontramos. Podemos facilmente concluir que temos uma direção "melhor" para seguir; logo após nos colocamos em movimento e ficamos atentos a tudo que está ocorrendo no espaço que precisamos utilizar. Se imaginarmos que é possível executar toda a tarefa somente utilizando um braço (certamente com um movimento de cintura também), a reflexão que estamos fazendo fica mais simples. Imaginemos que por algum motivo devemos percorrer esta distância até o objeto, segundo uma reta (restrições do espaço). Acreditamos que nossos atos podem ser vistos como atos que produzem movimentos diferenciais da nossa estrutura óssea, e que existe realimentação de informações com nosso cérebro em cada movimento diferencial, que por sua vez, fica controlando se os movimentos diferenciais realizados foram satisfatórios de acordo com o que a tarefa está exigindo. Como o movimento foi muito pequeno, se por um acaso ele foi totalmente errado, no movimento diferencial seguinte existirá uma ação no sentido de corrigí-lo da melhor forma possível, (por exemplo, com um movimento "um pouco mais" rápido, ou "mais lento" em alguma das juntas do braço para compensar).

É importante notar que aparentemente movemos todas as juntas ao mesmo tempo, ainda que uma delas possa estar se movendo com uma velocidade maior que as demais, etc... Mas vamos supor que isso só seja possível depois que já tenhamos "aprendido" como movimentar nossas juntas conjuntamente. E por um processo assim vamos nos movendo até chegarmos ao ponto onde está o objeto que queremos deslocar. Durante este movimento podem ocorrer batidas ocasionais em outros objetos do ambiente, ou outros tipos de ações que podem perturbar os movimentos. Acreditamos que ao fazermos pela primeira vez um movimento, cometemos vários erros, que ao nos prepararmos pensando que vamos ter que fazê-los de modo semelhantes mais tarde, guardamos algumas informações na memória (tanto informações sobre os momentos em que foram realizados os movimentos "bons" como os movimentos "ruins"); o importante é que estas informações guardadas, muitas vezes, não necessitam de dados temporais, e atentamos "fortemente" para àquelas situações onde ocorreram "mais erros", porque se tivermos que repetir, quando surgir uma situação parecida, saberemos que a estrutura deverá estar mais preparada (quem sabe realizar ali movimentos mais lentos, ou tensionar mais os músculos gastando mais energia, etc...).

Para nós, estas situações onde ocorreram "mais erros" são um indício forte de que para resolvê-las temos que aprender mais, ou reavaliar o que temos aprendido (claramente uma atitude de supervisão geral) para não cometer "tanto" erro assim.

Agora imaginemos que esquecemos completamente como nosso braço funciona ("por alguma amnésia repentina"). E temos que realizar a mesma tarefa citada anteriormente. Como fazer?

Poderíamos tentar começar a movimentar apenas a junta do ante-braço, colocando-a em movimento por um tempo muito pequeno. Em seguida verificamos se o movimento foi "bom" ou "ruim". Se foi "bom" OK! Vamos adiante por mais um movimento pequeno, se não foi, podemos movimentar da mesma forma a junta do braço propriamente dito, e fazemos uma análise semelhante (guardando de alguma forma estas informações). Podemos também pensar em criar, de modo parecido, uma seqüência de movimentos, onde por um tempo também muito pequeno, moveríamos seqüencialmente todas as juntas que temos no braço, e ao final de uma seqüência dessas fazer uma análise entre o erro que surgiu, e o erro que cada junta provocou, de forma a taxar com uma espécie de ponderação cada uma delas dentro da seqüência.

Isso nos permitiria ter uma idéia de como fazer o movimento (seqüência) seguinte, movimentando de forma proporcional à respectiva ponderação cada uma das juntas. Como a trajetória é uma função no espaço, encontrar estas ponderações não é fácil, uma vez que o movimento deve acompanhá-la, e vai sendo sempre em função dos erros causados em relação a esta trajetória.

Com esta idéia poderíamos aprender de forma razoável, sempre que temos que nos mover segundo funções que são semelhantes a que tentamos descrever, mesmo para uma tarefa diferente a ser executada em outra ocasião, armazenando uma espécie de melhor seqüência a realizar. A chave é encontrar qual seqüência melhor é essa. Aí entra a idéia de criar um ordenamento das informações de cada seqüência no sentido de se conseguir tirar conclusões, por exemplo, que para um determinado movimento elementar é melhor mover um par de juntas ao mesmo tempo, ou uma trinca, etc..., ao invés de uma só por vez. Assim estaremos trabalhando em um nível de organização estrutural da inteligência. Pode ser, entretanto, que nem se necessitem tais procedimentos se as determinadas tarefas não o exigirem.

A área de estudos sobre inteligência artificial tem sofrido muitos avanços nos últimos anos, principalmente dado ao seu grande potencial mostrado para resolver muitos problemas com soluções inovadoras e muito eficientes do ponto de vista computacional e econômico. Existem técnicas propostas por várias de suas sub-áreas que podem ser aplicadas em robótica; como: técnicas de aproximação estocástica, técnicas sintéticas por indução inferencial (empírica, abdutiva, construtiva, e através de exemplos), técnicas analíticas por dedução inferencial (inferências estatísticas, algoritmos ergódicos), técnicas de aprendizagem por modelos autômatos, técnicas de treinamento por métodos de prêmio/castigo através de instruções e experimentos, técnicas de aquisição de conhecimento por analogia e observação, técnicas de aprendizagem neuronal, e técnicas por programação heurística.

Tendo em mente a reflexão que apresentamos, e utilizando conhecimentos destas técnicas que acabamos de citar, principalmente dos métodos de programação heurística, fizemos um desenvolvimento formal que pode ser aplicado ao controle de robôs.

3.4.2 ☞ Definições e Argumentação Matemática

Emitimos uma hipótese $h_i(t)$ entre todas as possíveis: movimento de uma junta em um determinado sentido (e uma determinada quantidade), medimos a distância (d) do elemento terminal do robô ao ponto objetivo, seja com utilização da equação cinemática direta ou com algum processo de visão, dentre as $2n$ medidas (n é o número de graus de liberdade do robô) elegemos aquela em que d é a menor, e fazemos com que o respectivo movimento seja realizado; se trata de um método indutivo, já que só podemos concluir que h_i é a que mais satisfaz o objetivo entre todas as possibilidades. Portanto podemos definir h como:

$$h = \{ (mov_1, sentido^+), (mov_1, sentido^-), (mov_2, sentido^+), (mov_2, sentido^-), \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (mov_n, sentido^+), (mov_n, sentido^-) \}$$

Mas será que abordando o problema desta maneira, sempre existe pelo menos um movimento que faça com que a distância d diminua até anular-se, ou então que seja menor que uma precisão ε?

Vamos supor que X_P seja o ponto onde está situado o elemento terminal do robô, e que X_Q é o próximo ponto a ser atingido. Suponhamos que as suas juntas são rotativas. Então, posicionando-nos no espaço de qualquer uma de suas juntas (i-ésima junta), podemos abordar a situação de forma geométrica, conforme está mostrado na **figura 3.6**.

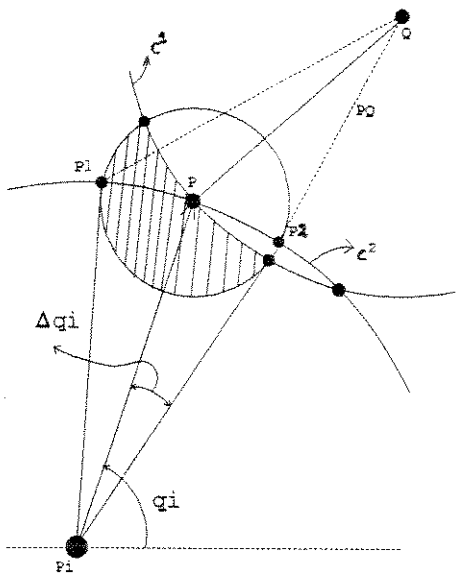


Figura 3.6 - Geometria Para Movimento Elementar i.

Para facilitar o entendimento, a **figura 3.6** mostra uma projeção dos pontos X_P (P) e X_Q (Q) num plano ortogonal ao eixo do movimento da i-ésima junta.

C^1 é uma circunferência de centro em Q e de raio $|P-Q|$, e C^2 é uma circunferência centrada no ponto de projeção da intersecção do eixo de movimento da junta i com o plano perpendicular ao seu movimento e de raio $|P-P_i|$. Assim, estas circunferências C^1 e C^2 sempre se interceptam. E existem dois casos, isto é, podem ser secantes (se interceptam em P e outro) ou tangentes (se interceptam em P). Quando elas forem secantes, se executarmos um movimento Δq_i no sentido horário, posicionaremos o elemento terminal do robô em um novo ponto X_{P_k}, cuja projeção no mesmo plano (P_k) está contida na circunferência C^2. Da mesma maneira, se executarmos um movimento de mesma amplitude no sentido anti-horário, o elemento terminal do robô será posicionado em um ponto X_{P_l}. E sempre somente um desses dois movimentos causará uma diminuição da distância entre o elemento terminal e o ponto objetivo X_Q, e o outro um aumento. Quando elas forem tangentes, qualquer movimento da junta i, causará um aumento na distância $d(|X_Q - X_P|)$, e portanto deverá ser um movimento descartado; esse fato, a priori, poderia causar problemas para o bom funcionamento da técnica, porque como se constata, não haveria um movimento para a junta i que pudesse diminuir d.

Como o robô tem outras juntas, o procedimento normal seria verificar todas as possibilidades e encontrar um movimento de outra junta que satisfaça este objetivo. Note que há situações de total singularidade, quando a estrutura do robô está completamente alinhada e o ponto X_Q pertence à reta definida por este alinhamento. Neste caso também existem duas situações, ou o ponto X_Q está fora do volume de trabalho do robô (caso que não iremos considerar) ou está dentro. Como esta situação singular só ocorrerá quando o ponto X_P estiver sobre o limite do volume de trabalho, se o ponto X_Q estiver dentro, basta que provoquemos um pequeno movimento em qualquer uma das juntas para que o processo prossiga normalmente. Fora este caso particular, sempre existe solução, conforme demonstramos abaixo:

Definimos ϕ como a seqüência de estados assumidos pelo robô ao rastrear uma determinada trajetória:

$$\phi = (\phi_1, \phi_2, \ldots, \phi_m)$$

onde:

$$\phi_i \in \{q_1^+, q_1^-, \ldots, q_n^+, q_n^-\},\ i = 1, \ldots, m$$

e

$$S = \{+,-\};\ s \in S;\ s \cup \bar{s} = S$$

$$N = \{1, \ldots, n\}$$

$$M = \{1, \ldots, m\}$$

Postulado 1:

$$Se\ \ \phi_j = q_k^s \ \Rightarrow\ \phi_{j+1} \neq q_k^{\bar{s}},\quad \begin{matrix} k \in N \\ j \in M \end{matrix}$$

Este postulado garante que uma ação não anule a ação anterior, fazendo um movimento com mesmo passo e de sinal oposto logo após ter havido uma ação sobre a mesma junta.

Postulado 2:

Como vimos, se as circunferências C^1 e C^2 são secantes, então $|X_Q - X_{Pk}|$ é menor que $|X_Q - X_{Pl}|$, ou vice-versa, dependendo da configuração ser *Cotovelo* acima ou *Cotovelo* abaixo. Vamos supor que o primeiro caso seja verdadeiro. Então:

$$Se \quad d_{P_i Q}^{j} < d_{P_l Q}^{j} \qquad d_{P_i Q}^{j+1} < d_{P_l Q}^{j+1} \qquad \textbf{(3.10)}$$

Desde que não se passe do ponto de secância.

Isso é importante, porque conclui-se que uma vez que esteja decidido o sentido de movimento que uma determinada junta deve ter para que seja atingido o próximo ponto objetivo, ela deverá ser movimentada sempre com este sentido, isso se for necessário mais de um movimento desta junta. E esta indução pode ser feita logo ao iniciar uma nova tentativa de aproximação a um próximo ponto objetivo que faz parte da trajetória a ser rastreada. Se durante um intervalo de movimento compreendido entre onde o robô está posicionado, e sua próxima posição objetivo, alguma das suas juntas mudar de sentido de movimento, é indício de que o mesmo movimento pode ser melhorado. Aqui é conveniente lembrar que o robô pode sofrer perturbações externas, e portanto, na prática, é perfeitamente possível que o algoritmo de controle tenha que compensar fazendo com que as juntas alterem seus sentidos de movimentos, e com esta idéia se poderia incrementar o mesmo algoritmo para que "aprenda" a promover movimentos cada vez melhores para fazer estas compensações necessárias durante perturbações externas.

Em geral não sabemos a relação entre $d_{P_i Q}^{j+1}$ e $d_{P_i Q}^{j}$, porque depende da configuração do robô, em cada instante, mas sabemos que existe; o fato é que determiná-la para cada movimento custaria uma carga computacional muito maior, o que prejudicaria muito o controle em tempo-real; por outro lado, conhecer esta relação poderia ser muito útil para melhorar os movimentos com respeito principalmente à velocidade. Abordamos esta questão mais adiante, por ocasião da determinação de passos sub-ótimos para os movimentos, e continuaremos fixando-nos no problema de posicionamento.

Postulado 3:

Dada $\phi = (\phi_1, \phi_2, ..., \phi_m)$ uma trajetória de $P_1 = X_P$ até $P_2 = X_Q$, se obteria a mesma solução cinemática para o robô em P_2 se fossem efetuados os movimentos das suas juntas com qualquer ordem destes ϕ_m estados, pois o resultado da somatória dos m movimentos de cada junta será o mesmo.

Ainda que a trajetória do elemento terminal descrita entre os respectivos pontos possa ser completamente distinta com respeito a diferentes seqüências desses estados, a somatória dos movimentos elementares de cada junta necessários para este movimento cartesiano é sempre igual, se for considerada, é claro, uma mesma configuração inicial e final do robô, independentemente do caminho ou percurso realizado para ir de P_1 a P_2. Deve-se lembrar no entanto que existem infinitos movimentos possíveis para ir de um ponto a outro do espaço, mas sempre haverá um melhor se for considerado algum critério de otimização para isso.

Então a idéia é encontrar o melhor movimento possível para cada junta, em cada instante de tempo; ou, mover as juntas de uma quantidade $\alpha = \Delta q_i$, i= 1,...n, baseando-se na configuração do robô no instante atual (*kT*), e na medida *d(kT)* (distância cartesiana atual entre o elemento terminal e o próximo ponto objetivo), tal que no instante de tempo *kT+1*, *d(kT+1)* atinja seu mínimo relativamente ao deslocamento $\pm\Delta q_i$. Como estamos interessados no melhor movimento para a *i*-ésima junta, fazemos uma projeção de todos os pontos de interesse para um plano ortogonal ao eixo de movimento desta junta (Z_{i-1}), conforme está mostrado na **figura 3.7**.

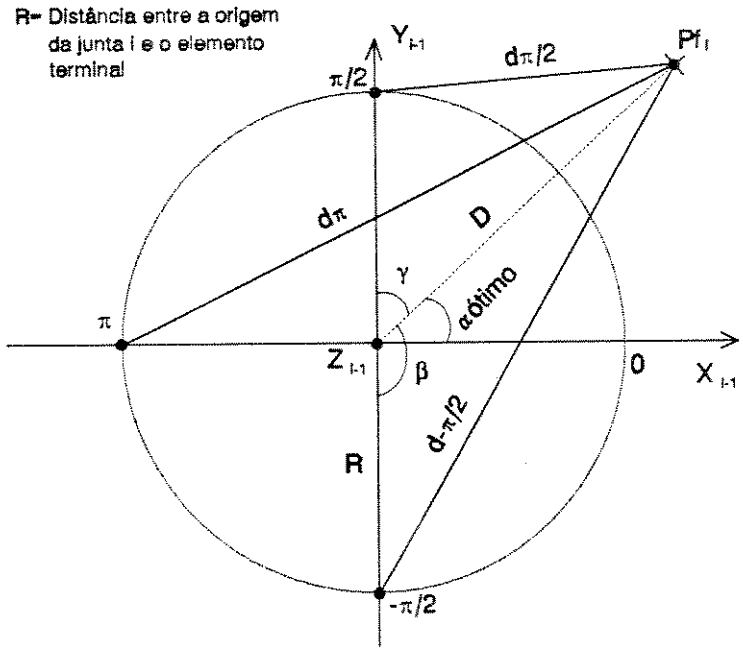


Figura 3.7 - Projeção de Distâncias Cartesianas no Plano Normal ao Eixo do Movimento da i-ésima Junta.

Com o auxílio da lei dos cossenos, e primeiramente considerando as projeções das distâncias, supondo movimentos da *i*-ésima junta até $\frac{\pi}{2}$ e até $-\frac{\pi}{2}$, podemos verificar que:

$$(d_{\frac{\pi}{2}})^2 = R^2 + D^2 - 2.R.D.\cos(\frac{\pi}{2} - \alpha^*)$$

$$(d_{-\frac{\pi}{2}})^2 = R^2 + D^2 - 2.R.D.\cos(\frac{\pi}{2} + \alpha^*)$$

então: $(d_{\frac{\pi}{2}})^2 + 2.R.D.\cos(\frac{\pi}{2} - \alpha^*) = (d_{-\frac{\pi}{2}})^2 + 2.R.D.\cos(\frac{\pi}{2} + \alpha^*)$

$$2.R.D.\cos(\frac{\pi}{2} - \alpha^*) - 2.R.D.\cos(\frac{\pi}{2} + \alpha^*) = (d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}})^2$$

$$2.R.D.(\cos(\frac{\pi}{2} - \alpha^*) - \cos(\frac{\pi}{2} - \alpha^*)) = (d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}})^2$$

mas $\cos(\frac{\pi}{2} - \alpha^*) - \cos(\frac{\pi}{2} + \alpha^*) = 2.sen(\alpha^*)$

logo:

$$4.R.D.sen(\alpha^*) = (d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}}) \qquad \textbf{(3.11)}$$

Supondo agora movimentos da junta *i*, até π, e até $\frac{\pi}{2}$, podemos verificar que:

$$(d_{\pi})^2 = R^2 + D^2 - 2.R.D.\cos(\pi - \alpha^*)$$

$$(d_{\frac{\pi}{2}})^2 = R^2 + D^2 - 2.R.D.\cos(\frac{\pi}{2} - \alpha^*)$$

então: $(d_{\pi})^2 + 2.R.D.\cos(\pi - \alpha^*) = (d_{\frac{\pi}{2}})^2 + 2.R.D.\cos(\frac{\pi}{2} - \alpha^*)$

$$2.R.D.\cos(\pi - \alpha^*) - 2.R.D.\cos(\frac{\pi}{2} - \alpha^*) = (d_{\frac{\pi}{2}})^2 - (d_{\pi})^2$$

$$2.R.D.(\cos(\pi - \alpha^*) - \cos(\frac{\pi}{2} - \alpha^*)) = (d_{\frac{\pi}{2}})^2 - (d_{\pi})^2$$

mas $\cos(\pi - \alpha^*) - \cos(\frac{\pi}{2} - \alpha^*) = -(\cos(\alpha^*) + sen(\alpha^*))$

logo:

$$2.R.D.(\cos(\alpha^*) + sen(\alpha^*)) = (d_{\pi})^2 - (d_{\frac{\pi}{2}})^2 \qquad \textbf{(3.12)}$$

Substituíndo $sen(\alpha^*)$ obtido com o auxílio da **equação 3.11** na **equação 3.12** temos:

$$2.R.D.\cos(\alpha^*) + 2.R.D.[\frac{(d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}})^2}{4.R.D}] = (d_{\pi})^2 - (d_{\frac{\pi}{2}})^2$$

$$2.R.D.\cos(\alpha^*) + \frac{(d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}})^2}{2} = (d_{\pi})^2 - (d_{\frac{\pi}{2}})^2$$

$$2.R.D.\cos(\alpha^*) = (d_{\pi})^2 - (d_{\frac{\pi}{2}})^2 + \frac{(d_{\frac{\pi}{2}})^2 - (d_{-\frac{\pi}{2}})^2}{2}$$

$$= \frac{2.(d_{\pi})^2 - 2.(d_{\frac{\pi}{2}})^2 + (d_{\frac{\pi}{2}})^2 - (d_{-\frac{\pi}{2}})^2}{2}$$

Portanto:

$$4.R.D.\cos(\alpha^*) = 2.(d_{\pi})^2 - (d_{\frac{\pi}{2}})^2 - (d_{-\frac{\pi}{2}})^2 \qquad \textbf{(3.13)}$$

A partir das **equações 3.11 e 3.13** podemos deduzir que:

$$\frac{(d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}})^2}{sen(\alpha^*)} = \frac{2.(d_{\pi})^2 - (d_{\frac{\pi}{2}})^2 - (d_{-\frac{\pi}{2}})^2}{\cos(\alpha^*)}$$

Ou

$$tg(\alpha^*) = \frac{sen(\alpha^*)}{\cos(\alpha^*)} = \frac{(d_{-\frac{\pi}{2}})^2 - (d_{\frac{\pi}{2}})^2}{2.(d_{\pi})^2 - (d_{\frac{\pi}{2}})^2 - (d_{-\frac{\pi}{2}})^2} \qquad \textbf{(3.14)}$$

Portanto com apenas três medidas hipotéticas de distâncias entre o elemento terminal e o próximo ponto P_f da trajetória, supondo três respectivos movimentos da *i*-ésima junta, podemos encontrar qual é o incremento ótimo para q_i, com respeito a máxima aproximação ao ponto objetivo que esta determinada junta pode ocasionar **[54]**.

Podemos particionar o problema geral de controlar o rastreamento de uma trajetória em dois sub-problemas; O primeiro diz respeito a etapa inicial de movimentos que o robô deve fazer até atingir o primeiro ponto que pertence a esta trajetória, e designaremos esta etapa como etapa de aproximação; E o segundo sub-problema é o de rastreamento da trajetória propriamente dita. Sendo que a 1ª etapa pode ser omitida se o elemento terminal já está posicionado sobre o ponto inicial da trajetória ao ser posto para rastrea-la.

3.4.3 ☞ Etapa de Aproximação

Conforme a perspectiva que estamos propondo, na qual em cada momento o sistema de controle gera unicamente o movimento seguinte do robô, ao invés de gerar com antecipação a trajetória completa que o conduz à meta, o robô ao sofrer uma perturbação, ou mesmo encontrar um obstáculo, não estaria desviando de sua trajetória, porque ela não foi especificada, seu objetivo é simplesmente a próxima meta; esse fato não produz nenhuma excessão ao modo de proceder do sistema de controle.

Uma trajetória pode ser vista como composta de N-metas que o robô deve atingir uma à uma, desde a primeira até a *n*-ésima. E a distância geométrica entre cada meta é o parâmetro importante que auxiliará para que os movimentos sejam mais suaves e mais precisos (quanto mais próxima uma meta da meta seguinte, o movimento será melhor com respeito a suavidade e a precisão), mas deve-se ter em consideração que o processo é digital, e sempre haverá uma distância mínima que o robô percorrerá de uma posição a outra, que é função do tempo escolhido para cada movimento elementar, e das respectivas velocidades de juntas (estes dois parâmetros poderão ser modificados durante a execução da trajetória, por exemplo, se forem utilizados métodos de aprendizagem para atualizá-los em função de análises de movimentos bons e ruins).

Antes de começar a realizar uma tarefa, ou trajetória propriamente dita, o robô estará em um determinado estado (denominamos aqui, condição inicial). A posição do elemento terminal poderá estar próxima ou longe da primeira meta exijida pela tarefa. Se não é importante o caminho que realize até chegar nesta primeira meta, nem é importante a velocidade, nem a suavidade, somente que chegue até ela, então o sistema de controle poderá realizar o movimento de modo extremamente simples; chamaremos

esta etapa de etapa de aproximação.

Supondo que o sistema de controle não possui sensores de visão, nem de proximidade, e que possui somente sensores de posição em suas juntas, poder-se-á realizar este controle apenas utilizando as medidas destes valores de posições de juntas, e o modelo cinemático direto do robô, conforme propomos no método mais simples, onde somente uma junta é movimentada por vez; a seguir apresentamos um possível algoritmo para esta etapa.

♦ ***ALGORITMO***

— Dado o ponto-meta (1ª meta - ponto inicial da trajetória).

— Simular através do modelo cinemático direto, um movimento elementar, supondo somente uma junta movendo-se por vez, e para todas as juntas do robô (cada junta possui dois sentidos de movimento possíveis, se elas possuem um só eixo de movimento). Cada um desses sentidos de movimento provoca uma alteração na distância entre o ponto no qual está o elemento terminal e o ponto-meta considerado. E sempre haverá um dos movimentos que provocará uma maior diminuição desta distância conforme vimos anteriormente, em função do tempo escolhido para cada movimento elementar, e da velocidade imprimida pela determinada junta.

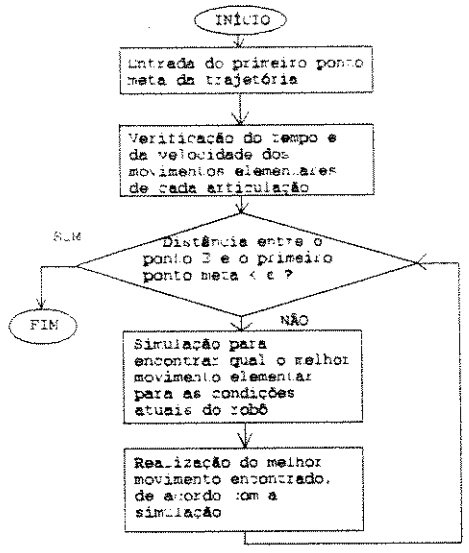
Obs: Se não houver algum critério para a alteração do tempo de movimento de cada junta, ou das respectivas velocidades durante estes tempos, então a distância cartesiana percorrida pelo elemento terminal, supondo o movimento elementar de cada junta, não poderá provocar um deslocamento cartesiano do elemento terminal de valor maior que a precisão exigida pela meta, do contrário, por este modo, poderá não haver convergência.

— Depois de encontrar os respectivos erros que cada movimento causaria, o sistema elegerá aquele que produz o menor erro, e para o próximo tempo elementar acionará a respectiva junta, com as mesmas condições da simulação. É provável que após o respectivo movimento, os sensores indiquem que o movimento não foi igual ao simulado, senão uma aproximação que dependerá da eficiência dos servomecanismos de juntas, e da intensidade dos efeitos da dinâmica, mas de acordo com o estabelecido anteriormente para a etapa de aproximação, isso não necessariamente causará maiores problemas, porque pequenos desvios colocam o robô em outra condição, digamos nova condição inicial, que será ponto de partida para a próxima avaliação do novo melhor movimento elementar a fazer em direção à meta. Também como observação citamos que estes erros detetados entre a simulação e a real ocorrência durante os movimentos, podem ser avaliados por outros algoritmos de inteligência artificial, e servir para auxiliarem na geração de níveis de controle de correção, caso seja necessário que o robô realize movimentos futuros semelhantes.

— Finalmente este procedimento é repetido até que a distância entre o elemento terminal, e o ponto-meta de aproximação seja menor do que a precisão (ε) exigida.

A seguir mostramos um possível fluxograma para que este algoritmo seja executado computacionalmente:

♦ ***FLUXOGRAMA***



Este algoritmo pode resolver esta etapa de aproximação conforme dito, se o caminho a seguir até o ponto-meta não precisar ser definido, e não houver restrições no espaço. Isso nem sempre ocorre, porque normalmente, as próprias juntas do robô tem movimentos limitados por construção. Então é necessário fazer um aprimoramento para considerar estas limitações. O exemplo a seguir serve para ilustrar uma situação na qual o algoritmo, assim como proposto, poderia fazer movimentos incorretos.

Supondo o robô de dois graus de liberdade mostrado esquematicamente na **Figura 3.8**, composto pelos segmentos l_1 e l_2 que podem mover-se respectivamente, no plano XY, conforme os ângulos θ_1 e θ_2, e que o ponto E (elemento terminal) se encontra na posição indicada; para atingir o ponto-meta, o sistema simularia os 4 movimentos possíveis (cada junta em sentido horário e anti-horário) E^I, E^{II}, E^{III}, e E^{IV} e encontraria que o melhor movimento elementar seria no sentido de E para E^I, isto é, mover a junta 2 no sentido anti-horário; claramente se executar este movimento, e os seguintes nesta direção, a junta 2 seguirá em direção ao seu limite físico determinado pelo ângulo limite entre l_1 e l_2, impossibilitando atingir-se o ponto-meta.

Uma maneira de resolver este problema é fazer com que o algoritmo seja executado normalmente testando até que alguma das juntas (ou mais de uma) atinja seu limite, então bloquear seus movimentos para o sentido não permitido, e verificar quais são os outros movimentos melhores; isto resolve desde que o ambiente dentro da região de ação do robô esteja livre, do contrário, será necessário a incorporação de outras regras de decisão, com utilização de sinais de outros sensores mais adequados.

É importante destacar que este algoritmo pode ser aplicado para robôs com qualquer número de graus de liberdade. E também que na realidade é gerada uma seqüência de movimentos que poderão ser armazenados na memória do sistema, para utilização futura, por exemplo, para que possam ser utilizados na execução desta mesma tarefa novamente; isso talvez não seja necessário para a etapa de aproximação, uma vez que o caminho a seguir não é importante, por outro lado, uma análise desta seqüência de movimentos gerada, pode resultar na obtenção de seqüências melhores para o mesmo

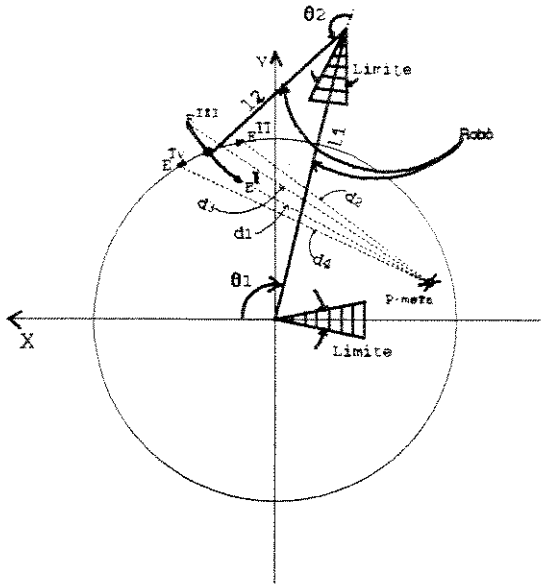


Figura 3.8 - Robô de 2 GDL.

movimento; por exemplo, aumentar a rapidez dos movimentos, movendo mais de uma junta num mesmo tempo elementar, por aprendizagem, em função de como foi a seqüência imediatamente anterior. Em **[3][4][5][8][20][26][27]** estão propostas idéias e métodos de aprendizagem, embora com uma filosofia diferente desta, mas que podem inclusive ser incorporadas sob esta perspectiva.

Ao repetir uma mesma seqüência o sistema só necessitará amostrar, em cada tempo elementar, os respectivos valores "aprendidos", e verificar se a resposta do robô foi semelhante através da análise das respostas dos sensores; em caso de semelhança, prosseguir até que o ponto-meta seja atingido, e em caso de diferenças acentuadas, decidir através do mesmo algoritmo se deve abandonar a seqüência aprendida, ou apenas adaptá-la a esta nova situação, etc...

No caso do robô de 2 graus de liberdade citado no exemplo anterior, podemos definir os movimentos possíveis como:

- 1D. Movimento no sentido horário da junta 1.
- 1E. Movimento no sentido anti-horário da junta 1.
- 2D. Movimento no sentido horário da junta 2.
- 2E. Movimento no sentido anti-horário da junta 2.

E uma seqüência de movimentos elementares poderia ser:

1D 1D 2E 1D 2E 2E 2E 1D 1D 2D 1D...

Note que através de uma seqüência assim pode-se fazer agrupamentos de movimentos, por exemplo, sempre que houver movimentos elementares seguidos de juntas diferentes, eles podem ser executados no mesmo tempo elementar, numa execução futura, e sempre que a determinada ocorrência de movimento de uma junta for seguida, mesmo depois de vários movimentos elementares das outras, por um movimento em sentido

contrário desta mesma junta, os dois movimentos poderão ser cancelados, desde que os respectivos tempos elementares sejam os mesmos.

Neste caso a seqüência anterior ficaria:

$$\begin{matrix} 1D\ 1D & 1D\ 1D & \\ 1D & 2E & 1D \\ 2E\ 2E & 2E\ 2D & \end{matrix} \Rightarrow \begin{matrix} & 1D\ 1D\ 1D & \\ 1D & & 1D\ 1D \\ & 2E\ 2E\ 2E & \end{matrix}$$

Portanto, se os efeitos dinâmicos originados por esta nova seqüência não forem significativos, um movimento equivalente poderia ser realizado em 5 tempos elementares a menos. Se os controladores de cada junta garantirem boas respostas, um procedimento assim, faria o robô responder de forma muito mais rápida. Como existe este problema intrínseco ao método, dependendo da condição inicial em que se encontra o robô do próximo ponto-meta, e dos limites impostos pela construção física do robô, pode ocorrer erro na decisão do caminho a seguir, e em conseqüência o robô ficar parado (*dead lock*) em alguma posição, sem o sistema de controle encontrar a solução.

O critério utilizado como base para que o sistema encontre soluções é a distância cartesiana entre o elemento terminal e cada ponto-meta; então pode ocorrer a situação mostrada pela seqüência a, b e c da **figura 3.9**.

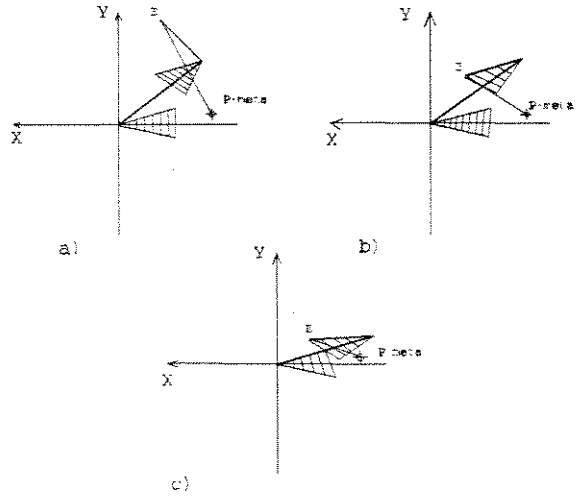


Figura 3.9 - Situação de Travamento.

Note que no caso c, tanto a junta 1 como a 2 estão situadas sobre a sua linha limite de movimentos, e o algoritmo faria o manipulador ficar travado, porque partindo desta condição, para diminuir a distância *d* entre E e P, seria necessário mover as juntas no espaço que fisicamente é impossível. Portanto, faz-se necessário incrementar o algoritmo com uma rotina que "perceba" quando ocorrem situações assim, e tome decisões de movimentos no sentido de evitar que ocorram travamentos. A idéia é inserir pontos-meta intermediários entre a posição atual do robô e o próximo ponto de passagem requisitado pela tarefa, de forma que o caminho seja possível de ser seguido.

Na realidade o que ocorre é que para determinadas regiões do espaço de trabalho, o robô pode ter um número variado de configurações estruturais para atingí-las,

dependendo dos graus de liberdade que possui e das restrições em cada grau. Pode acontecer, por exemplo, que para algum determinado ponto, só tenha uma configuração possível, ou pode ser que tenha mais de uma, etc... Também dependendo do passo diferencial que cada junta faz, segundo a escolha do projetista, pode também ocorrer travamento (ver **figura 3.10**).

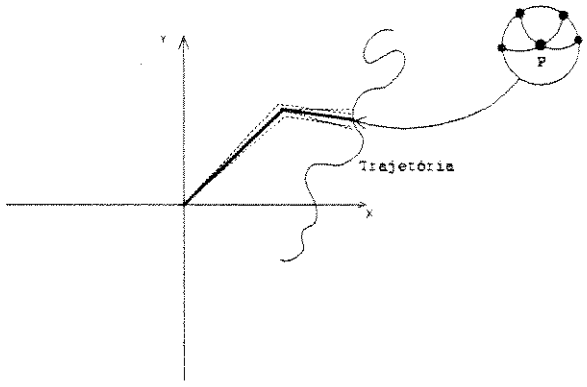


Figura 3.10 - Possibilidade de Travamento.

Isto é, podem ocorrer situações onde qualquer movimento diferencial que o robô tentar fazer, o erro (módulo), em relação ao próximo ponto-meta, seja igual ao erro anterior, e portanto, não poderia progredir no sentido de diminuí-lo. Se tal erro é menor que a precisão que se queira, então tudo bem, senão é necessário que o algoritmo, de alguma forma, faça mudanças convenientes no passo diferencial das juntas (por exemplo como estamos propondo no terceiro modo de encontrar o passo para mover cada junta - ver mais adiante). Isso pode ser facilmente resolvido, se houver uma parte da rotina que fique constantemente verificando a ocorrência de tais situações, e faça alterações nos passos em função da atual configuração da estrutura. Existe um agravante, que então é necessário que o sistema de controle fique armazenando os valores dos passos dados para cada junta, no tempo, como parte integrante da seqüência de movimentos, se deseja-se que o robô repita posteriormente o mesmo movimento (Isto é, se quiser-se "aprender"), Isto pode se tornar um problema, porque há necessidade de armazenar maior quantidade de dados na memória. Por outro lado, pode-se à priori, encontrar quais devem ser os máximos movimentos diferenciais tais que sempre o robô trabalhe dentro da precisão estipulada (evidentemente quanto menores os passos elementares, mais lento ficará o robô, para períodos de amostragem iguais da malha de trajetórias).

Com a idéia de incrementar o algoritmo com pontos-meta intermediários entre a posição atual do robô, e o próximo ponto exigido pela trajetória, propomos o algoritmo que está mostrado na **figura 3.11** em forma de fluxograma (Cada meta intermediária é chamada de meta de aproximação).

Basicamente este é um método onde o sistema de controle gera referências de posição para os servomecanismos de juntas de forma a reagir diretamente em função da medidas recebidas dos sensores destas juntas, buscando soluções, ainda que não ótimas, mas razoáveis relativamente a convergência, e a suavidade dos movimentos da estrutura, em base à incerta e escassa informação disponível. E com possibilidade de melhorar seu desempenho ao repetir tarefas semelhantes com o passar do tempo, sempre em função de uma próxima meta a seguir até que seja atingida. Como dispomos de uma arquitetura de controle, citada anteriormente, que faz parte do projeto **JECAII**, e é

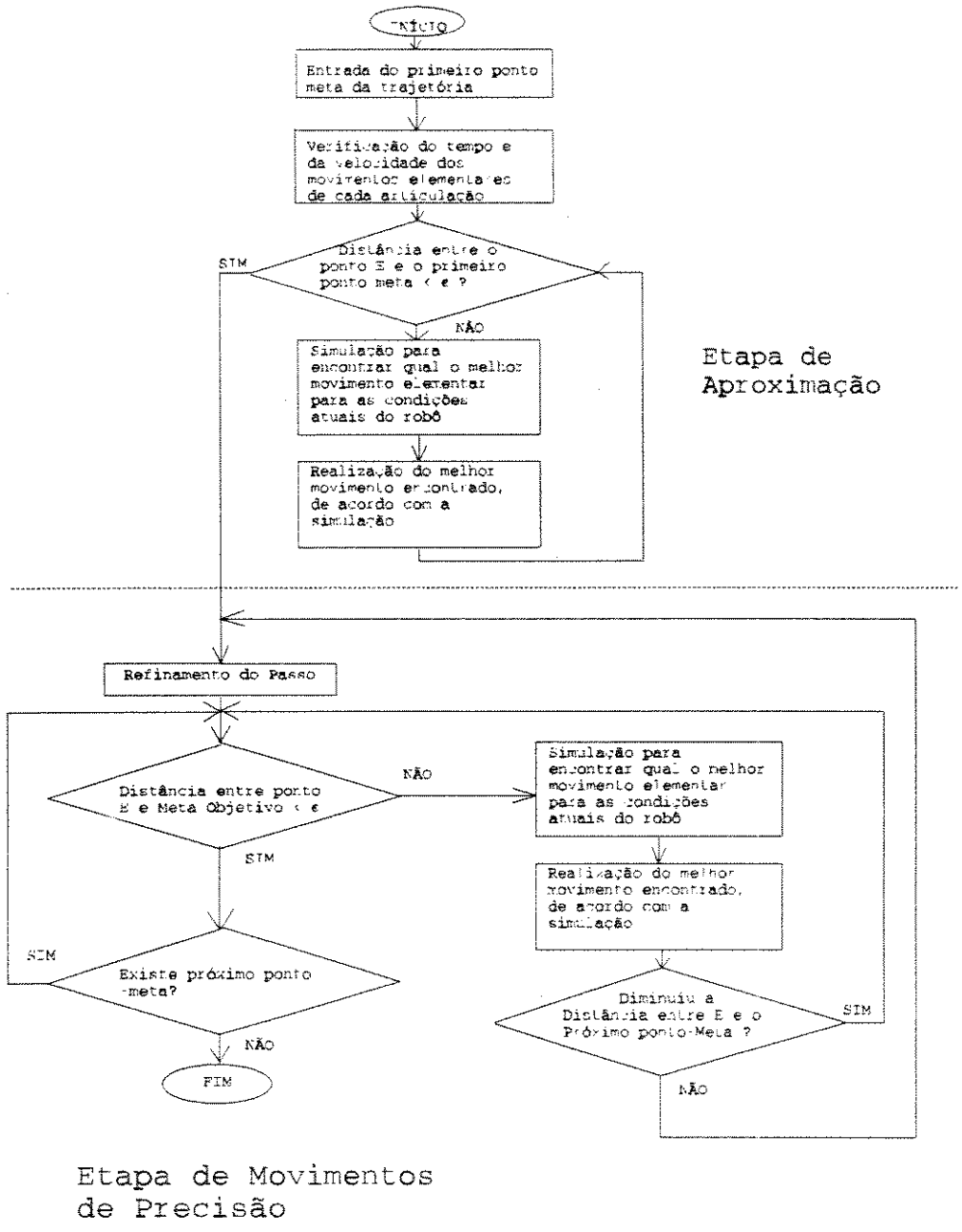


Figura 3.11 - Fluxograma para Etapa de Aproximação com Metas Intermediárias.

hierárquica com dois níveis tipo Mestre-Escravos, onde há um servomecanismo para o controle de cada junta do robô, em um nível hierarquicamente inferior (Escravos), e um sistema de coordenação geral de movimentos em um nível hierarquicamente superior (Mestre). Adequamos este método de controle de trajetórias proposto a tal sistema, através da incorporação de uma malha de controle que pode ser vista na forma de diagrama de blocos na **figura 3.12**, onde a característica principal é a não utilização dos modelos inversos do robô.

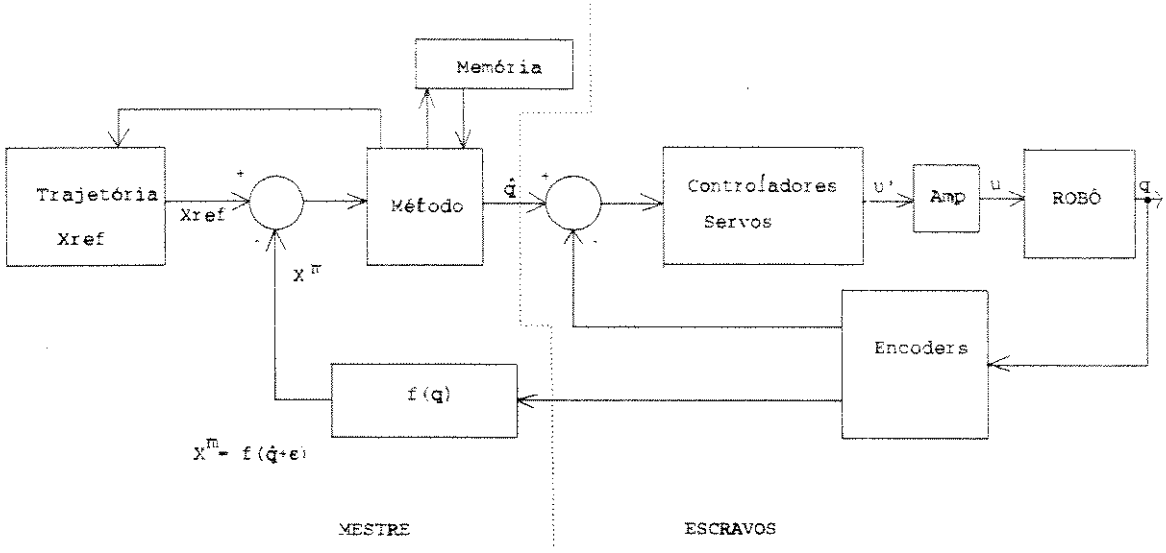


Figura 3.12 - Diagrama de Blocos do Método.

3.4.4 ☞ Etapa de Rastreamento da Trajetória

No método fazendo com que cada junta seja movida com um passo fixo, determinamos apenas qual o melhor sentido de seu movimento para que sempre a distância entre o elemento terminal e o próximo ponto objetivo seja diminuida até estar dentro de uma precisão (ε) especificada pela trajetória; garantimos que, em posição, o problema de rastrear a trajetória pode ser resolvido, mas verificamos que em velocidade e aceleração os movimentos não têm boas características, isto é, produzem-se acelerações muito intensas e intermitentes, fato que pode ocasionar vibrações indesejáveis na estrutura; por outro lado há uma característica interessante, porque o projeto dos controladores de juntas, neste caso, fica bastante simples, bastando projetá-los para que atuem com eficiência, por exemplo, de forma criticamente amortecida para variações sempre constantes dos níveis de ajustes fornecidos pelo gerador de trajetórias (entradas). No caso em que somente uma junta é movida por vez, a resposta é também muito lenta, e os movimentos muito pouco otimizados com relação a diminuição da distância entre o elemento terminal e o próximo ponto objetivo em cada intervalo de amostragem da malha de controle de trajetórias, mas com este modo de abordar o problema, podemos reavaliar a seqüencia de procedimentos ocorridos durante a execução do rastreamento da trajetória, e fazer melhoramentos se acaso a mesma trajetória for repetida novamente; ou seja, inserir procedimentos de inteligência artificial no algoritmo. Outra vantagem evidente desse procedimento é quanto a exigência computacional extremamente minimizada, portanto, dependendo do tipo de aplicação pode ser muito eficiente.

Podemos melhorar enormemente a questão de aproximação mais rápida entre o elemento terminal do robô e cada próximo ponto objetivo, mantendo passos fixos

para o movimento de cada junta, porém permitindo o movimento de uma ou mais juntas, segundo a melhor combinação possível para cada período da malha de controle de trajetórias; para este modo, as exigências computacionais são bem maiores do que para o modo anterior, enquanto naquele modo é necessário verificar 6 possibilidades de movimentos (para três graus de liberdade), ou seja, para cada junta nos dois sentidos; neste, estas possibilidades aumentam para 27 (3 fatorial), mas ainda assim, como em cada verificação de possibilidade de movimento, a quantidade de processamento é mínima; dependendo da aplicação, também pode resultar num modo bastante eficiente de se controlar o rastreamento de trajetórias, e como constatamos, há uma melhoria bastante acentuada nas respostas de acelerações e velocidades, ainda que continuem ocorrendo oscilações acentuadas nas acelerações.

Nestes dois métodos citados anteriormente garantimos que os pontos de interesse da trajetória sejam atingidos com precisão, e também a convergência dos respectivos algoritmos é sempre garantida, mas não está garantida a suavidade dos movimentos, porque não utilizamos nenhum critério de otimização. Então surge a pergunta: Será possível, com o mesmo tipo de abordagem, encontrar movimentos de juntas que tenham em consideração as características de eficiência destes dois métodos, e que otimizem os critérios de movimentos em tempo mínimo e de suavidade nos movimentos do elemento terminal?

A resposta a esta pergunta é sim, porque como verificamos anteriormente, é possível encontrar o movimento que cada junta deve realizar para que este seu movimento cause a máxima aproximação possível em um determinado período da malha de controle de trajetórias do elemento terminal do robô com respeito ao próximo ponto objetivo imposto pela trajetória. Lembramos que este método, assim como os anteriores, também é sub-ótimo, porque apesar de encontrarmos o movimento ótimo para cada junta, não temos a garantia do movimento ótimo conjunto, que em outras palavras, é a solução cinemática exata que pode ser obtida com a solução analítica do modelo cinemático inverso do robô, mas que estamos propositalmente evitando, devido a como é de conhecimento, necessitar de grande carga computacional, e não ter como corrigir erros causados por perturbações imprevistas durante os movimentos. Mas como se poderá verificar mais adiante, quando forem mostrados alguns resultados de simulações, é de grande robustez, mantém a característica de convergência do algoritmo, minimiza enormemente o tempo necessário para atingir cada ponto da trajetória, e suaviza muito as respostas de velocidades e acelerações da estrutura, o que nos indica que é possível inserir critérios de controle destas variáveis conjuntamente com o controle da variável de posição para o rastreamento.

Como também fica evidente, para a utilização eficiente deste terceiro modo proposto, os servomecanismos de juntas devem ter projetos um pouco mais sofisticados que para os dois modos anteriores, porque os níveis de ajustes de controle (entradas), não terão variações constantes, apesar de também, mais adiante, mostrarmos que mesmo com servomecanismos não muito precisos, pode-se conseguir movimentos eficientes dependendo das precisões exigidas pelas tarefas que impõem as trajetórias a serem rastreadas pelo robô.

A seguir fazemos a proposição de um algoritmo que utiliza conceitos de inteligência artificial e integra estas idéias apresentadas anteriormente.

3.4.5 ☞ Algoritmo de Busca Heurística em Tempo-Real

Conforme a definição apresentada em **[62]**, heurísticas são critérios, métodos, ou princípios para decidir quais dentre diversas alternativas de ação permitem mais eficácia no sentido de conseguir um determinado objetivo. E elas representam compromissos entre dois requisitos: a necessidade de fazer tais critérios simples, e ao mesmo tempo, de tentar que estas alternativas sejam discriminadas corretamente entre boas e más escolhas.

Nós dirigimos nossos carros mesmo sem saber bem como eles funcionam, e somente com um vago quadro mental das condições das estradas a frente, escrevemos programas complexos para computadores atentando somente para frações das possibilidades e interações que podem estar colocadas na atual execução destes programas, escolhemos nosso caminho com sucesso em intrincadas situações de trânsito tendo somente uma leve informação da situação, ou lugar onde estão as pessoas à volta, e ainda quando esta informação gera uma expectativa falha, nós utilizamos fortemente o "humor" e tentamos graciosamente de novo... ainda não há máquinas ou programas de computadores que exibam tais capacidades assim como nós seres humanos as possuimos, mas, no futuro, quem sabe possa haver esta aproximação. Evidentemente, as poucas informações que temos estão efetivamente organizadas tão bem, que agimos normalmente, e eficazmente nestas e em outras atividades diárias.

Estas informações, ou conhecimento, devem por algum caminho, serem resumidas ou abstraídas, tal que possam prover, como a intuição humana, uma seqüência de estados de tentativas, que muitas vezes não necessitam ser refinadas, com "conselhos" velozes de informação para administrar os passos das primitivas computacionais que fazem a solução do processo.

O conteúdo da informação desses "conselhos" poderia ser chamado de conhecimento heurístico, e certamente para obtê-los são necessários estudos empíricos e teóricos, para encontrar como adquiri-los, armazená-los e usá-los, assim como as pessoas o fazem, e também como representá-los e processá-los utilizando máquinas. E por fim conseguir sucesso em soluções de problemas através de métodos melhores que aqueles que não possuem eficácia, ou falham neste intento.

Podemos abordar o problema de rastreamento de trajetórias que fazem parte da execução de uma tarefa por um robô, como um problema onde temos uma série de objetivos a alcançar (atingir pontos no espaço com seu elemento terminal, e de uma determinada maneira) com compromissos de custo e performance de seus movimentos estruturais. Portanto, como um problema de otimização, onde temos um espaço de candidatos objetos (movimentos de juntas), que eleitos adequadamente, no tempo, conduzem à solução.

Colocando desta forma, necessitamos de três requisitos para solucionarmos o problema com utilização de um computador:

1) Representar cada objeto candidato a participar da solução com um código ou símbolo estrutural (estados das variáveis de posições de juntas e suas duas primeiras derivadas).

2) Produzir ferramentas computacionais capazes de efetuar pesquisas para relacionar os códigos dos objetos segundo o objetivo destas pesquisas e transformá-los sistematicamente, se for o caso, (algoritmo de avaliação e geração de estados para as variáveis de juntas de acordo com as premissas impostas pelas trajetórias a serem seguidas pelo robô).

3) Produzir um método eficiente de escalonamento dessas tranformações tal que se produza um objeto eficaz o mais rapidamente possível (algoritmo de decisões de movimentos rápido e eficiente).

A qualidade que conseguirmos dar para cada um destes ítens anteriores, é que determinará o desempenho que o robô terá ao ser posto para realizar uma tarefa.

♦ *ALGORITMO A* PARA TEMPO-REAL (RTA*)*

O algoritmo A* é um algoritmo que incorpora as premissas apresentadas nos três ítens anteriores, e está proposto como uma técnica para solução de problemas com base em conhecimentos heurísticos sobre tais problemas **[62]**. Como especificamente estamos interessados no problema de controle de trajetórias de robôs, direcionaremos sua aplicação para isso. Em passos, o algoritmo pode ser posto como segue, onde f é a função custo a otimizar:

passo 1. Colocar o nó inicial (estado inicial do robô, ao ser posto para realizar uma tarefa) em ABERTO.

passo 2. Se ABERTO está vazio, terminar com indicação de falha.

passo 3. Remover de ABERTO e colocar em FECHADO o nó n para o qual f é mínima.

passo 4. Se n é o nó objetivo, terminar com sucesso e obter a solução por verificação de retorno dos apontadores de n para s.

passo 5. Senão expandir n, gerando todos os seus sucessores, e assinalando todos eles com apontadores de retorno por n. Para cada sucessor n' de n:

a) Se n' não está pronto em ABERTO ou FECHADO, estimar $h(n')$ (uma estimação do custo do melhor caminho para n' para algum nó objetivo), e calcular $f(n') = g(n') + h(n')$ onde $g(n') = g(n) + c(n,n')$ e $g(s) = 0$. Onde $c(n,n')$ é o custo para passar de n para n'.

b) Se n' está pronto em ABERTO ou FECHADO, direcionar seu ponteiros para o caminho que leva ao menor $g(n')$.

c) Se n' requisitar ajuste de ponteiro e for encontrado FECHADO, reabri-lo.

passo 6. Ir ao passo 2.

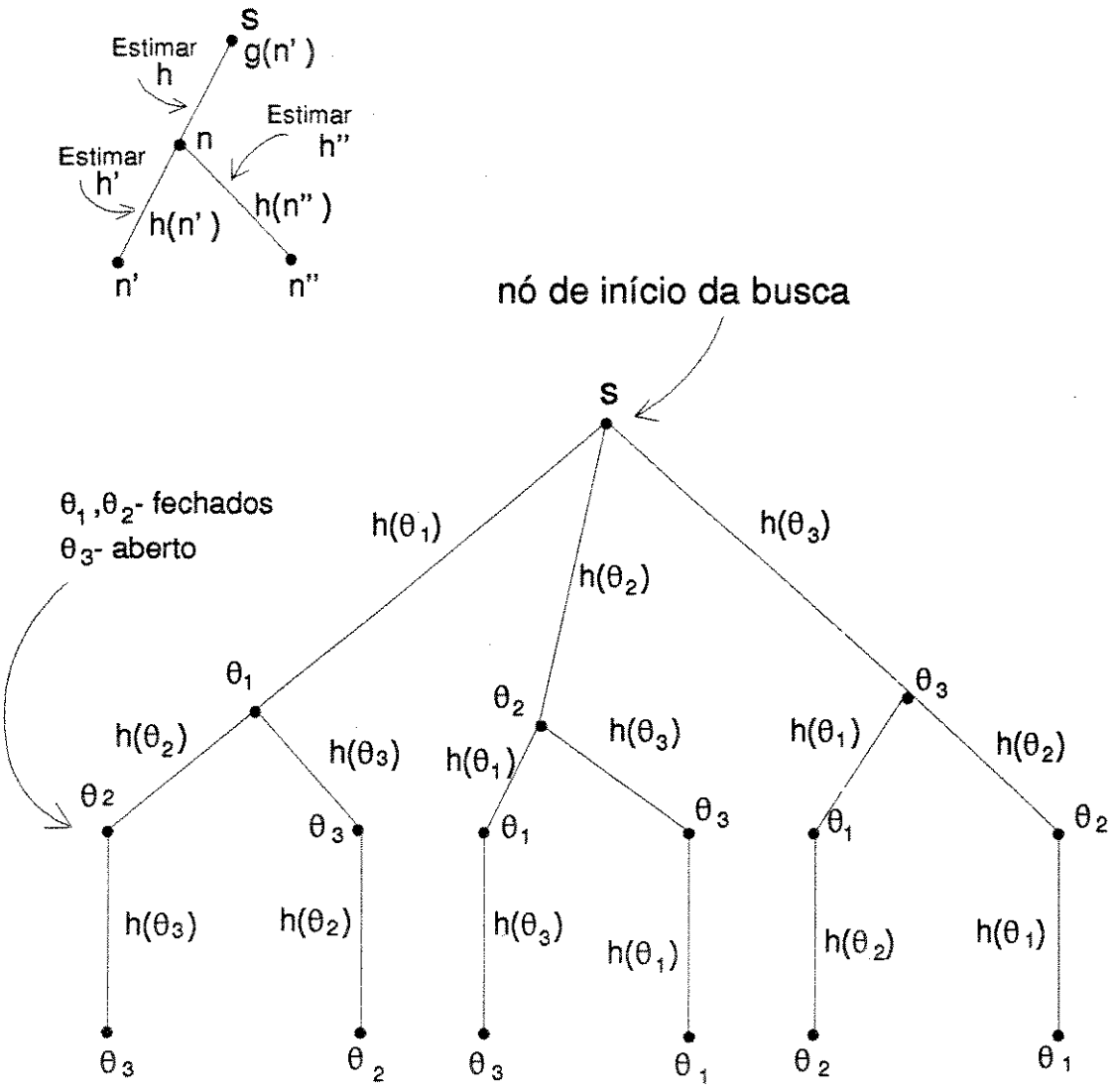


Figura 3.13 - Árvore de Busca para 3 GDL s={θ_1,θ_2,θ_3}.

Cada nó que sucede o nodo *n* é chamado de nó "filho" de *n*. E este algoritmo é utilizado para pesquisar e encontrar soluções baseado em informações heurísticas postas

em uma estrutura de grafo, onde as vizinhanças de um corrente estado (nó), são geradas a partir de uma função heurística que provê dados para estas gerações, até que um novo estado que minimize esta função seja encontrado. O algoritmo repete este ciclo tantas vezes quantas forem necessárias até que f seja mínima ou satisfaça algum critério de precisão.

O algoritmo RTA* faz decisões ótimas localmente, e proporciona garantia de encontrar solução para qualquer problema que tenha solução, independentemente dos valores heurísticos iniciais. A prova desta última colocação está em **[38]**, e por possuir esta característica é um algoritmo de grande interesse para solucionar uma ampla gama de problemas de modo simples, enquanto que através de métodos tradicionais a solução talvez exija um procedimento muito mais complexo.

Posto isso, faremos agora a apresentação do algoritmo que propomos com base no desenvolvimento que apresentamos anteriormente.

Definimos, com índice k= 1...n, as funções:

$$f(k) = h(k-1) - h(k)$$
$$d(k) = \sum_{i=1}^{k} f(i) = h(0) - h(k)$$
$$onde\ h(k) = |Pf - Pe(k)|^2 \quad e \quad g = h(0) = |Pf - Pe(0)|^2$$

e os conjuntos

$$K(0) = \emptyset$$
$$S(0) = I = \{1,...,n\}$$
$$K(k) = K(k-1) \cup \{i_k\} : \textit{lista de nós fechados,}$$
$$onde\ i_k \in S(k)\ ou\ S(k-1) : \textit{nó selecionado na iteração } k\textit{-ésima}$$
$$S(k) = I - K(k) : \textit{lista de sucessores de } i_k$$

Com estas definições podemos formular comodamente o algoritmo de busca. A idéia do algoritmo consiste em selecionar iterativamente uma seqüência de juntas que individualmente realizem a máxima aproximação ao ponto objetivo, excluíndo as juntas já selecionadas. Neste sentido é um algoritmo de busca heurística: as sucessoras de uma junta serão aquelas juntas não selecionadas previamente. A função de custo que utilizaremos será $f(k)$ **[16]**.

Passadas n iterações haverão sido selecionadas todas as juntas, podendo-se repetir o processo indefinidamente até alcançar o objetivo dentro de uma precisão especificada ou até que se esgote o período de amostragem da malha de controle de trajetórias do robô.

Demonstramos mais adiante, que o algoritmo converge para a solução ótima, assegurando-se a aproximação ao objetivo em cada uma das iterações do processo (mais adiante também falaremos de uma situação limite que chamaremos singular, quando P_f

se situa no eixo x_i $\forall i$), pelo qual a lista de nós ABERTOS pode ser FECHADA depois de cada iteração.

o **ALGORITMO**

Repetir

FOR $k = 1...n$

{ 1) $\forall\, i_j \in S(k-1),\ (j = k...n)$

a) Cálculo do deslocamento Δq_j

(Ver seção seguinte)

b) Cálculo de $f(j)$

c) Escolha de $i_k \mid f(k) = \max_j f(j)$

2) *Atualizar* $K(k)$ *e* $S(k)$

}

Até a precisão desejada ou até que o tempo de cálculo disponível se esgote.

Executar $q(k+1) = q(k) + \Delta q$

Pode-se verificar que a ordem de complexidade do algoritmo é $O(n^2)$, sendo n o nº de graus de liberdade. Entretanto, é possível paralelizá-lo nos *passos 1.a e 1.b*, reduzindo a sua complexidade para $O(n)$, mas deve-se levar em consideração a comunicação necessária entre os processadores para resolver o *passo 1.c*.

Em termos absolutos, o tempo de execução em um PC-486 (50MHz), seqüencial portanto, é de 0.7ms. Neste caso, se consideramos um período de amostragem da malha de controle de trajetórias de 5ms, será possível melhorar a precisão de rastreio repetindo este processo de busca até aproximadamente 7 vezes. Nas simulações realizadas foram necessárias apenas de 2 a 3 repetições para alcançar as precisões desejadas (ver **Capítulo IV**).

♦ ***CÁLCULO DO DESLOCAMENTO INCREMENTAL***

Na **Figura 3.14** está representada a evolução de h das três primeiras juntas do robô **JECAII** (Ver **figura 4.1**), em relação a q entre $-\pi$ e π, para uma condição inicial em que $\theta1=\theta2=\theta3 = \frac{\pi}{4}$, e o próximo ponto requisitado por uma trajetória genérica é (X,Y,Z)=(200,200,200) (em milímetros). Verifica-se que h têm um máximo em um ponto que chamaremos *p2*, e um mínimo em outro que chamaremos *p1*, situados de tal forma que $p1 - p2 = \pi$. A reta horizontal indica o valor de *h(0)*, ou seja, a distância do elemento

terminal ao objetivo para $\Delta q = 0$. Vemos que salvo para um caso, a primeira junta alcança seu ótimo, e existe um intervalo de valores de Δq que aproximam o elemento terminal do braço ao objetivo, ou seja, existe um intervalo de valores de Δq onde h diminui, para todas as juntas.

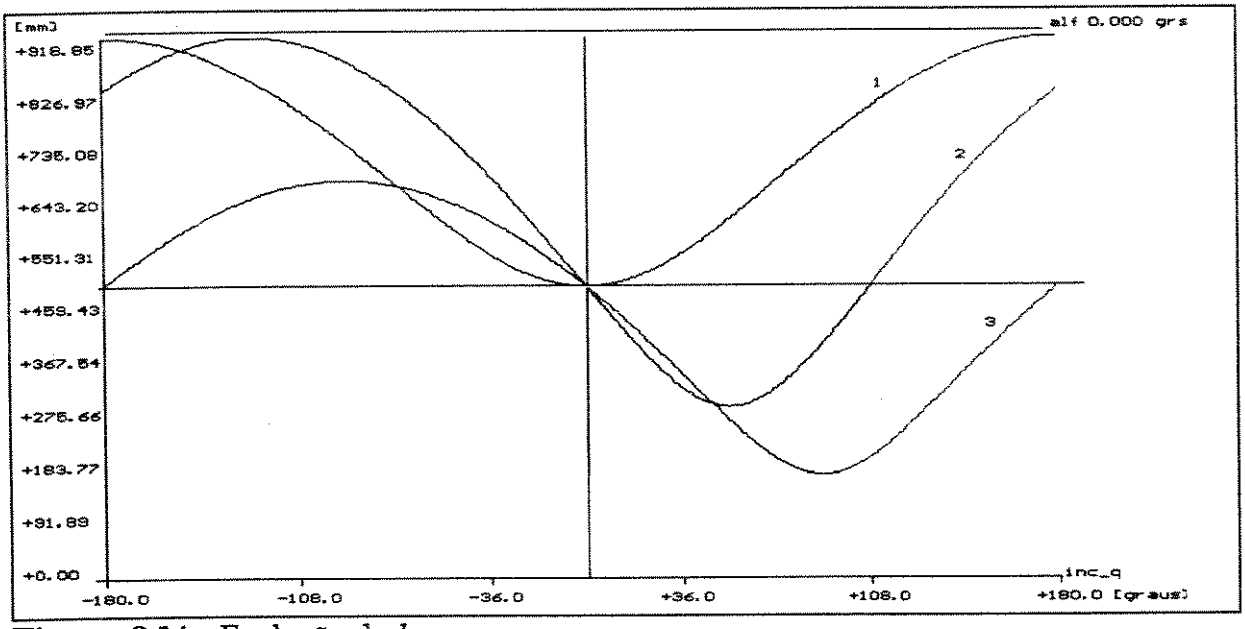


Figura 3.14 - Evolução de h.

Pode-se encontrar o mínimo analiticamente derivando h com respeito a q:

$$h(k+1) = (P_f^i - P_e^i(k+1))^T.(P_f^i - P_e^i(k+1))$$
$$= P_f^{iT}.P_f^i + P_e^{iT}(k).P_e^i(k+1) - 2.P_f^{iT}.P_e^i(k+1)$$

com: $P_e^i(k+1) = R_{zi}(\Delta q_i).P_e^i(k)$

$$e\ R_{zi}(\Delta q_i) = \begin{bmatrix} C_i & -S_i & 0 \\ S_i & C_i & 0 \\ 0 & 0 & 1 \end{bmatrix} : Ortogonal \qquad \begin{matrix} C_i = \cos(\Delta q_i) \\ S_i = sen(\Delta q_i) \end{matrix}$$

Portanto,

$$\frac{dh(k+1)}{d\Delta q_i} = -2.P_f^{iT}.\ [-r_i S_i\ r_i C_i\ 0]^T = k_1 C_i + k_2 S_i$$

r_i é a distância do elemento terminal ao centro de coordenadas da junta i.

já que $P_e^i = (r_i\ 0\ 0)^T$

com, $k_1 = -2.r_i.P_{f_y}{}^i \quad ; \quad k_2 = 2.r_i.P_{f_x}{}^i$

e daqui que:

$$tg(\Delta q_i) = -\frac{k_1}{k_2} = \frac{P_{f_y}{}^i}{P_{f_x}{}^i} \qquad \textbf{(3.15)}$$

Por esta razão sempre poderemos encontrar em cada iteração do algoritmo, um movimento incremental de alguma junta que faz f positivo. Sem dúvida, deve ficar claro que a solução não é ótima, senão tão só sub-ótima **[84]**. A solução ótima faz $d(n)$ positiva, porém, pode ser que algum valor intermediário de f deva ser negativo. A escolha de f como medida a maximizar é, por tanto, demasiadamente conservadora, já que perde de vista o acoplamento entre juntas. Um algoritmo mais avançado deveria ter em conta algum aspecto de coordenação de movimentos. Como dissemos pretendemos que o algoritmo seja executado em tempo-real. Assinalamos que o robô não só tem limitações de posicionamento extremas:

$$q_{i,\min} \le q_i \le q_{i,\max} \qquad \textbf{(3.16)}$$

E também de velocidade e aceleração; estas são impostas pelos motores, e uma vez definido o período de amostragem, fica condicionado o valor máximo que pode alcançar Δq_i, e por outro lado, deve-se levar em consideração os aspectos de fricção que condicionam o valor mínimo:

$$|\Delta q_{i,\min}| \le |\Delta q_i| \le |\Delta q_{i,\max}| \qquad \textbf{(3.17)}$$

Todas estas classes de limitações devem ser incluídas no critério de seleção da junta. Desta forma se estará em condições de avaliar de maneira realista a aplicabilidade do algoritmo.

A solução da **equação 3.15** fornece o valor de deslocamento incremental da i-ésima junta, que utilizaremos no *passo 1.a* do algoritmo de busca. Se este valor estiver fora dos limites **(3.16)** ou **(3.17)** se escolherá o valor máximo possível, e pode-se obter o sentido de movimento calculando a projeção de P_f^i sobre o plano (x_i,y_i) e observando que:

$$k_1 = \frac{(h_{\frac{\pi}{2}} - h_{-\frac{\pi}{2}})}{2} \quad ; \quad k_2 = \frac{(h_\pi - h_0)}{2} \qquad onde\ h_\alpha = |Pf - Pe|^2 \ \ para\ \Delta q_i = \alpha$$

Pelo fato de que o critério de seleção de juntas se apoia inteiramente no cálculo de h, ou seja, na equação cinemática direta, qualquer erro no valor dos parâmetros cinemáticos ou na medida de q dará uma estimação incorreta de h o que pode provocar que o algoritmo não convirja. No entanto, constatamos mediante simulações (ver mais adiante) que se a amplitude do erro (modelado como uma constante ou como ruído aleatório) é pequena em relação com a precisão que é exijida do resultado, o algoritmo converge, ainda que mais lentamente. Naturalmente a utilização de algum sensor estereoceptivo situado no elemento terminal do robô permitiria obter $h(0)$ com maior precisão e os erros não seriam acumulativos.

CAPÍTULO IV

MODELOS, EXPERIMENTOS, RESULTADOS E CONCLUSÕES

4.1 ☞ AMBIENTE UTILIZADO NOS EXPERIMENTOS

Realizamos vários experimentos de simulação, utilizando desde geração e rastreamento de curvas simples como retas e circunferências, até curvas mais complexas, com utilização de funções trigonométricas, com auxílio do pacote de *software* desenvolvido especialmente para esta finalidade (Ver **Capítulo I**), e obtivemos resultados que comprovam a eficiência do método para o rastreamento dessas trajetórias dentro dos graus de precisão requisitados, e com rapidez suficiente para o controle em tempo-real. Utilizamos computadores tipo IBM PC AT - compatíveis para executá-los. E as medidas dos elos do robô utilizadas foram (**figura 4.1**) l1 = 340mm, l2 = 400mm, e l3 = 250mm (medidas reais do robô **JECAII**). Para todos os experimentos que fizemos utilizando uma CPU 286, o tempo máximo de cálculos, utilizando o modo computacional mais complexo do método, por iteração, nunca ultrapassou os 60ms, e em uma CPU tipo 486 (relógio 50MHz), este tempo nunca ultrapassou os 4ms. Estes tempos foram obtidos utilizando-se uma precisão de rastreamento de posição de 2 centésimos de milímetro (erro de aproximação máximo admitido a cada ponto pertencente as trajetórias) para todos aqueles experimentos onde não são citadas informações adicionais.

Realizamos comparações relativas ao método de solução com a equação inversa analítica, e verificamos que, conforme exigimos mais precisão no rastreamento, os resultados obtidos pelo método proposto convergem para tal solução.

Como faz-se necessária a utilização do modelo cinemático direto do robô em todos os modos do método proposto, e possuímos o robô **JECAII**, primeiramente fazemos a apresentação deste seu modelo matemático para os graus de liberdade utilizados no seu posicionamento espacial, de forma a podermos argumentar e discutir os resultados das simulações que apresentamos em seguida.

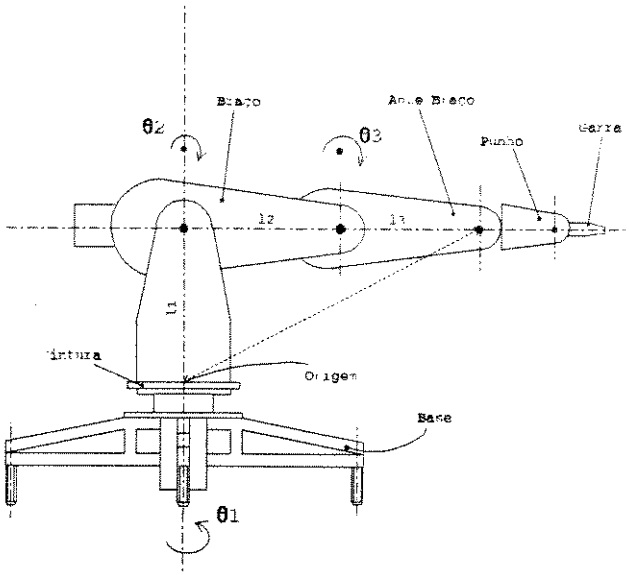


Figura 4.1 - Robô **JECAII**.

4.2 ☞ MODELO CINEMÁTICO DIRETO DE POSIÇÃO DO JECAII

Com o auxilío da **figura 4.2**, que mostra a configuração geométrica dos três primeiros graus de liberdade do **JECAII**, podemos verificar que:

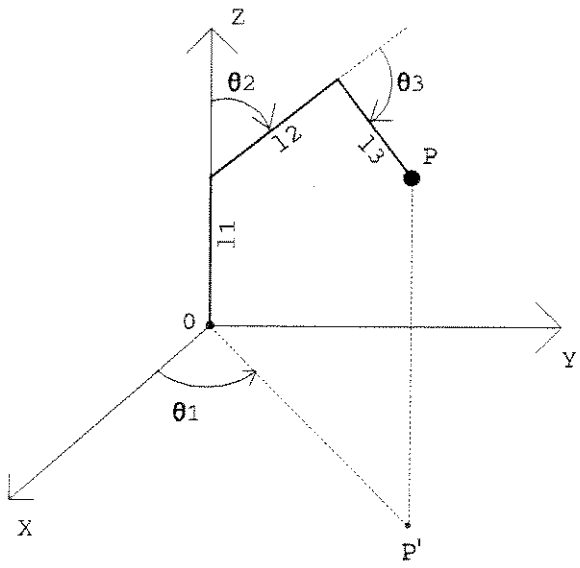


Figura 4.2 - Geometria para Modelo Cinemático Direto.

$$Px = (l2.sen(\theta 2) + l3.sen(\theta 2 + \theta 3)) * \cos(\theta 1) \quad \textbf{(4.1)}$$

$$Py = (l2.sen(\theta 2) + l3.sen(\theta 2 + \theta 3)) * sen(\theta 1) \quad \textbf{(4.2)}$$

$$Pz = l1 + l2.\cos(\theta 2) + l3.\cos(\theta 2 + \theta 3) \quad \textbf{(4.3)}$$

Como faremos algumas comparações de desempenho relativas a soluções obtidas com a utilização do modelo cinemático inverso também apresentamos o modelo matemático que utilizamos para isso.

4.3 ☞ MODELO CINEMÁTICO INVERSO DE POSIÇÃO DO JECAII

Como para este tipo de configuração estrutural existem quatro soluções geométricas distintas possíveis para o posicionamento do ponto *P* no espaço **[27]**, supondo a estrutura sem restrições nas juntas, definimos dois indicadores de configuração para que o modelo apresentado se torne válido para todas estas soluções. Chamamos estes indicadores respectivamente de Indicador de BASE e Indicador de COTOVELO definidos assim:

Base: Direita - θ1 positivo (+1)
Esquerda - θ1 negativo (-1)

Cotovelo: Acima - Cotovelo acima
Abaixo - Cotovelo abaixo

4.3.1 ☞ Solução Para a Junta 1

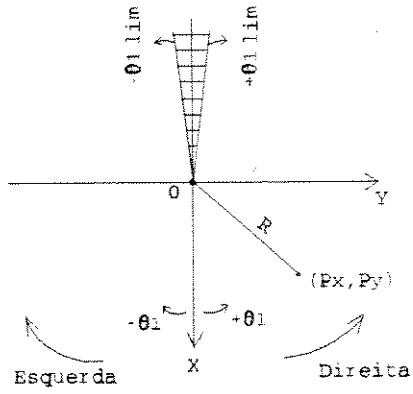


Figura 5.3 - Plano XY.

Projetando P no plano XY:

$$R = \sqrt{Px^2 + Py^2} \qquad \textbf{(4.4)}$$

$$tg(\theta 1) = \frac{Py}{Px} \qquad \textbf{(4.5)}$$

1) *Se*: $Px = 0 \wedge Py \geq 0 \rightarrow \theta 1_{calc} = \frac{\pi}{2}$

$Px = 0 \wedge Py < 0 \rightarrow \theta 1_{calc} = -\frac{\pi}{2}$

Senão:

$$A = tg^{-1}\left(\frac{Py}{Px}\right)$$

$SE\ (A \geq 0) \wedge Py \geq 0 \rightarrow \theta 1_{calc} = A$

$SE\ (A > 0) \wedge Py < 0 \rightarrow \theta 1_{calc} = (A - \pi)$

$SE\ (A < 0) \wedge Py > 0 \rightarrow \theta 1_{calc} = (\pi + A)$

$SE\ (A < 0) \wedge Py < 0 \rightarrow \theta 1_{calc} = A$

2) *Se*: $(Base.\theta 1_{calc}) \geq 0 \rightarrow \theta 1 = \theta 1_{calc}$

Senão:

Se $\theta 1_{calc} > 0 \rightarrow \theta 1 = \theta 1_{calc} - \pi$

Se $\theta 1_{calc} < 0 \rightarrow \theta 1 = \theta 1_{calc} + \pi$

4.3.2 ☞ Solução Para a Junta 2

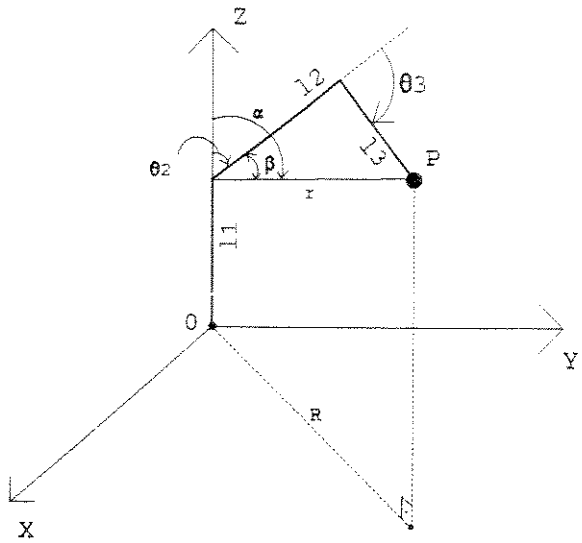


Figura 4.4 Projeção de P em XYZ.

$$r = \sqrt{Px^2 + Py^2 + (Pz - l1)^2} \tag{4.6}$$

$$R = \sqrt{Px^2 + Py^2} \tag{4.7}$$

$$tg(\alpha) = \frac{R}{(Pz - l1)} \quad \rightarrow \quad \alpha = tg^{-1}\left(\frac{R}{(Pz - l1)}\right) \tag{4.8}$$

Se: $R = 0$

$$\wedge \ (Pz - l1) < 0 \quad \rightarrow \alpha = -\frac{\pi}{2}$$

$$\wedge \ (Pz - l1) \geq 0 \quad \rightarrow \alpha = \frac{\pi}{2}$$

Observando a **figura 4.4** e utilizando a lei dos cossenos:

$$l3^2 = l2^2 + r^2 - 2.l2.r.\cos(\beta) \tag{4.9}$$

$$\cos(\beta) = \frac{l2^2 + r^2 - l3^2}{2.l2.r} \tag{4.10}$$

$$\beta = \cos^{-1}\left(\frac{l2^2 + r^2 - l3^2}{2.l2.r}\right) \tag{4.11}$$

Com β e α calculados, e com os indicadores BASE e COTOVELO configurados:

$$\theta 2_{calc} = \alpha - (Cotovelo).\beta \tag{4.12}$$

Se: $(base.\theta 1_{calc}) \geq 0 \quad \rightarrow \theta 2 = \theta 2_{calc}$

Senão: $\theta 2 = -(\pi + \theta 2_{calc})$

4.3.3 ☞ Solução Para a Junta 3

Observando novamente a **figura 4.4**, e utilizando a lei dos cossenos:

$$r^2 = l2^2 + l3^2 - 2.l2.l3.\cos(\pi - \theta 3) \tag{4.13}$$

como $\cos(\pi - \theta) = -\cos(\theta)$, então:

$$\cos(\theta 3) = \frac{r^2 - l2^2 - l3^2}{2.l2.l3} \tag{4.14}$$

pela lei dos senos:

$$\frac{r}{sen(\pi - \theta 3)} = \frac{l3}{sen(\beta)} \qquad (4.15)$$

mas $sen(\pi - \theta) = sen(\theta)$

logo,

$$sen(\theta 3) = \frac{r.sen(\beta)}{l3} \qquad (4.16)$$

e

$$tg(\theta 3) = \frac{2.l2.r.sen(\beta)}{r^2 - l2^2 - l3^2} \qquad (4.17)$$

1) $Se \quad r^2 = l2^2 + l3^2 \quad \rightarrow \theta 3_{calc} = (Cotovelo).\frac{\pi}{2}$

2) $\theta 3_{calc} = tg^{-1}\left(\frac{2.l2.r.sen(\beta)}{r^2 - l2^2 - l3^2}\right).(Cotovelo)$

Se: $(Base.\theta 1_{calc}) \geq 0 \quad \rightarrow \theta 3 = \theta 3_{calc}$

Senão:

Se: $\theta 3_{calc} > 0 \quad \rightarrow \theta 3 = \theta 3_{calc} - \pi$

Se: $\theta 3_{calc} < 0 \quad \rightarrow \theta 3 = \pi - \theta 3_{calc}$

4.4 ☞ RESULTADOS EXPERIMENTAIS, OBSERVAÇÕES E CONCLUSÕES

A seguir apresentamos uma seqüência de gráficos onde se pode verificar a eficiência dos três modos da técnica de busca heurística propostos no **Capítulo III** para a perseguição e rastreio de uma trajetória especificada. Antes de apresentá-los, primeiramente descrevemos as condições utilizadas nas simulações que geraram tais gráficos.

Nesta primeira seqüência de gráficos que estão mostrados nas **figuras de 4.5 a 4.23** mostramos os resultados referentes a experimentos supondo que o robô **JECAII** é posto para rastrear uma reta com as seguintes equações paramétricas:

$$X = k$$

$$Y = 2.k \qquad (4.18)$$

$$Z = 500$$

Parametrizada por k, tal que $0 \leq k \leq 500$, e discretizada com passo de discretização $\Delta k = 1$.

Exigimos que a precisão de aproximação a cada ponto pertencente a esta reta discretizada seja menor ou igual a 1mm. E como primeiramente nossa intenção é mostrar as características da evolução no rastreamento espacial, apresentamos os gráficos de posicionamento em *3D* gerados pela tela gráfica básica de simulação do pacote de *software* desenvolvido para esta finalidade, deixando para fazer comentários sobre a evolução das velocidades e acelerações de juntas, bem como períodos de amostragem da malha de controle de trajetórias, mais à frente quando apresentamos os resultados de experimentos utilizando a técnica proposta para o controle de trajetórias mais complexas.

Nestes primeiros gráficos é interessante fixar a atenção ao detalhe muito importante que pode ser constatado sobre a independência da perseguição da trajetória com respeito a configuração inicial na qual se encontra o robô antes de começar os movimentos de rastreamento em todos os três modos propostos da técnica.

Nos gráficos apresentados em *3D*, propositalmente deixamos apresentadas também as retas que representam o robô, quando ele atinge o último ponto pertencente a trajetória. Em todas as simulações consideramos que cada junta do robô possui limitações de final de curso, e porisso quando a trajetória atravessa uma região fora do volume de trabalho, ela é apresentada mas o robô não persegue estes pontos que a compõem em tal região, limitamos todas as juntas com um curso que está compreendido entre -170° e +170°.

Note que próximo a X=Y=0, o robô não pode atingir os pontos pertencentes a reta que impuzemos para ele rastrear, e que o primeiro ponto rastreado da trajetória está um pouco afastado do eixo Z por esta razão. O próprio algoritmo que supervisiona os movimentos faz a verificação se os pontos gerados por esta equação estão dentro do volume de trabalho do robô, e gera níveis de ajustes adequados, com as devidas mensagens para o projetista das trajetórias.

As **figuras de 4.5 a 4.12** são referentes a aplicação do primeiro modo da técnica, onde somente faz-se movimentos elementares das juntas movendo uma em cada período de amostragem da malha de controle de trajetórias, e com passos fixos, as **figuras de 4.13 a 4.17** referem-se a aplicação do segundo modo, onde são feitos movimentos combinados das juntas em um mesmo período, mas também com passos fixos em cada junta, e as **figuras de 4.18 a 4.22** referem-se a aplicação do terceiro modo da técnica, onde são feitos movimentos combinados das juntas com passos variáveis segundo o critério de otimalidade apresentado anteriormente.

Ainda a **figura 4.23** mostra a evolução do rastreamento da mesma trajetória utilizando o terceiro modo da técnica, mas supondo que existem perturbações que causam erro de respostas dos Escravos de juntas.

Para efeitos de fornecer dados que permitam fazer boas comparações, realizamos 5 experimentos para cada aplicação de um dos modos da técnica, supondo em cada um desses experimentos condições iniciais idênticas da estrutura mecânica, para que cada modo realizasse o controle do rastreamento. Por exemplo, as **figuras 4.5, 4.13, e 4.17** apresentam os resultados para uma mesma condição inicial do robô, e respectivamente, quando são utilizados os modos 1, 2, e 3 da técnica para o controle do rastreamento, e assim sucessivamente. Os gráficos apresentados nas **figuras 4.6, 4.7, e 4.8** estão postos com finalidade ilustrativa, apresentando as projeções nos planos XY, XZ, e YZ do mesmo rastreamento mostrado no gráfico da **figura 4.5**, que também pode ser uma ferramenta útil para análise que o mesmo pacote de *software* permite visualizar. Os tempos mostrados nos cantos superiores direitos dos gráficos são os tempos decorridos até os finais dos rastreamentos.

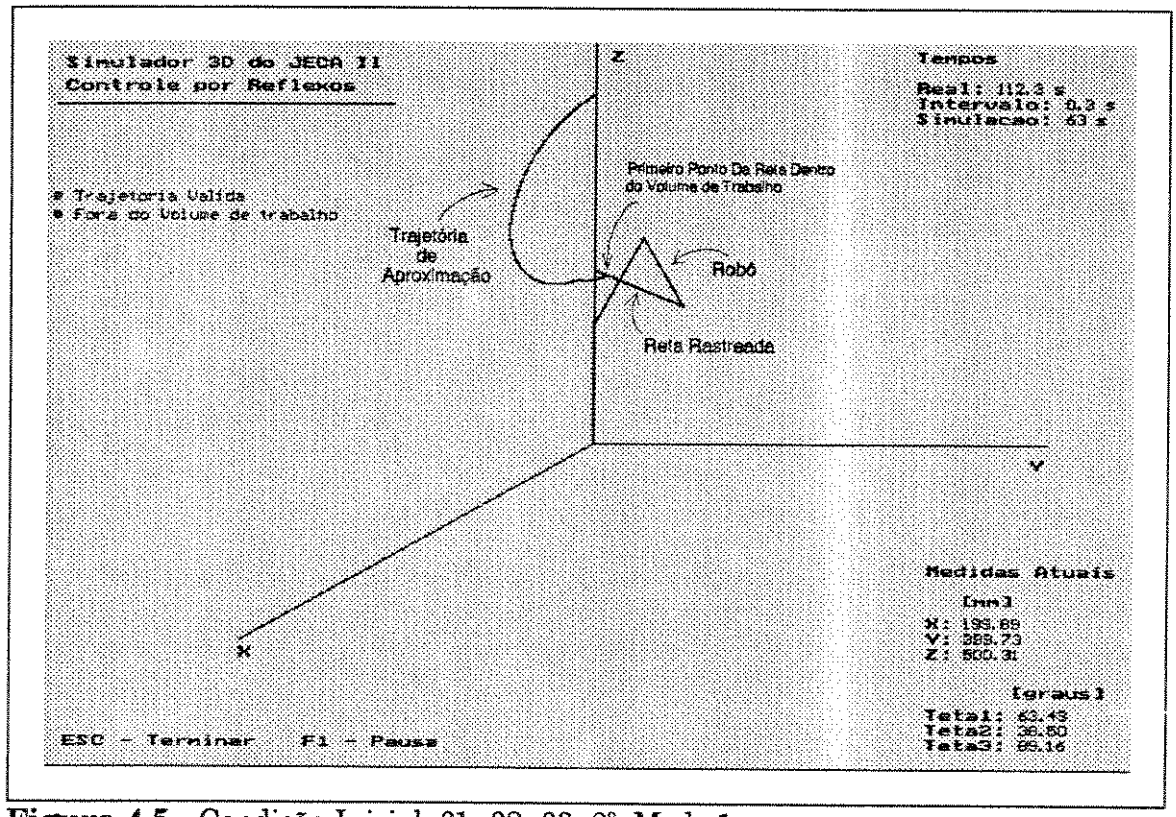


Figura 4.5 - Condição Inicial: $\theta1=\theta2=\theta3=0°$. Modo 1.

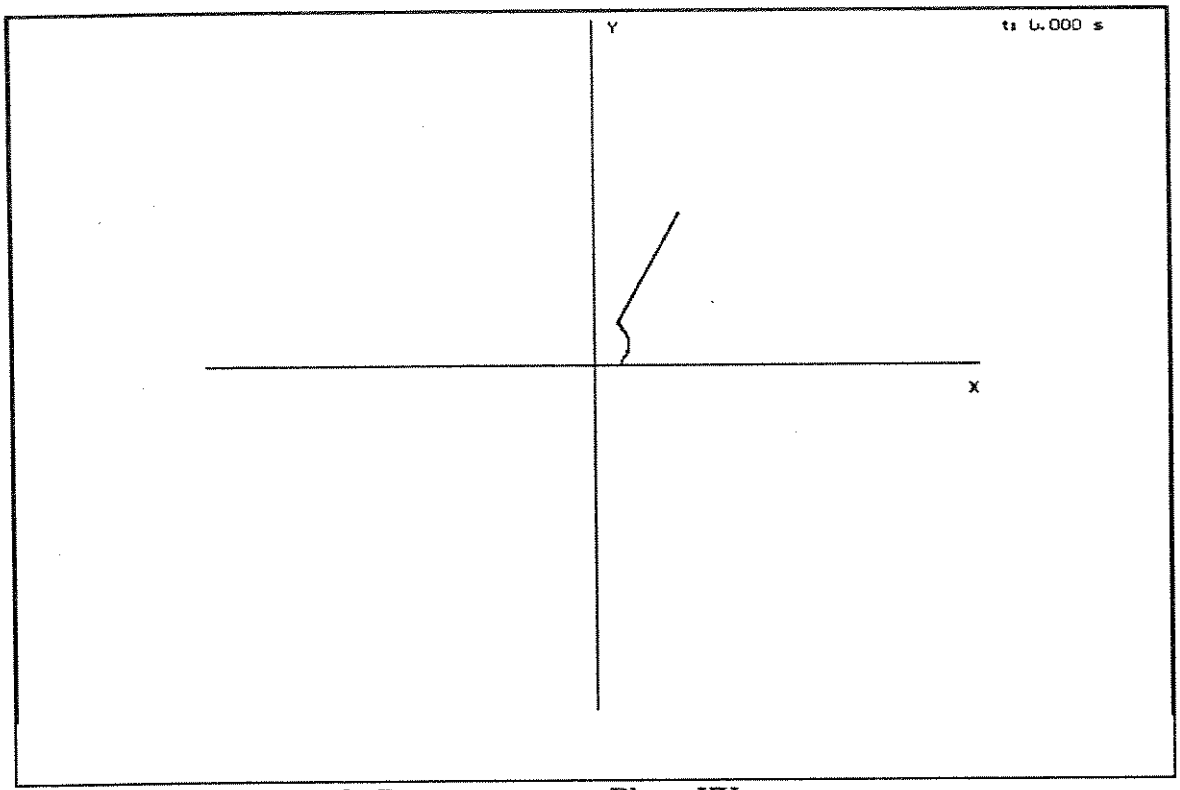


Figura 4.6 - Projeção do Rasteamento no Plano XY.

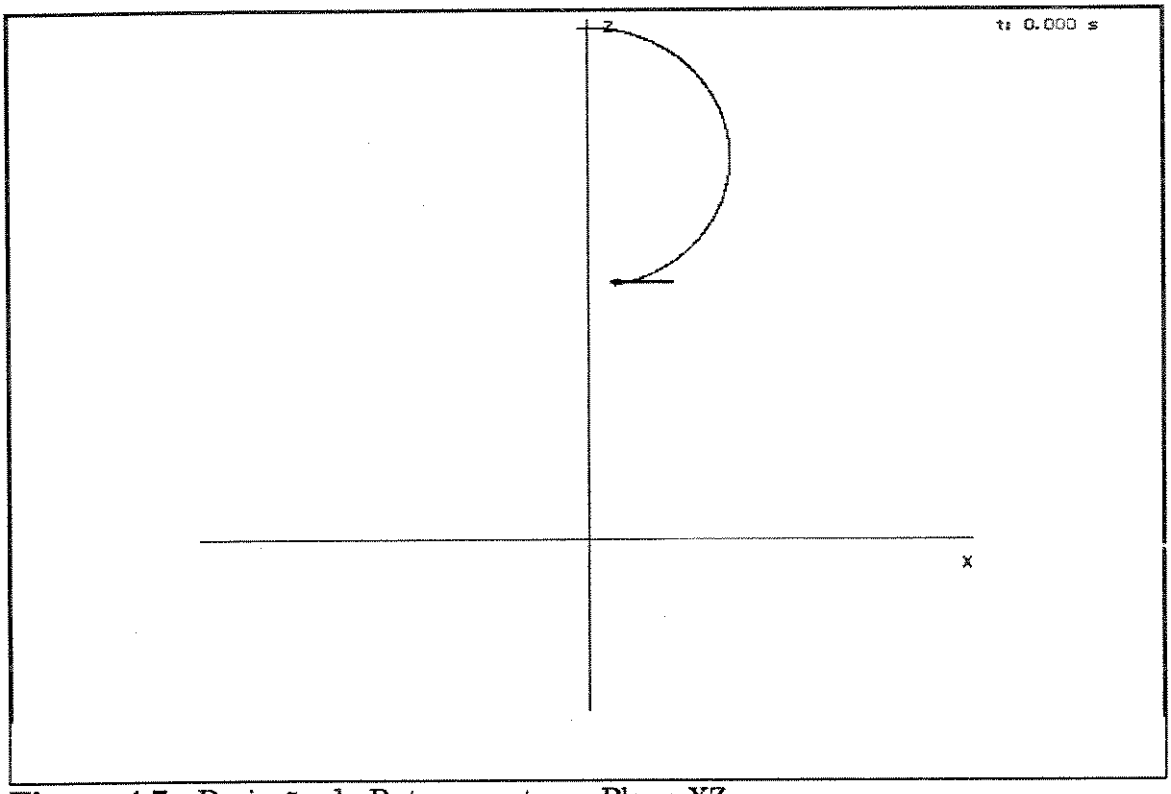


Figura 4.7 - Projeção do Ratreamento no Plano XZ.

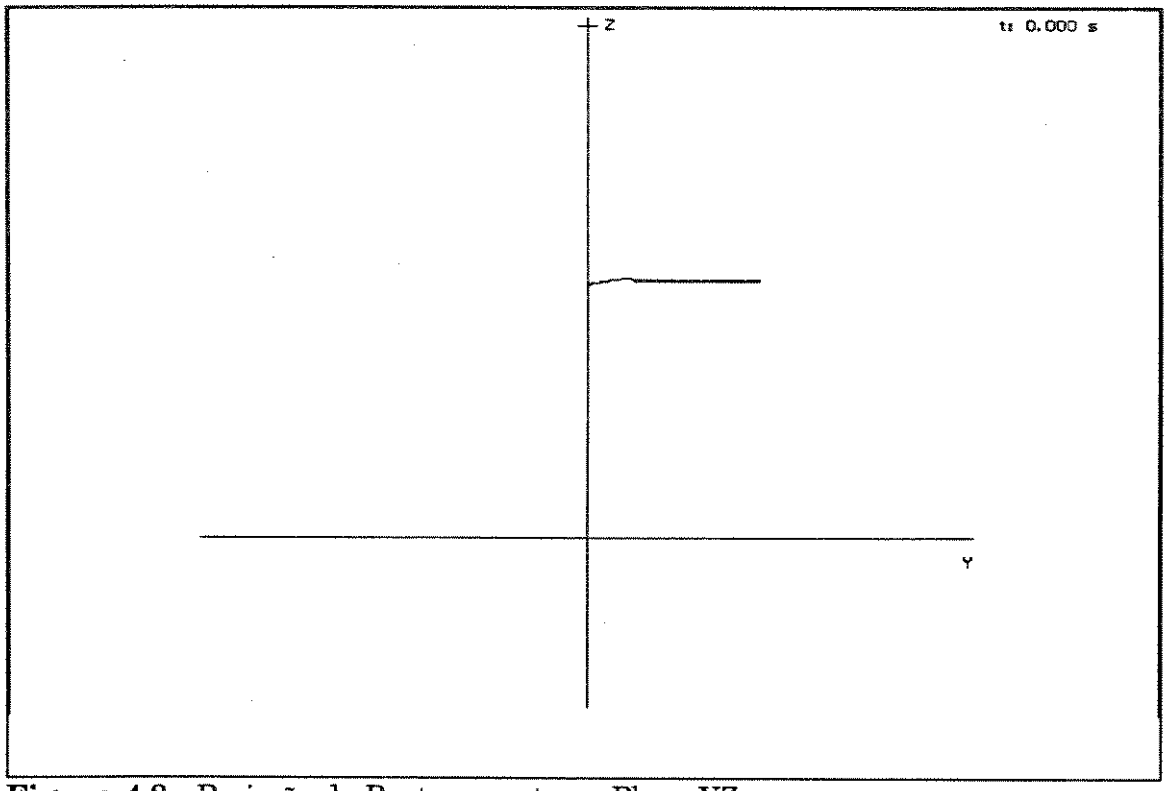


Figura 4.8 - Projeção do Rastreamento no Plano YZ.

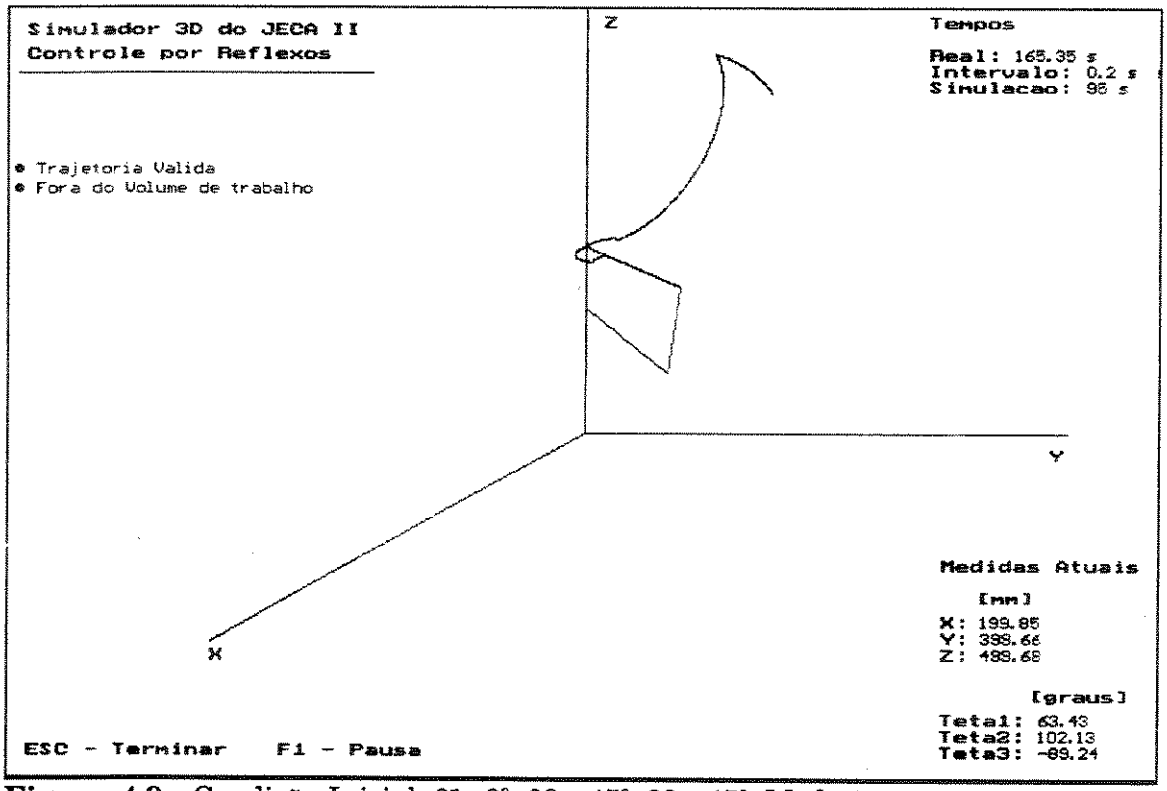


Figura 4.9 - Condição Inicial: $\theta 1=0°$, $\theta 2=-45°$, $\theta 3=-45°$. Modo 1.

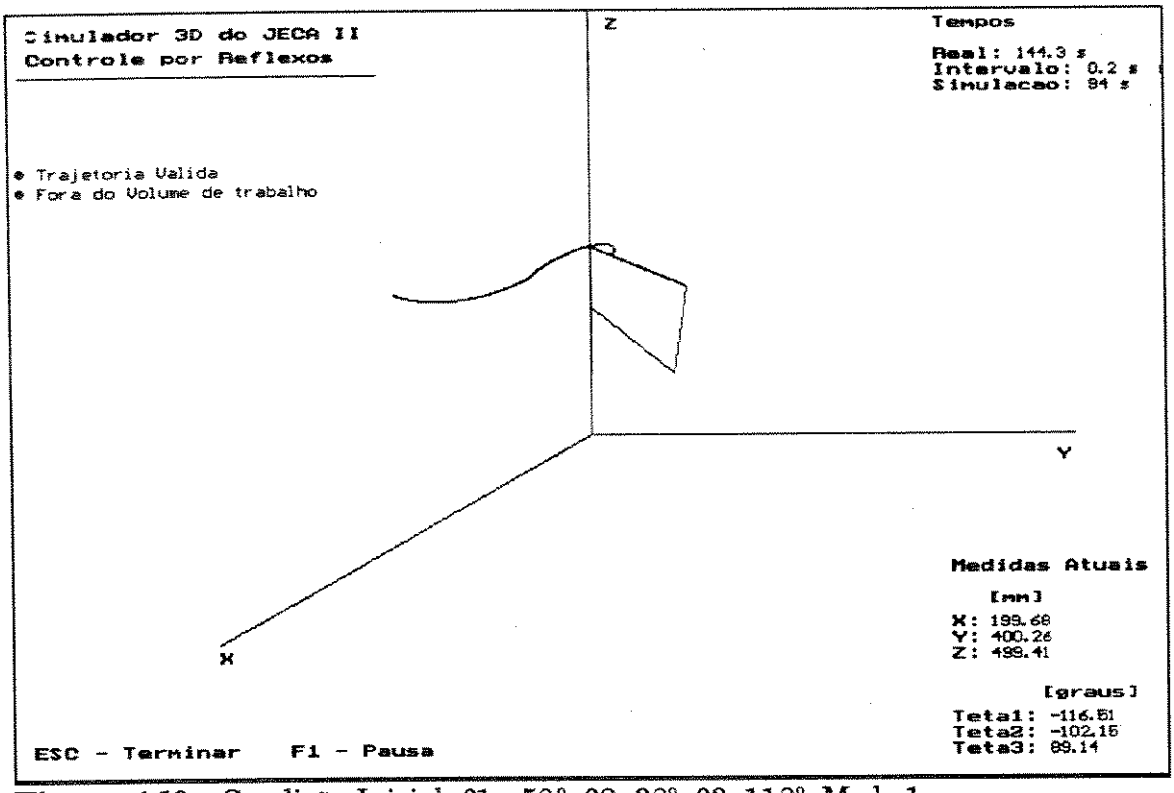


Figura 4.10 - Condição Inicial: θ1=-50°, θ2=30°, θ3=110°. Modo 1.

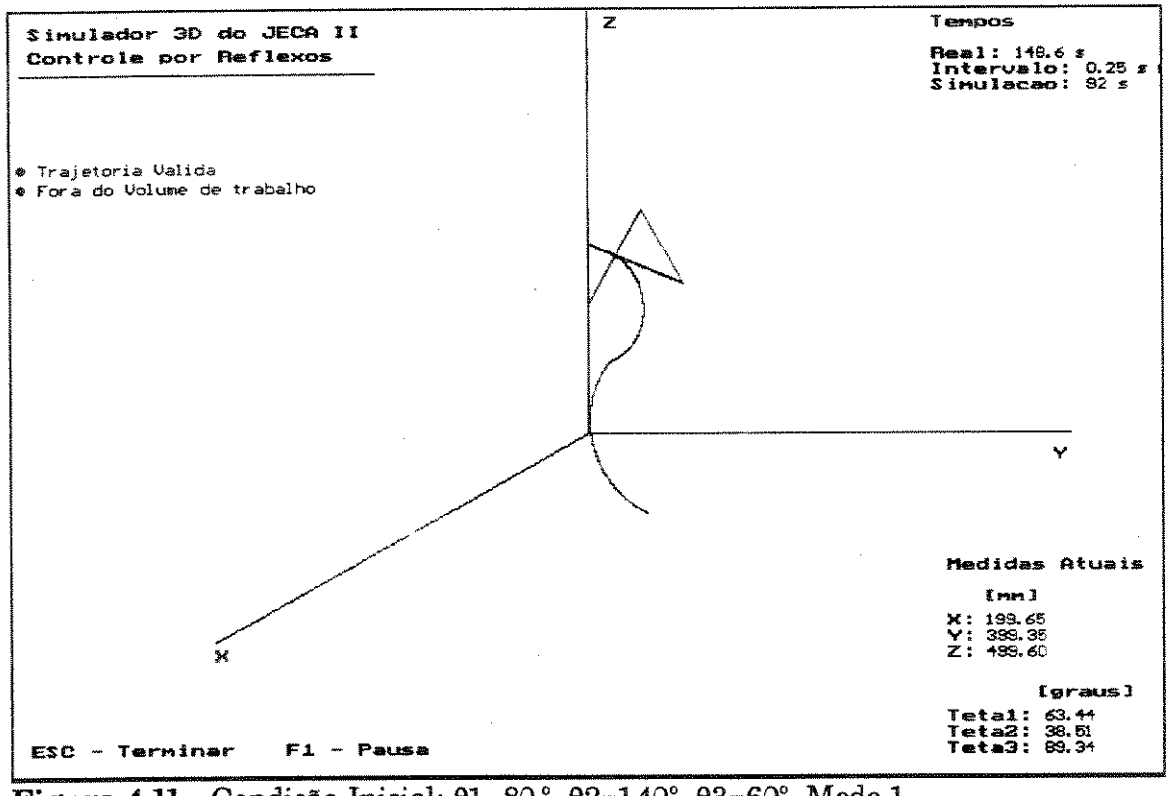


Figura 4.11 - Condição Inicial: θ1=80°, θ2=140°, θ3=60°. Modo 1.

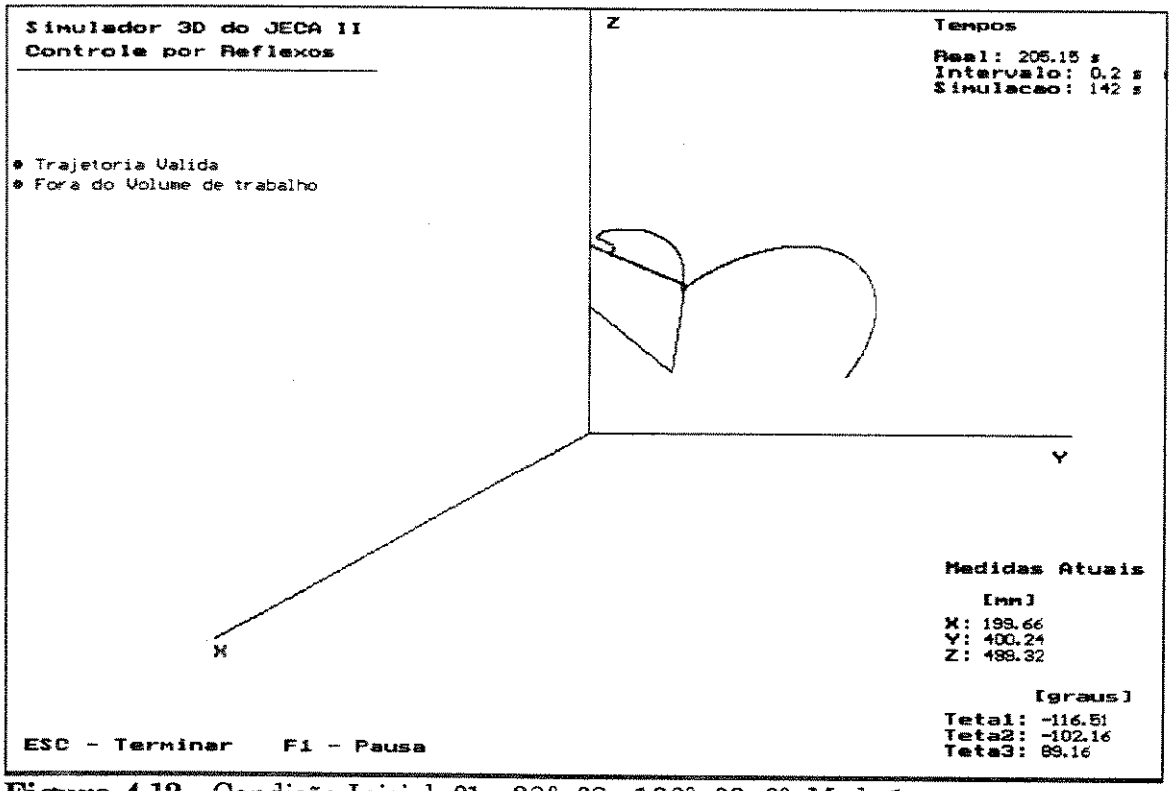


Figura 4.12 - Condição Inicial: θ1=-30°, θ2=-130°, θ3=0°. Modo 1.

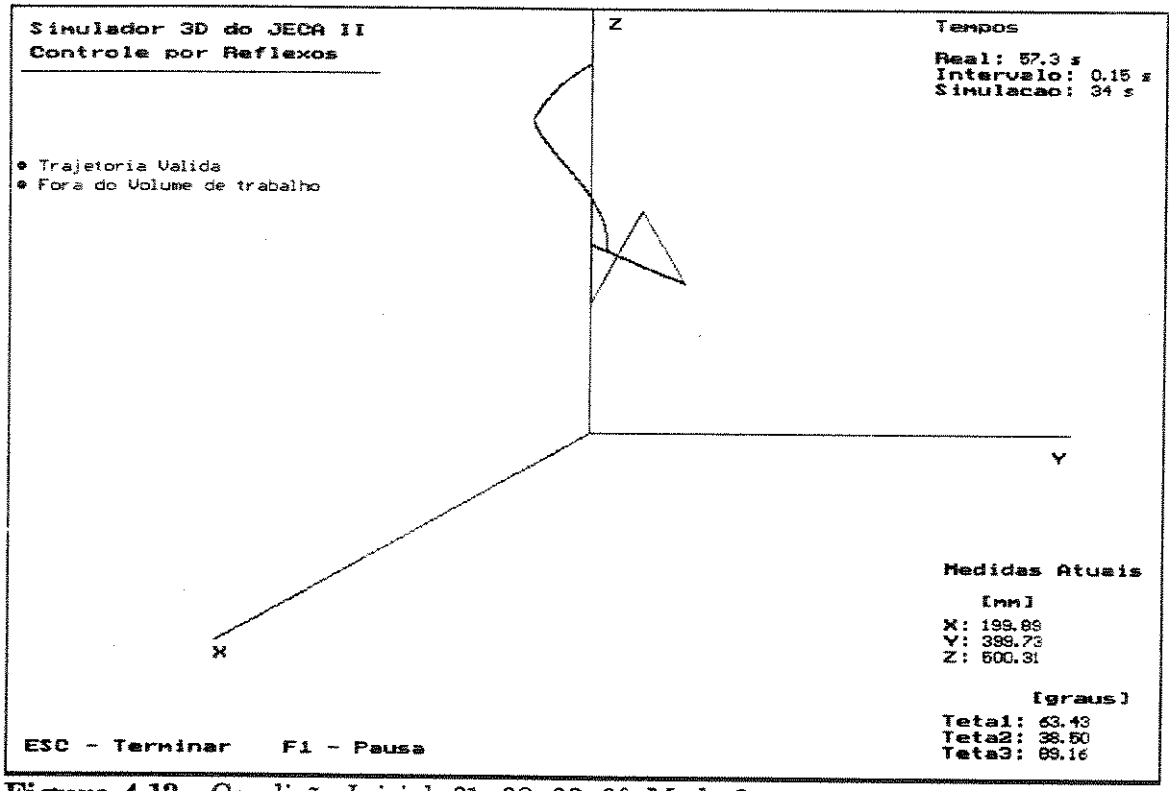


Figura 4.13 - Condição Inicial: θ1=θ2=θ3=0°. Modo 2.

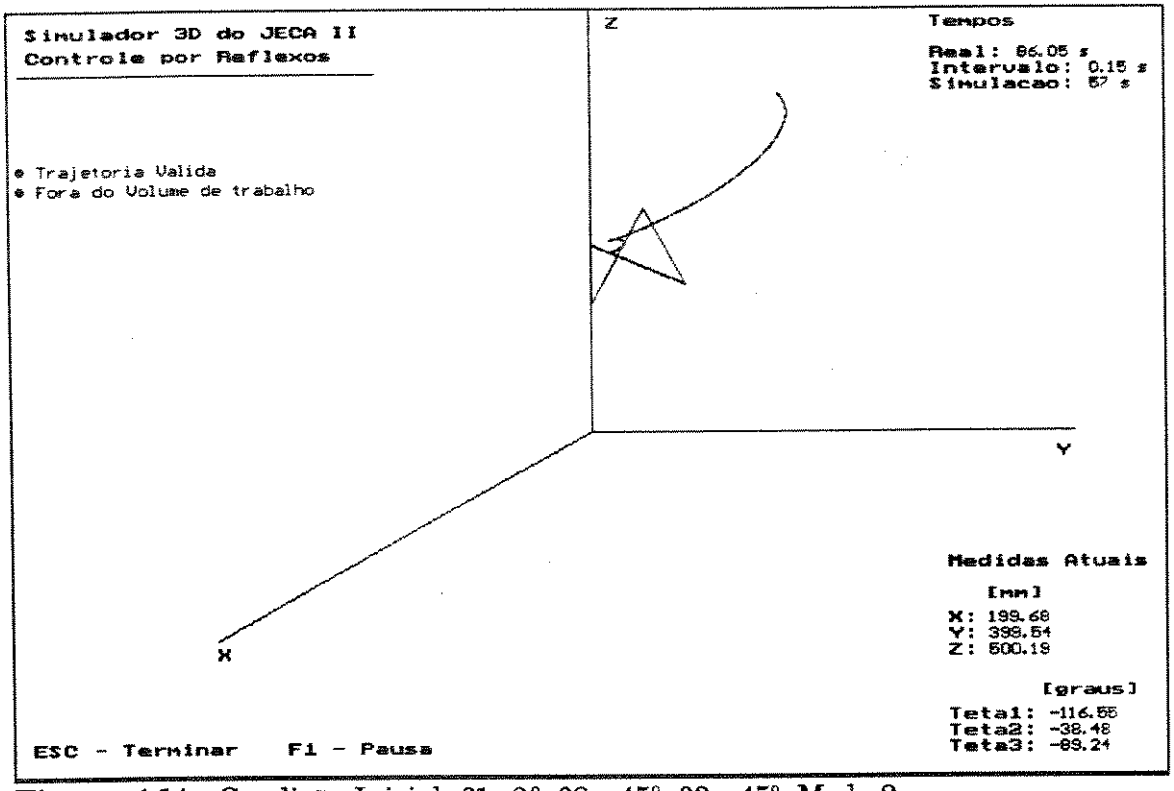


Figura 4.14 - Condição Inicial: θ1=0°, θ2=-45°, θ3=-45°. Modo 2.

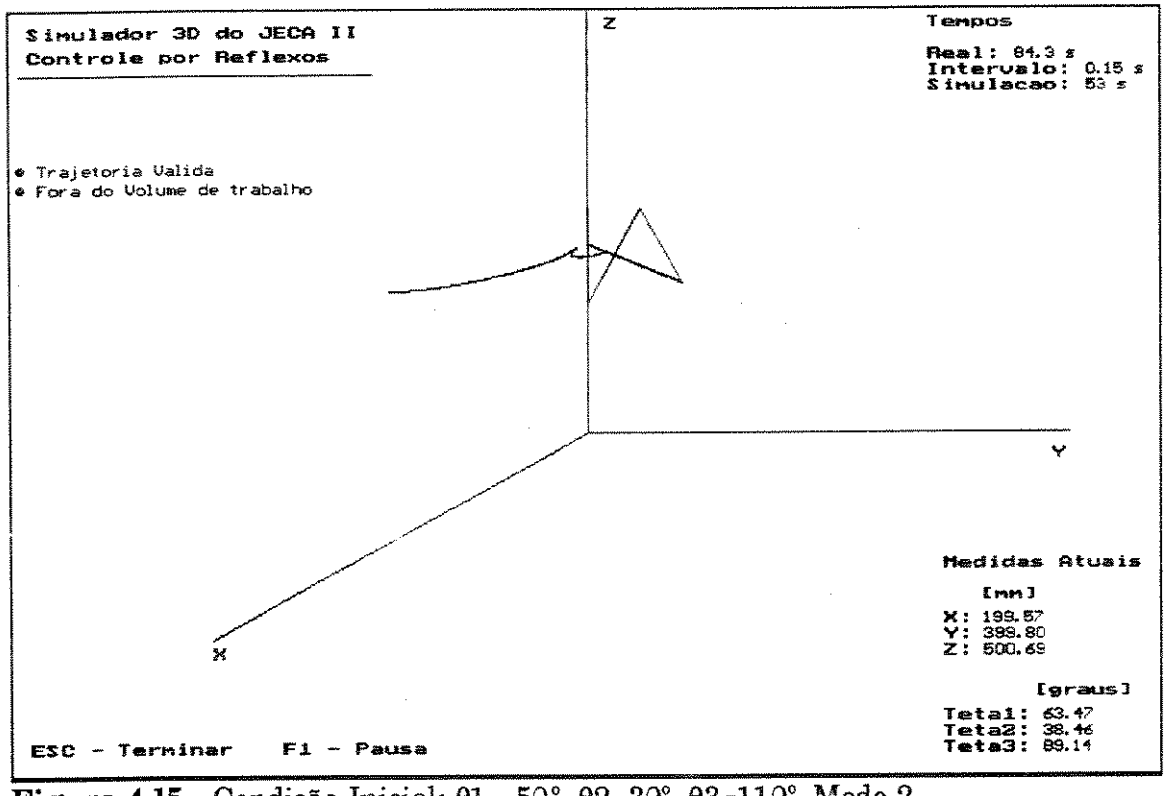


Figura 4.15 - Condição Inicial: θ1=-50°, θ2=30°, θ3=110°. Modo 2.

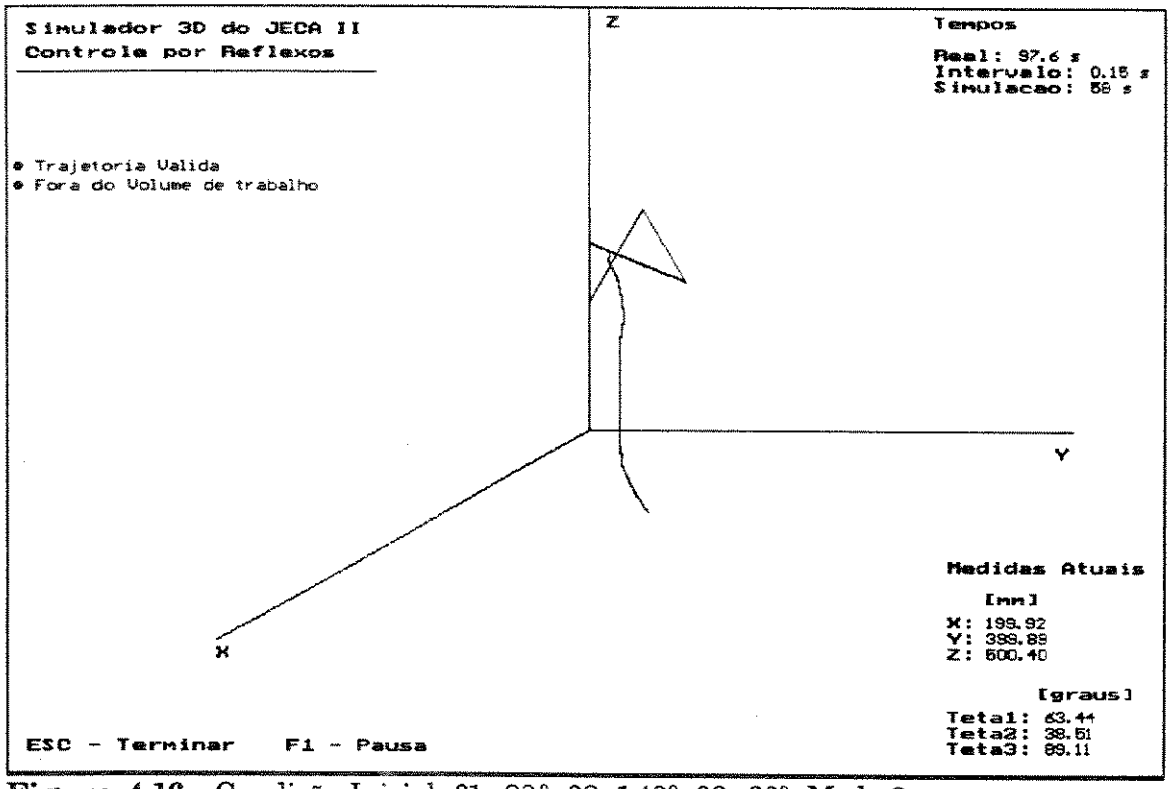


Figura 4.16 - Condição Inicial: θ1=80°, θ2=140°, θ3=60°. Modo 2.

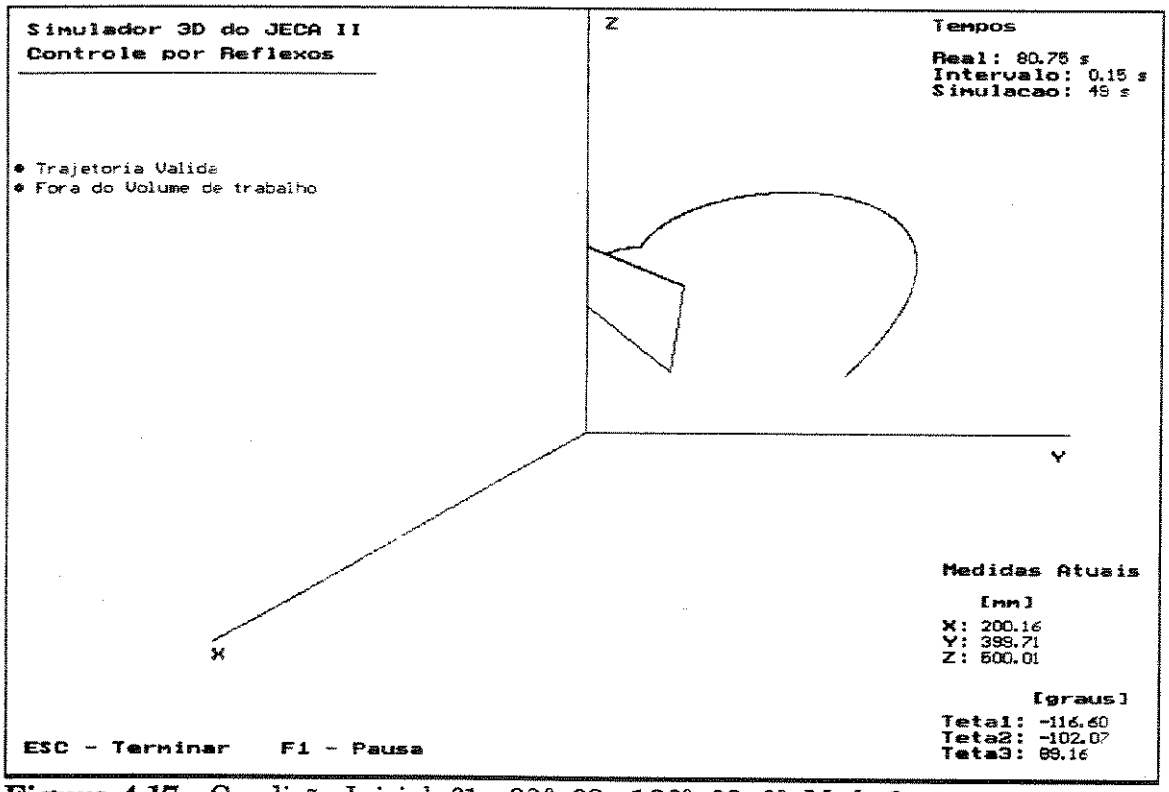


Figura 4.17 - Condição Inicial: θ1=-30°, θ2=-130°, θ3=0°. Modo 2.

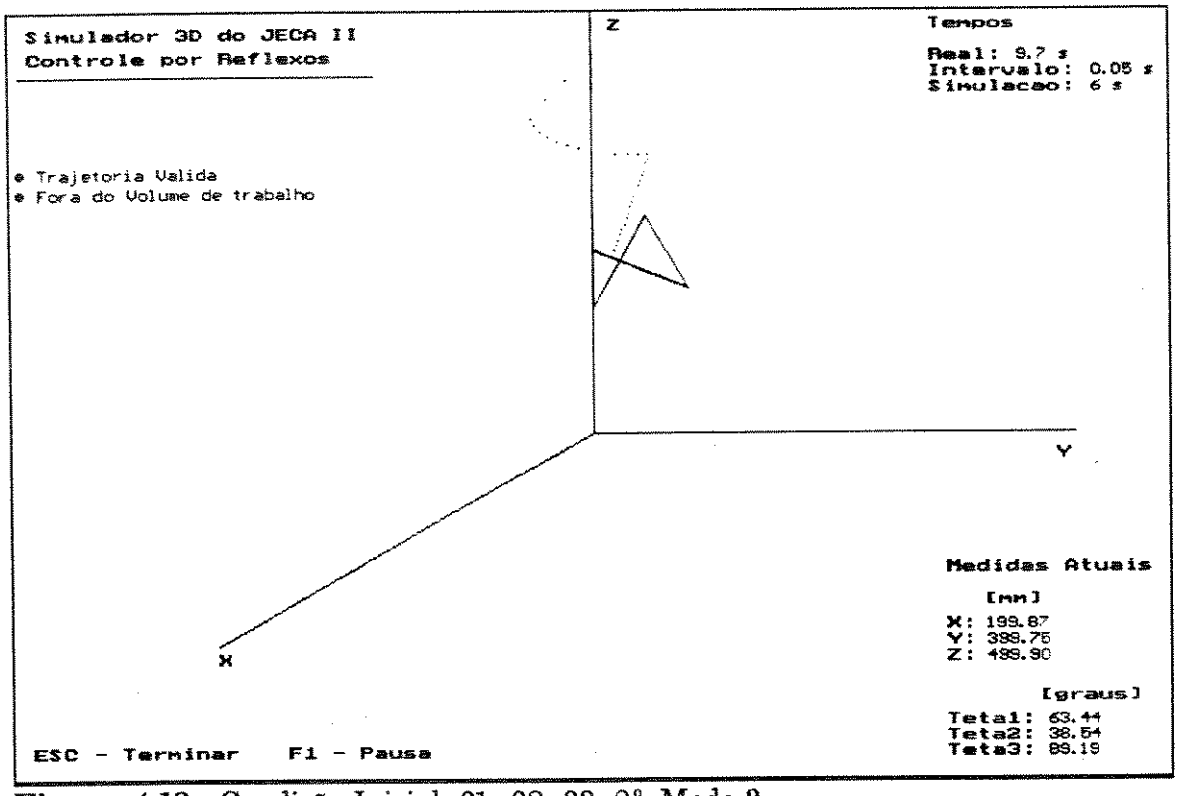


Figura 4.18 - Condição Inicial: θ1=θ2=θ3=0°. Modo 3.

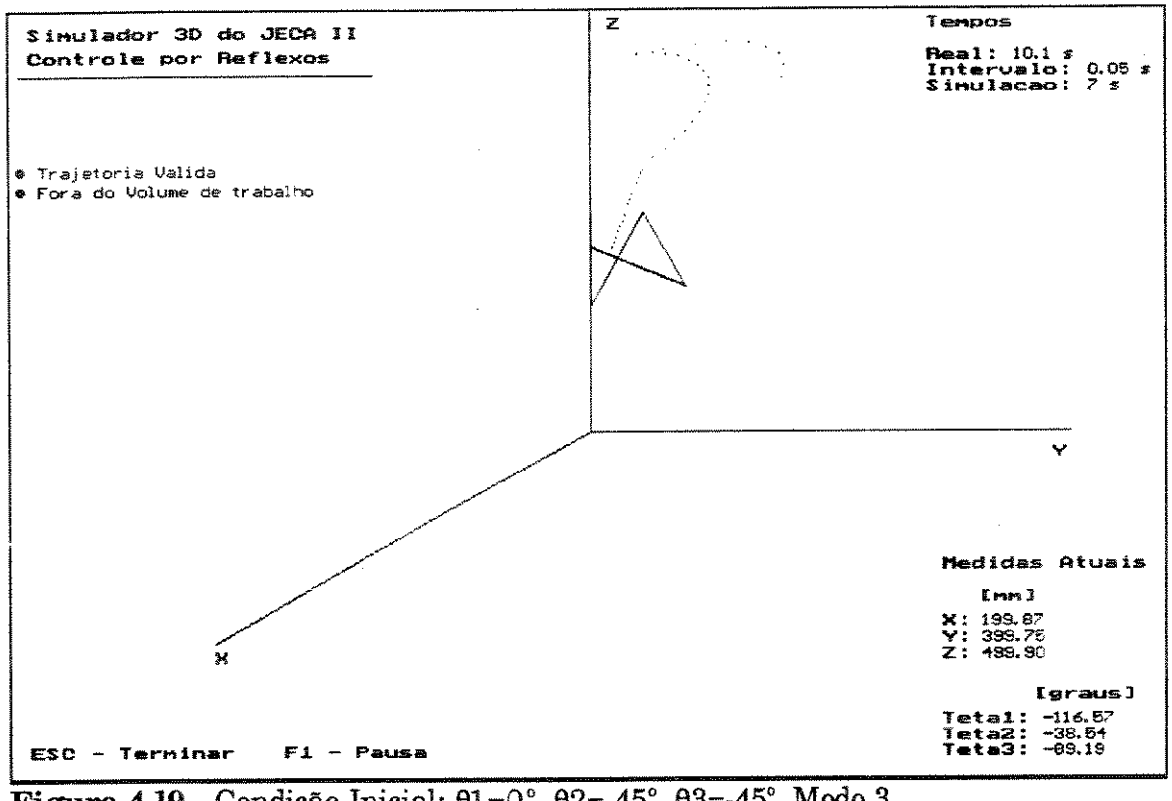


Figura 4.19 - Condição Inicial: θ1=0°, θ2=-45°, θ3=-45°. Modo 3.

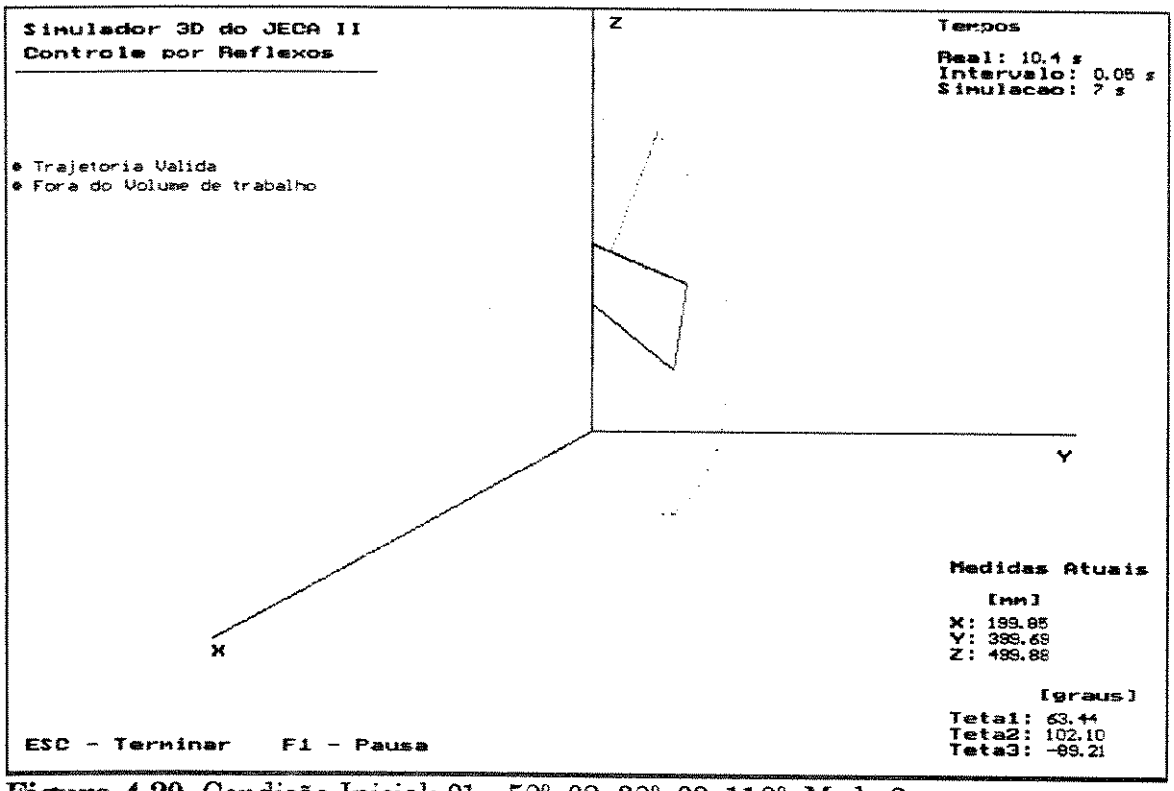


Figura 4.20. Condição Inicial: θ1=-50°, θ2=30°, θ3=110°. Modo 3.

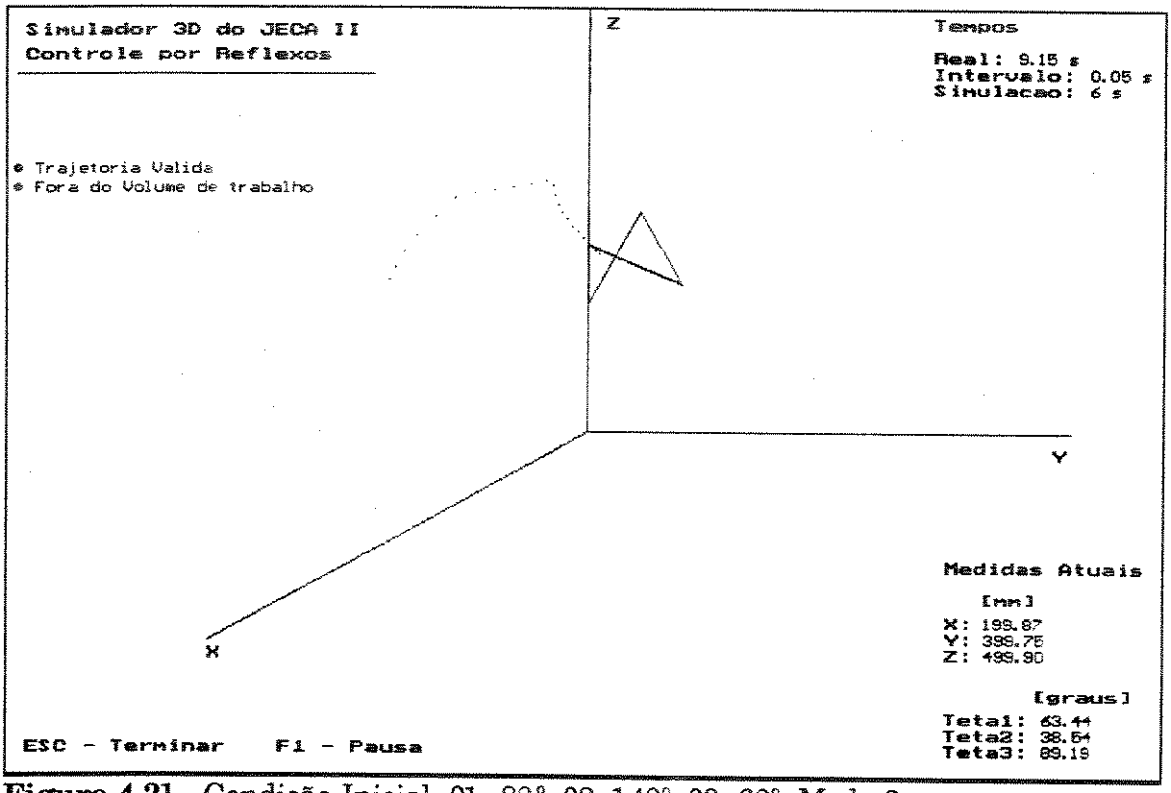


Figura 4.21 - Condição Inicial: θ1=80°, θ2=140°, θ3=60°. Modo 3.

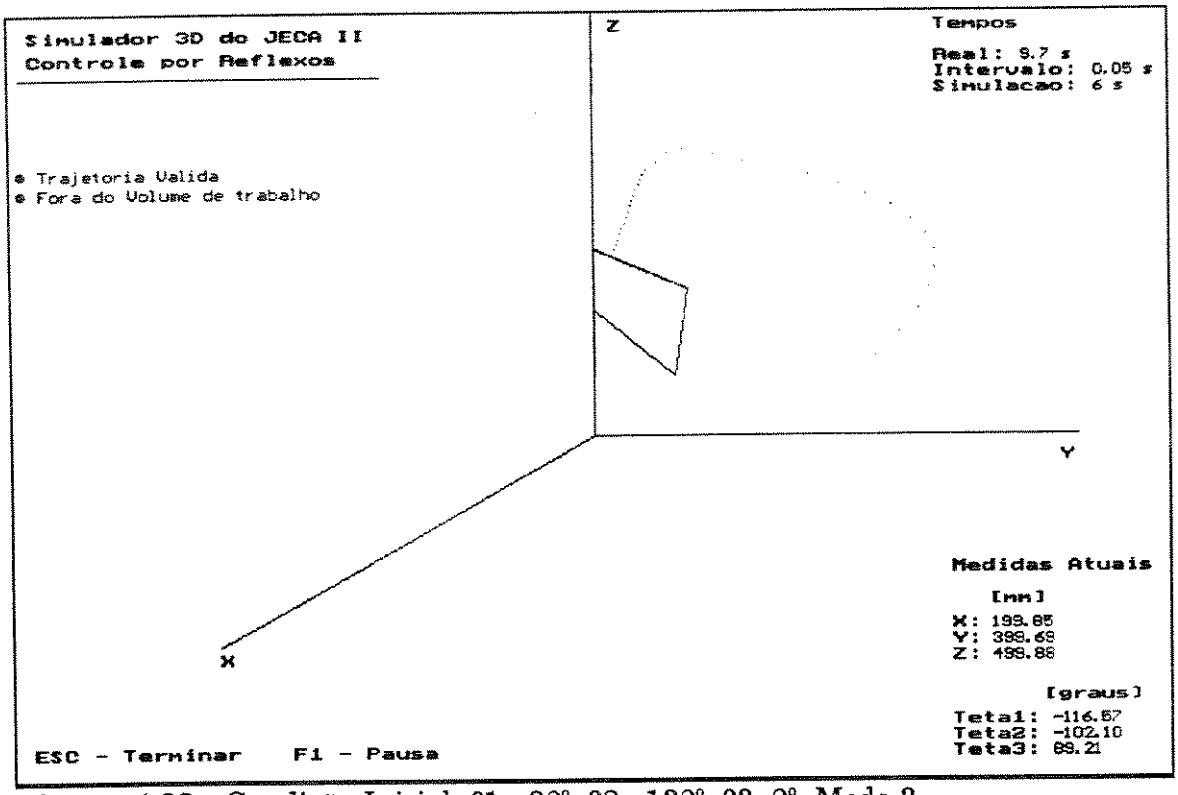


Figura 4.22 - Condição Inicial: θ1=-30°, θ2=-130°, θ3=0°. Modo 3.

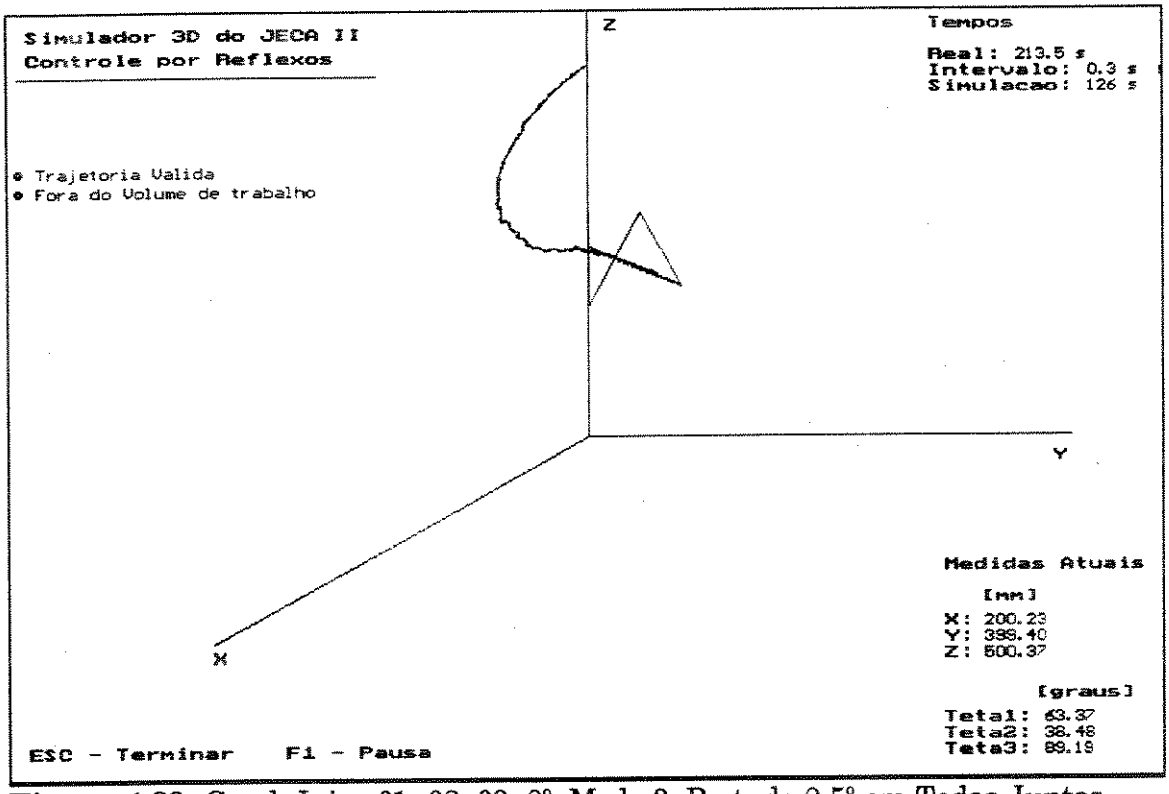


Figura 4.23. Cond. Inic.: θ1=θ2=θ3=0°. Modo 3. Pert. de 0,5° em Todas Juntas.

Os próximos gráficos que apresentamos têm a finalidade de mostrar os resultados obtidos com a aplicação da técnica, em seus três modos, quando submetemos o robô **JECAII**, em simulação, a rastrear uma circunferência cujas equações paramétricas referidas ao seu sistema de coordenadas de base são:

$$X = 200 + 100.sen(k)$$

$$Y = 200 + 100.\cos(k) \qquad \textbf{(4.19)}$$

$$Z = 400$$

com medidas X,Y,Z em mm, e k em rd.

Ou seja, uma circunferência contida em um plano paralelo ao plano XY deslocado de 400mm da origem, na direção positiva de Z, centrada no ponto $C_{(X,Y,Z)}$=(200,200,400) e raio 100mm.

Discretizamos estas equações fazendo $0 \leq k \leq 2\pi$, com passo de discretização para k de $\pi/180$rd. E fazemos a exigência para que o extremo do elo do seu terceiro grau de liberdade atinja cada ponto desta trajetória discretizada com um erro, em distância, menor ou igual a 1mm.

Estaremos também supondo que o período de amostragem para a malha de controle de trajetórias seja de 50ms, e primeiramente, que os Escravos atuem de modo perfeito, isto é, façam que cada respectiva junta siga exatamente os níveis de ajustes gerados pelo Mestre quando executa as rotinas de controle baseadas nos três modos da técnica.

As velocidades de juntas são limitadas em 1,57rd/s (90°/s), que segundo a construção do **JECAII**, são perfeitamente alcançáveis. E para que as comparações de desempenho sejam mais significativas, iremos sempre antes de cada experimento posicionar o robô com θ1=+60°, θ2=+60°, e θ3=+90° que correspondem ao ponto $P_{(X,Y,Z)}$=(235,706;408,253;323,494), e iniciaremos estes experimentos com ambas as juntas postas em velocidades zero.

Utilizando o algoritmo para o primeiro modo, onde somente é realizado o movimento de uma junta em cada período de amostragem, com módulo de movimento angular constante para cada junta, e calculando estes módulos em função da precisão exigida (1mm), para que um único movimento angular nunca cause um deslocamento do extremo do terceiro Elo, de amplitude maior que esta precisão em um único período de amostragem, obtivemos:

módulo 1= 0,09° ou 0,0016rd

módulo 2= 0,09° ou 0,0016rd

módulo 3= 0,23° ou 0,004rd

Observemos que para esta exigência de precisão, e para este período de amostragem, as velocidades permitidas para as juntas são relativamente muito baixas:

$$Vmáx_1 = 0{,}032 rd/s$$

$$Vmáx_2 = 0{,}032 rd/s \qquad (4.20)$$

$$Vmáx_3 = 0{,}08 rd/s$$

● *Portanto, o robô necessariamente irá mover-se muito lentamente. Avaliados os resultados de simulações, a velocidade tangencial média seria aproximadamente 1,5cm/s, mas a necessidade de tempo computacional para este modo é muito pequena, e dependendo do computador utilizado como Mestre, poder-se-ia realizar movimentos muito mais rápidos, fazendo variar o período de amostragem, etc..., ou ainda, se a intenção fosse fazer com que o sistema "aprenda" em função dos dados obtidos em movimentos já realizados, numa segunda passada, poder-se-ia acelerar muito este processo, conforme explicamos anteriormente, fazendo associações convenientes destes movimentos elementares em um mesmo período de amostragem.*

Os gráficos mostrados nas **figuras 4.24, 4.25, 4.26, e 4.27** mostram respectivamente a evolução do rastreamento da trajetória de posição cartesiana em *3D*, a projeção deste rastreamento no plano XY, a projeção no plano XZ, e a projeção no plano YZ.

A partir da posição inicial, o robô sob as condições postas anteriormente, atinge o primeiro ponto da trajetória em 17,55s, e finaliza os movimentos em 93,45s. A janela apresentada na mesma **figura 4.24**, mostra o momento em que o tempo decorrido é de 40s, com uma amplificação de 10 vezes relativa a região próxima ao cursor, para que possa-se ver que os movimentos sofrem pequenas oscilações devido às características deste modo de aplicar a técnica. Lembramos também que estas oscilações podem evidentemente ser prejudiciais, e causarem vibrações na estrutura mecânica. Este é um assunto que ainda deveremos estudar com maior profundidade.

● *O importante a destacar neste ponto, é que o método manteve a convergência no rastreamento da trajetória até sua total perseguição.*

A **figura 4.28** mostra a evolução da velocidade tangencial do extremo final do terceiro Elo do robô durante todo o rastreamento. Note que suas características não são muito interessantes do ponto de vista de suavidade de movimentos; pelas características deste modo da técnica sempre haverão oscilações, embora possam ser de pequena amplitude (neste caso, a máxima variação é de 1,44cm/s em 50ms, ou seja, aceleração de 0,29m/s²). A janela colocada em torno do cursor mostra uma amplificação de 5 vezes, em sua proximidade, para que possa-se verificar melhor como são estas variações.

● → *Indicador de Conclusão Parcial.*

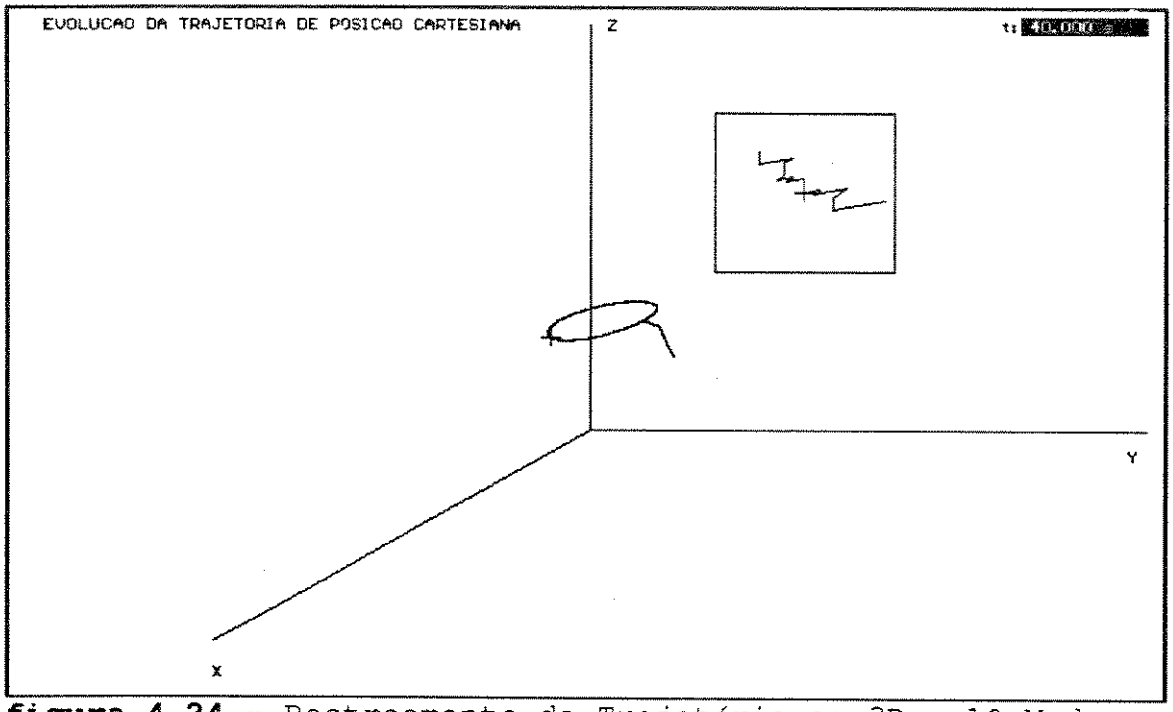


figura 4.24 - Rastreamento da Trajetória em 3D - 1º Modo.

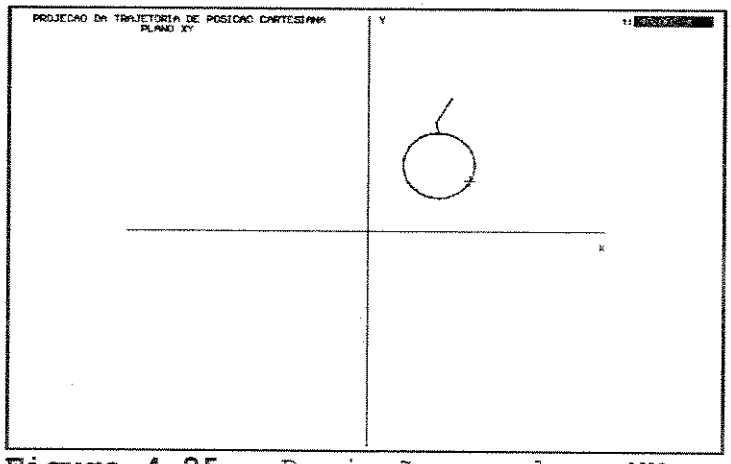


Figura 4.25 - Projeção no plano XY.

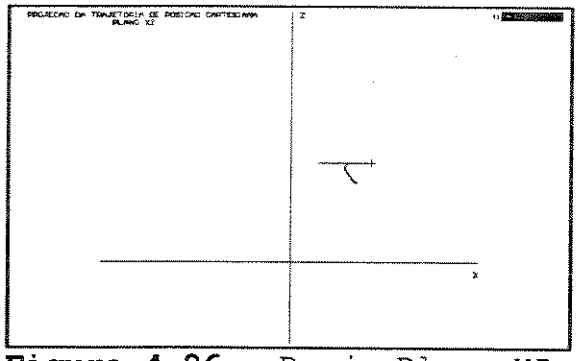


Figura 4.26 - Proj. Plano XZ.

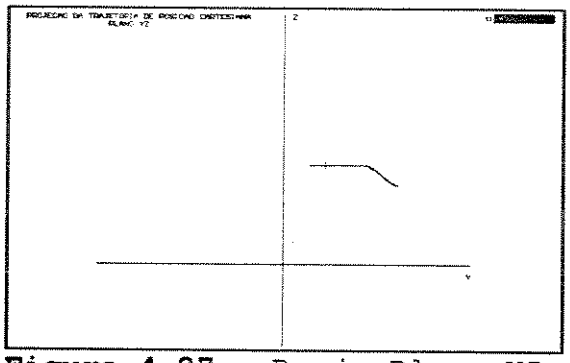


Figura 4.27 - Proj. Plano YZ.

Entre cada período, a evolução foi considerada linear (Interpolamos uma reta entre cada dois pontos consecutivos), só para efeitos de visualização, mas em realidade, somente podemos certificar estes pontos extremos, pois entre cada um deles, a evolução é dependente dos movimentos de cada junta.

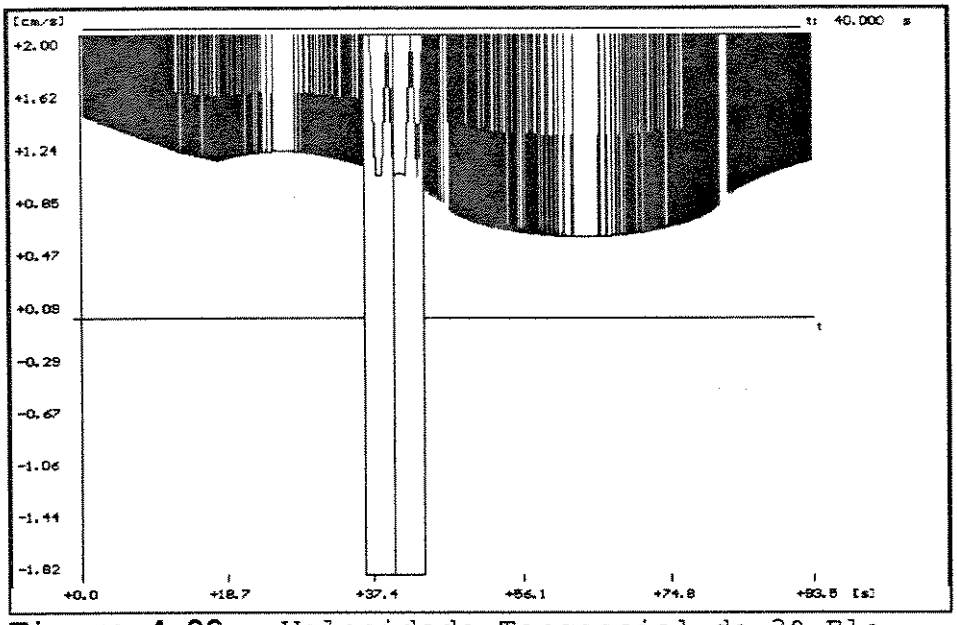


Figura 4.28 - Velocidade Tangencial do 3º Elo.

A **figura 4.29** mostra a evolução em posição das três juntas: cintura (1), braço (2), e ante-braço (3) durante todo o rastreamento. Neste gráfico pode-se verificar que apesar de somente uma junta ser movida por vez, o movimento global não apresenta descontinuidades muito expressivas nestas curvas. O cursor está colocado no instante em que é atingido o 1º ponto da trajetória.

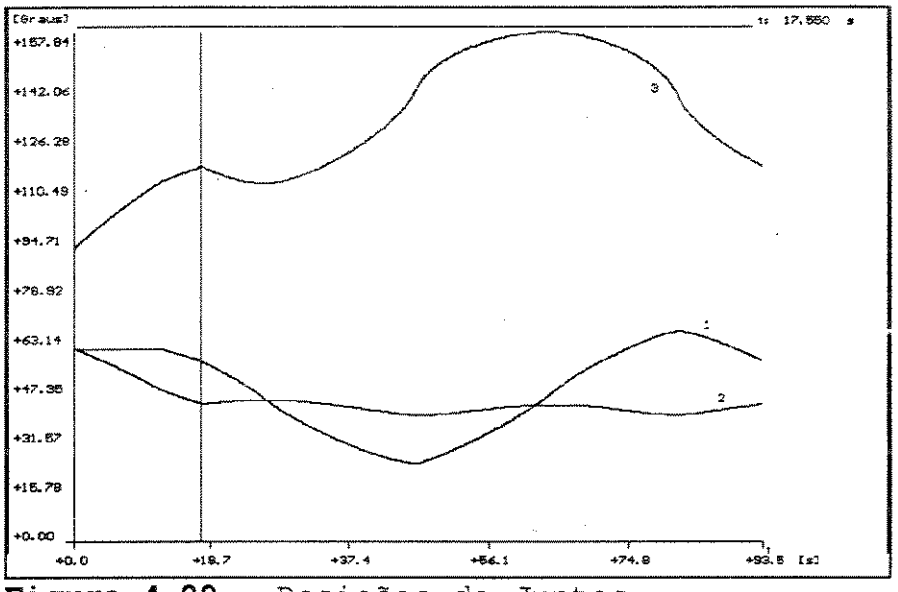


Figura 4.29 - Posições de Juntas.

As **figuras 4.30, 4.31, e 4.32** mostram respectivamente as evoluções das velocidades das juntas 1, 2, e 3. Verifique que apesar de serem oscilatórias, suas variações são de pequena amplitude, e como é evidente, entre zero e a máxima permitida para a junta dentro de um período de amostragem da malha de controle de trajetórias. E as **figuras 4.33, 4.34, e 4.35** mostram as respectivas acelerações ocorridas nestas juntas.

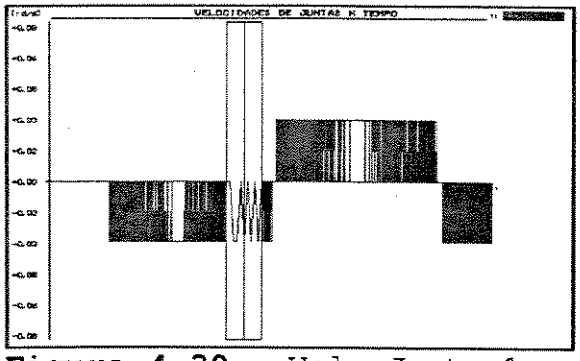


Figura 4.30 - Vel. Junta 1.

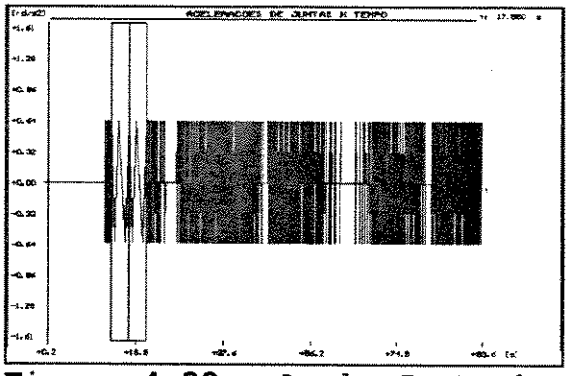


Figura 4.33 - Acel. Junta 1.

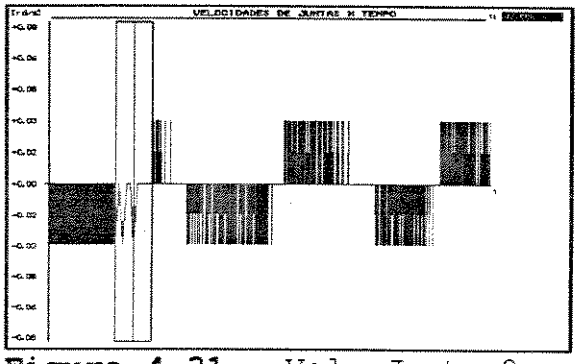


Figura 4.31 - Vel. Junta 2.

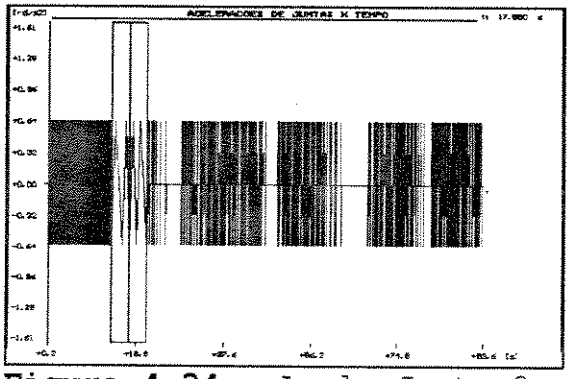


Figura 4.34 - Acel. Junta 2.

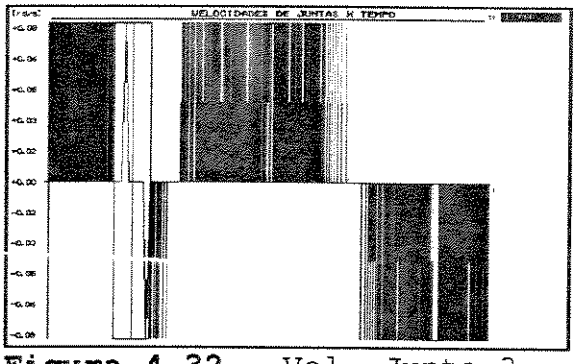


Figura 4.32 - Vel. Junta 3.

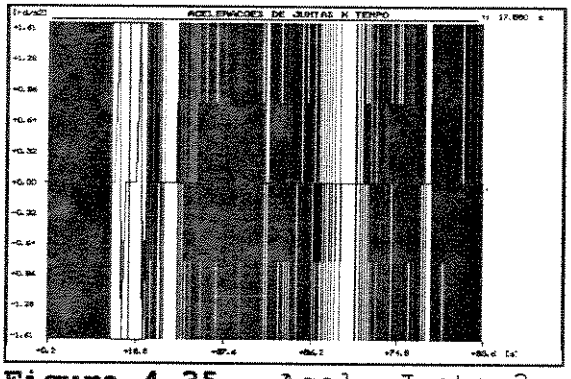


Figura 4.35 - Acel. Junta 3.

A **figura 4.36** mostra o gráfico da evolução do erro de rastreamento em cada aproximação a pontos sucessivos da trajetória (entre cada ponto referente ao erro de aproximação foi interpolada uma reta, somente para a finalidade de visualização). Note

que o erro de cada aproximação sempre foi menor ou igual a precisão exigida, e o erro médio foi de 0,853mm.

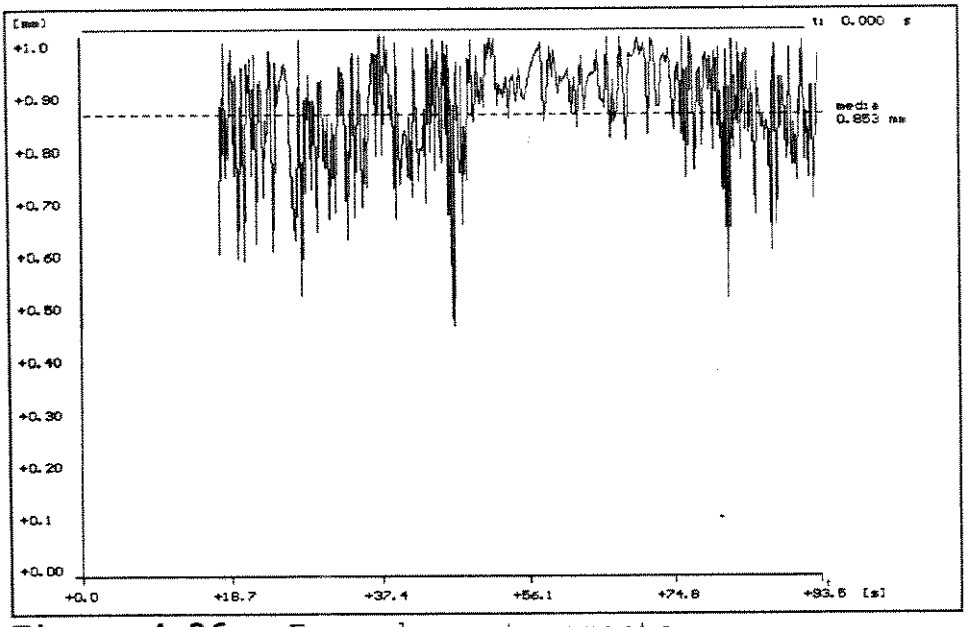


Figura 4.36 - Erro de rastreamento.

A próxima seqüência de figuras mostra o desempenho da técnica quando é utilizado o segundo modo que propusemos, onde podem ser movidas uma ou mais juntas em cada período de amostragem da malha de controle de trajetórias, e mantendo o módulo dos movimentos elementares de cada uma sempre constante (em realidade é uma variação que poderia ser "aprendida" em função de movimentos realizados com a utilização do primeiro modo visto anteriormente).

Comparando as **figuras 4.37, 4.38, 4.39, e 4.40**, obtidas da repetição do mesmo experimento, mas com utilização do algoritmo de controle proposto como segundo modo da técnica, respectivamente com as **figuras 4.24, 4.25, 4.26, e 4.27**, constatamos que os movimentos em posição foram muito semelhantes durante ambos rastreamentos, mas claramente mais suaves na segunda execução do experimento.

● *Além do mais, com a utilização do segundo modo, o robô atingiu o primeiro ponto da trajetória com muito maior rapidez, como era de se esperar, já que com composições de movimentos pode-se produzir movimentos resultantes com maiores velocidades tangenciais. O tempo para atingir o primeiro ponto da trajetória, partindo da mesma condição inicial, foi de 9,75s, que ainda é um tempo expressivo, mas que mostra a existência da possibilidade de aprimoramento do algoritmo básico, e sem perder a convergência no rastreamento.*

Relembramos que os módulos dos movimentos elementares permitidos para cada junta foram os mesmos utilizados no experimento anterior. Neste caso, o tempo total para rastrear completamente a trajetória foi de 60,45s.

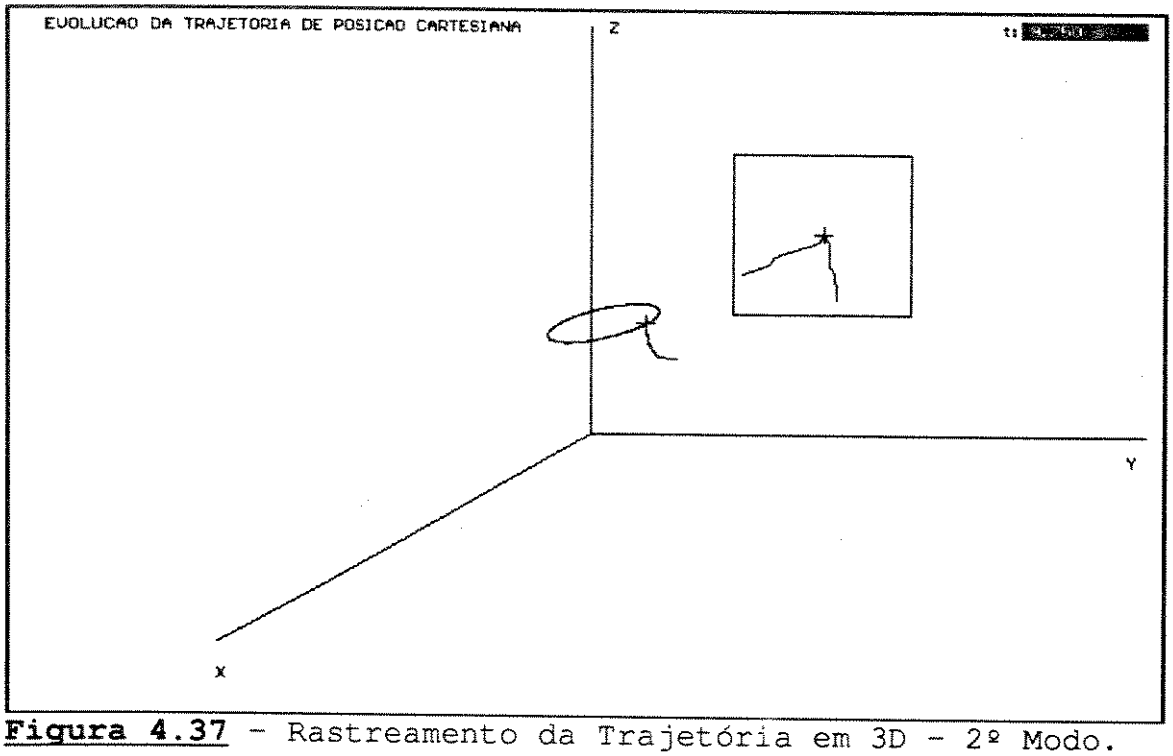


Figura 4.37 - Rastreamento da Trajetória em 3D - 2º Modo.

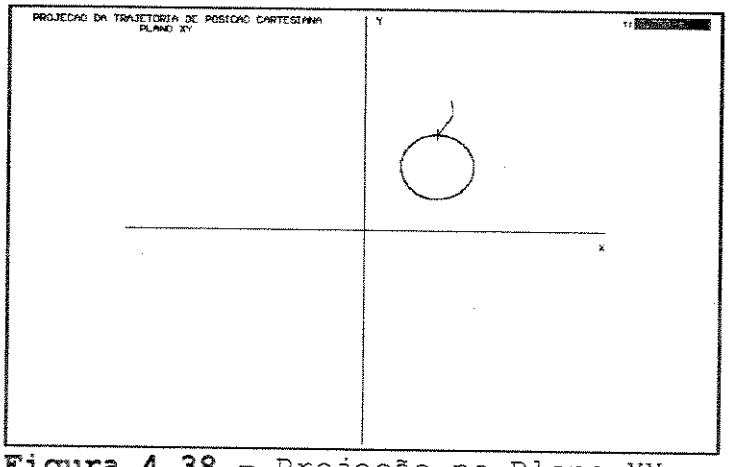


Figura 4.38 - Projeção no Plano XY.

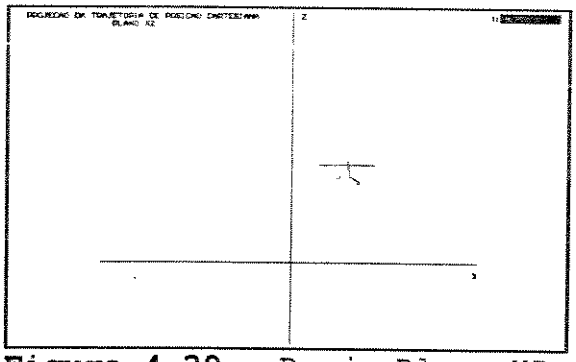


Figura 4.39 - Proj. Plano XZ.

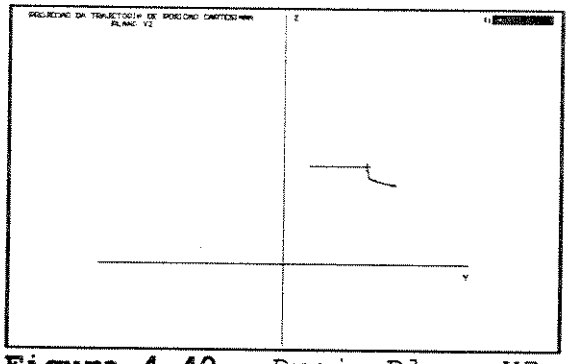


Figura 4.40 - Proj. Plano YZ.

● *Aqui citamos que realizamos vários outros experimentos considerando módulos de movimentos distintos para cada diferente experimento, utilizando a mesma trajetória, com as mesmas características e restrições, e constatamos que pequenas modificações nestes módulos fazem com que o caminho seguido pelo robô possa ser completamente diferente entre cada tramo de rastreamento, mas se a composição dos movimentos individuais permite realizar movimentos, em um período de amostragem, com amplitude menor que a precisão requisitada para atingir cada ponto, o método sempre converge, ainda que tarde mais ou menos tempo para atingir cada ponto, e desde que o espaço não esteja restringido dentro do volume de trabalho exigido para a realização da tarefa, confirmando a previsão teórica feita pela argumentação matemática no início deste capítulo.*

● *Também através destas constatações, fica claro que a técnica pode oferecer um maneira muito robusta de perseguição a uma trajetória qualquer, porque fazendo modificações de forma adequada nestes módulos dos movimentos elementares (ou passos elementares), em função sempre da situação atual na qual o robô se encontra, e da próxima situação desejada, pode-se rapidamente atingí-la. Imagine-se, por exemplo, que existam perturbações externas, por alguma razão, ou que algum dos Escravos atuou mal em determinado instante, isso não significaria nenhum problema grave do ponto de vista de atingir-se o próximo ponto objetivo, unicamente poderiam haver atrasos ou avanços, que pelo mesmo modo de proceder também poderiam ser compensados, incrementando cada algoritmo com programação para gerar níveis de ajustes adequados. Esse assunto certamente deve gerar outros trabalhos futuros (veja* **capítulo V**).

Deixamos claro que os Escravos sempre têm um papel fundamental, que é respeitar os comandos do Mestre. Quanto melhor o fizerem, melhores serão os movimentos. Ou seja:

● *Se todos os Escravos funcionarem mal, não adianta exigir precisão do Mestre. O que esta técnica permite é que possam ocorrer pequenas "falhas" dos Escravos, em determinados momentos do rastreamento, e porisso relaxar um pouco a necessidade de se projetar servocontroladores com alto nível de sofisticação que encarecem muito qualquer projeto.*

Imagine-se também que uma pessoa com "mal de *Parkinson*", ou qualquer outro problema de controle muscular, deseja fazer o traçado de uma linha reta utilizando uma caneta com pena fina. Até poderia realizar esta tarefa bem, com muito "treino", ou pelo menos fazer boas aproximações, mas certamente não deveria "escolher" um trabalho onde as exigências estão além das suas possibilidades físicas. Esse comentário ilustrativo é para reafirmar que o projeto dos servocontroladores de juntas dos robôs, deve ser adequado àquelas exigências feitas pelas tarefas as quais será posto para realizar.

As **figuras de 4.41 a 4.48** mostram a mesma seqüência de gráficos utilizada para a apresentação anterior sobre os desempenho do controle quando foi aplicado o primeiro modo da técnica, mas agora para mostrar o desempenho quando utilizado o segundo modo; pode-se notar que houveram melhoras em todos os sentidos, no que diz respeito ao rastreamento, a única desvantagem é que para a aplicação deste 2º modo é necessária uma maior capacidade computacional, se considerado um mesmo tempo para o processamento dos respectivos algoritmos. Mesmo assim, a carga computacional é muito reduzida, e permite sua execução em tempo-real, mesmo que o Mestre seja uma máquina

com poucos recursos (consideramos aqui, que computadores tipo AT-286, ou um AT 386-33MHz, têm poucos recursos para esta tarefa), e que, por exemplo, a malha de controle de trajetórias tenha um período de 50ms para proceder um *loop* completo de controle.

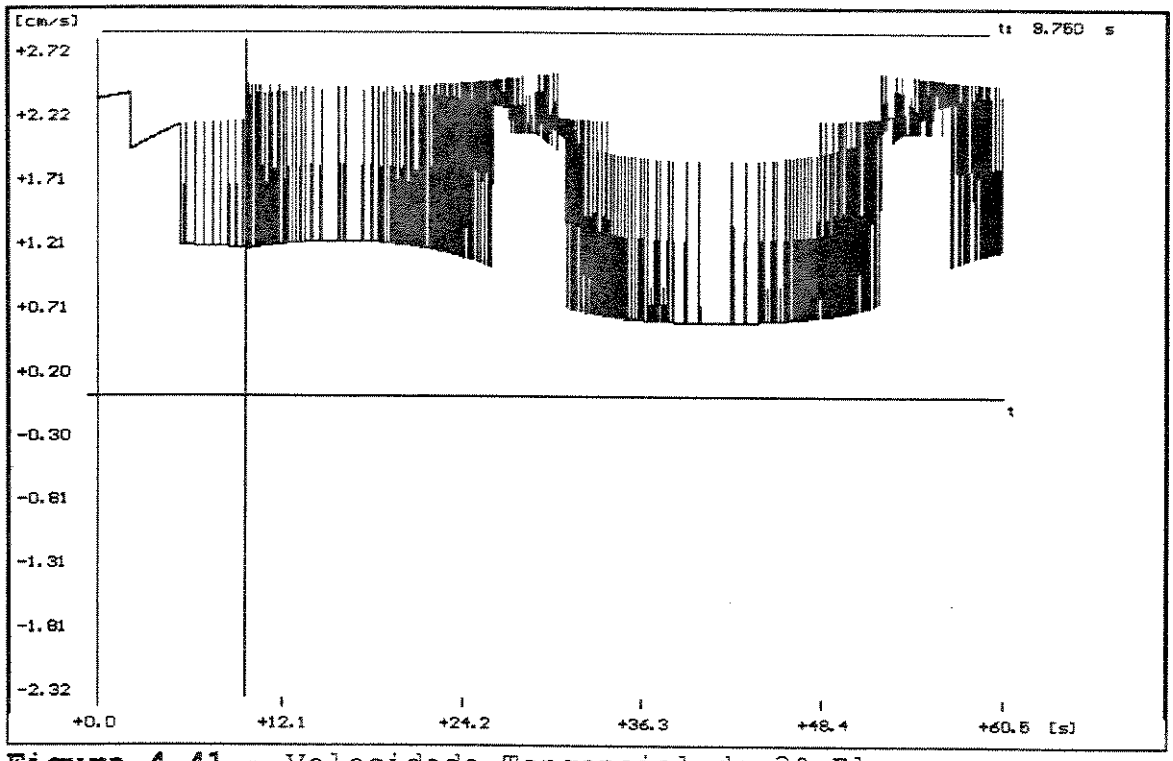


Figura 4.41 - Velocidade Tangencial do 3º Elo.

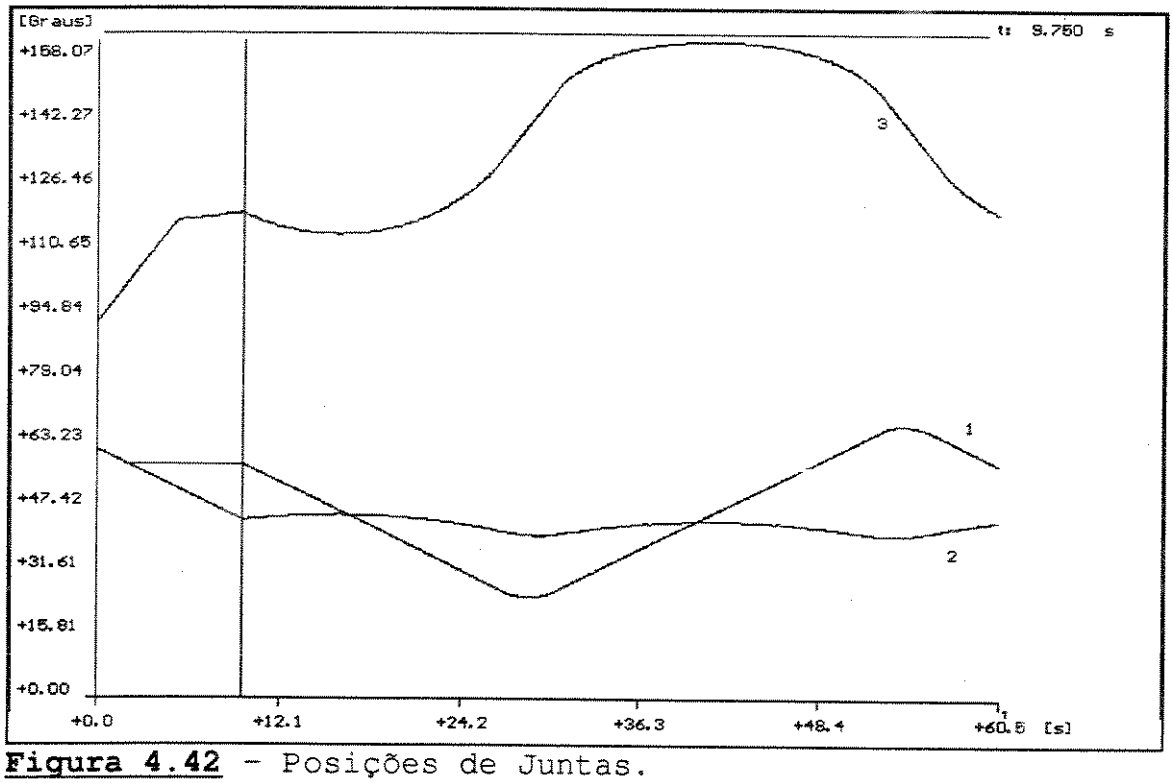


Figura 4.42 - Posições de Juntas.

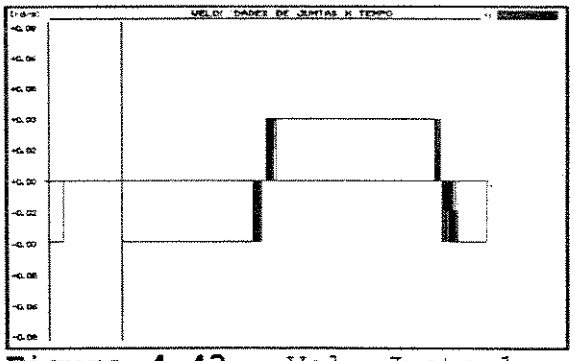


Figura 4.43 - Vel. Junta 1.

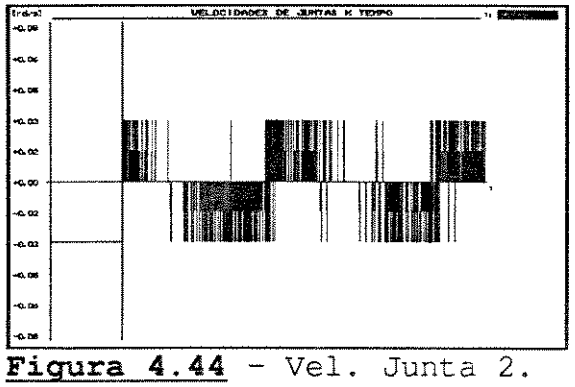


Figura 4.44 - Vel. Junta 2.

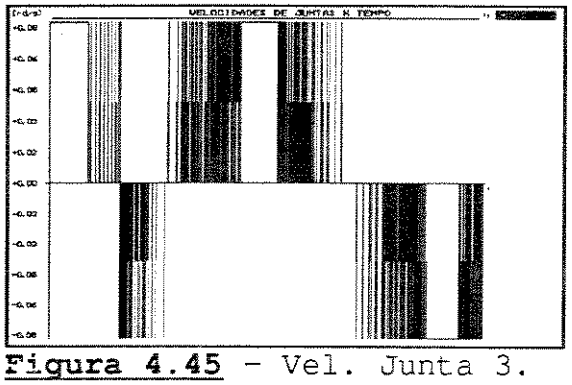


Figura 4.45 - Vel. Junta 3.

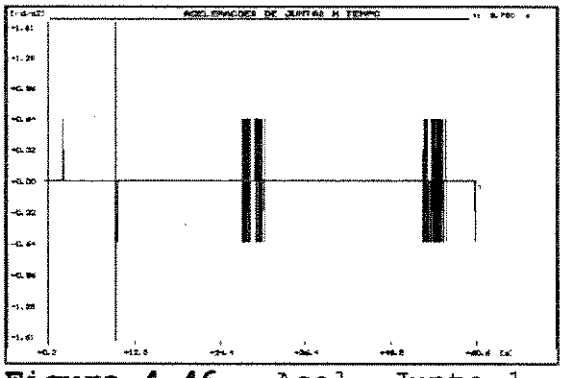


Figura 4.46 - Acel. Junta 1.

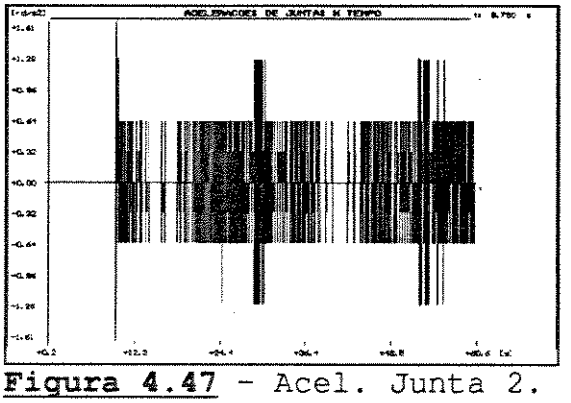


Figura 4.47 - Acel. Junta 2.

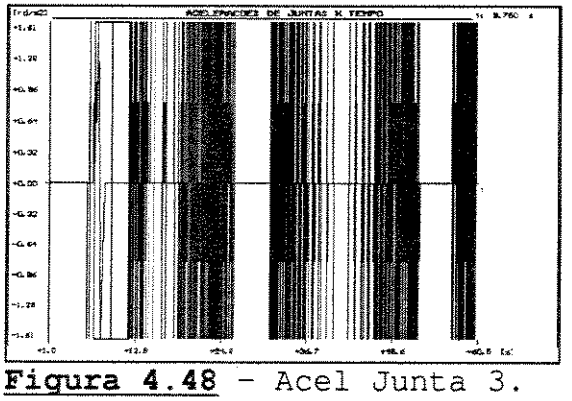


Figura 4.48 - Acel Junta 3.

Verifique através do gráfico da **figura 4.49** que a média do erro de rastreamento também diminuiu, o que era de se esperar, porque com combinações de movimentos pode-se conseguir movimentos tangenciais com amplitudes de deslocamento tanto maiores como menores, se comparados aos movimentos produzidos por uma junta movida individualmente.

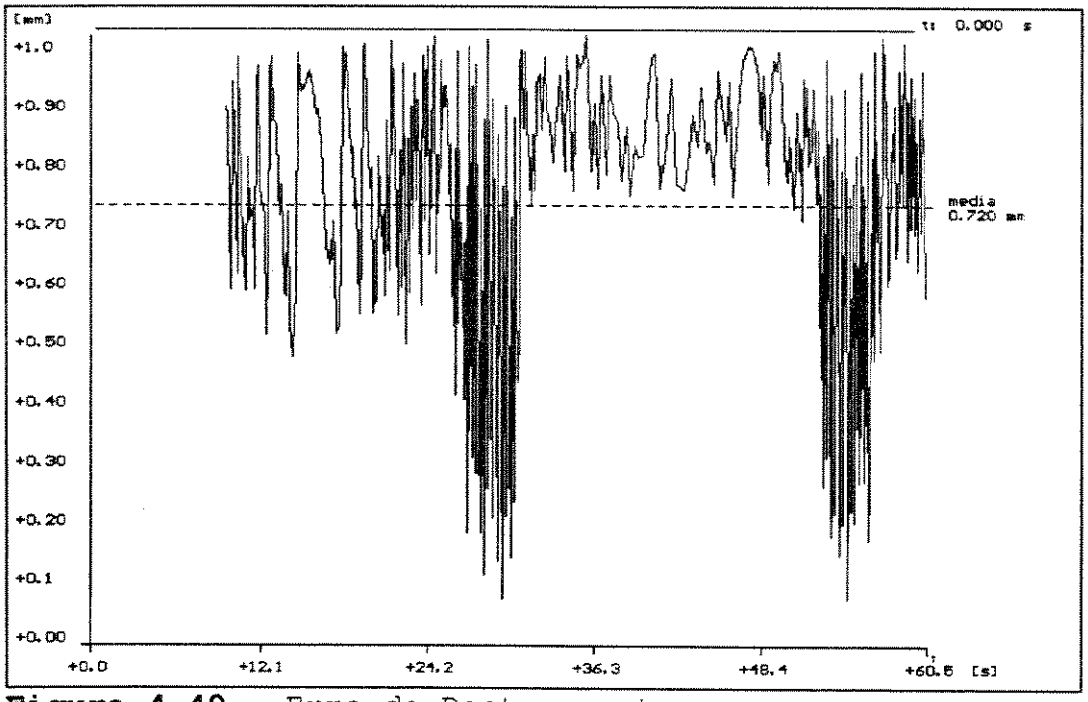


Figura 4.49 - Erro de Rastreamento.

Ao repetirmos o experimento utilizando o terceiro modo proposto para a técnica, também considerando as mesmas características e restrições, verificamos que as melhoras no rastreamento da trajetória foram muito acentuadas (ver **figuras de 4.50 a 4.62**). Houve muito maior rapidez nos movimentos. O robô atingiu o primeiro ponto da trajetória em apenas 300ms, e ela foi perseguida completamente em 18,35s. As oscilações entre os tramos formados pelos intervalos entre pontos diminuiram muitíssimo. A velocidade tangencial foi praticamente constante durante o percurso da trajetória propriamente dita (**figura 4.54**), mas esse fato não significa que sempre deva ser constante, porque não estivemos considerando nenhum critério direto de controle da velocidade tangencial para compor o algoritmo. Verificamos no entanto, que para rastrear retas e circunferências, as velocidades tangenciais se mantém aproximadamente constantes, se os Escravos atuam de maneira correta, mas para curvas mais complexas, como por exemplo, séries de funções trigonométricas, ou exponenciais, etc.., o perfil desta velocidade pode se alterar completamente, ainda que permaneça bastante mais suave que para os outros dois modos.

Nas **figuras 4.59, 4.60, e 4.61** apresentamos os gráficos relativos às acelerações de juntas, e ampliamos a região compreendida entre 3,5s e 18s, para termos maiores detalhes das suas evoluções quando elas possuem baixas amplitudes. Devido a este modo permitir variações nos módulos dos movimentos elementares.

● *De acordo com o critério de otimalidade que apresentamos anteriormente, e dependendo das combinações de movimentos em cada período de amostragem, no tempo inicial e final dos movimentos, podem ocorrer acelerações de maior amplitude, que estão limitadas pelas características físicas do robô, e que são implicitamente consideradas ao limitarmos as velocidades em níveis máximos.*

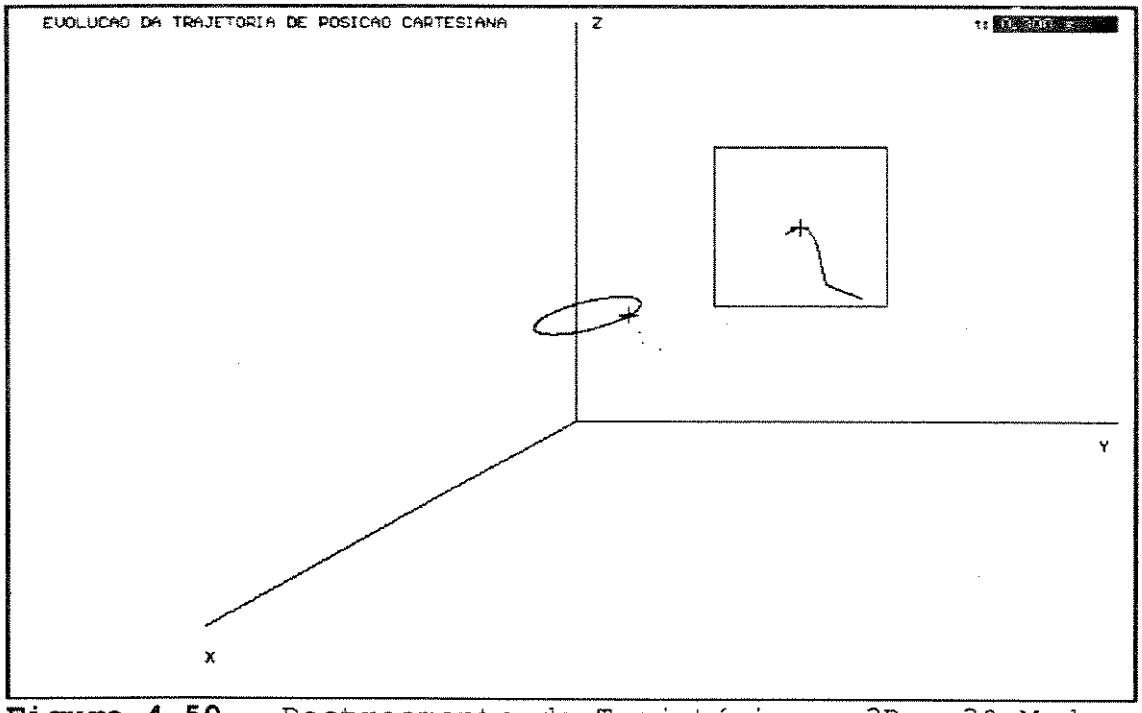


Figura 4.50 - Rastreamento da Trajetória em 3D - 3º Modo.

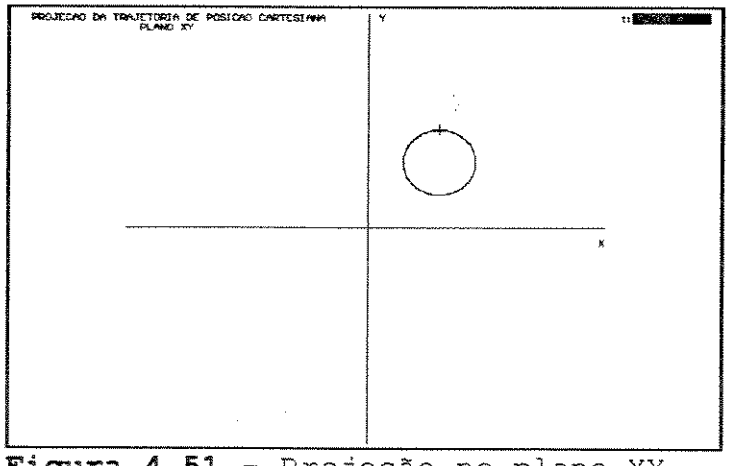


Figura 4.51 - Projeção no plano XY.

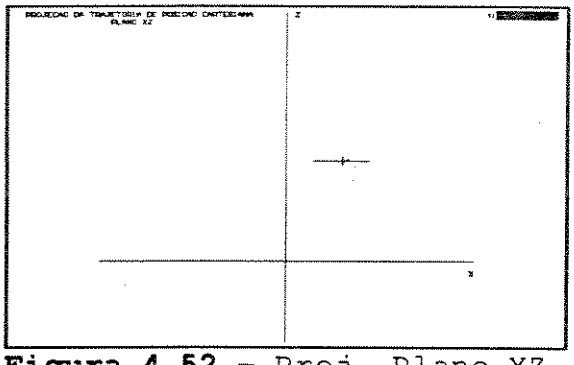


Figura 4.52 - Proj. Plano XZ.

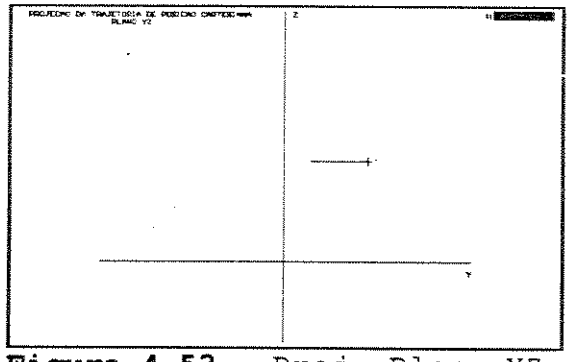


Figura 4.53 - Proj. Plano YZ.

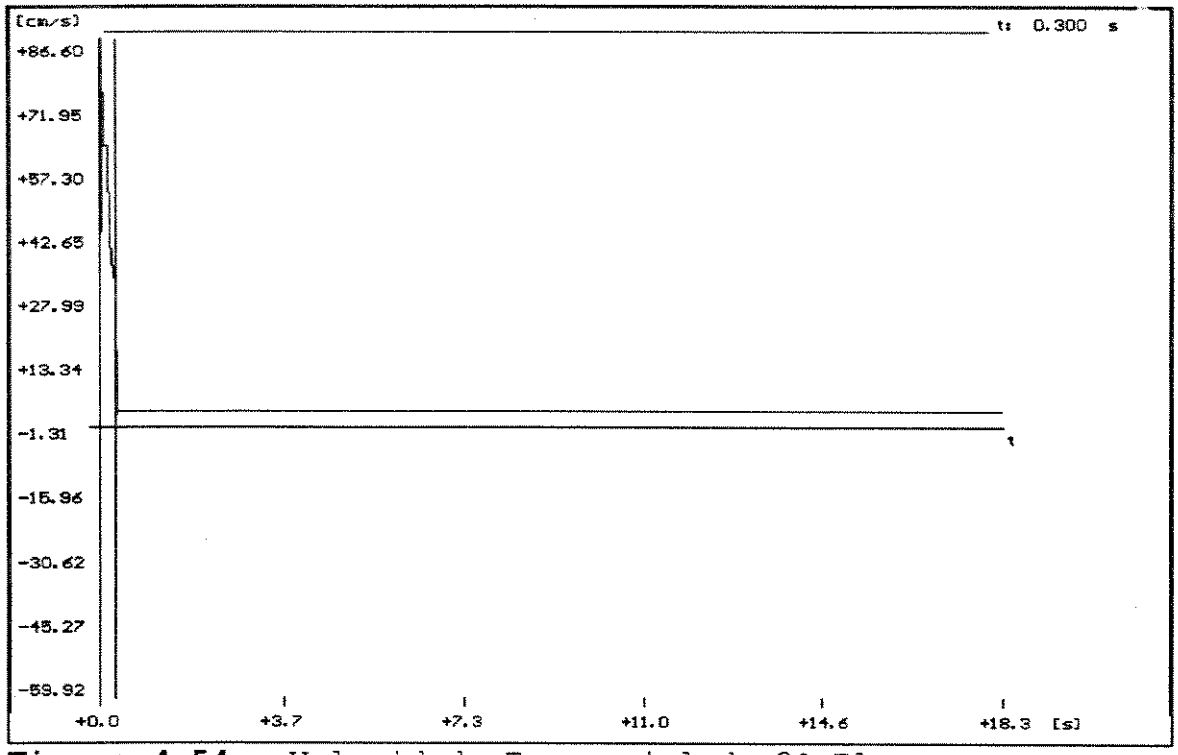


Figura 4.54 - Velocidade Tangencial do 3º Elo.

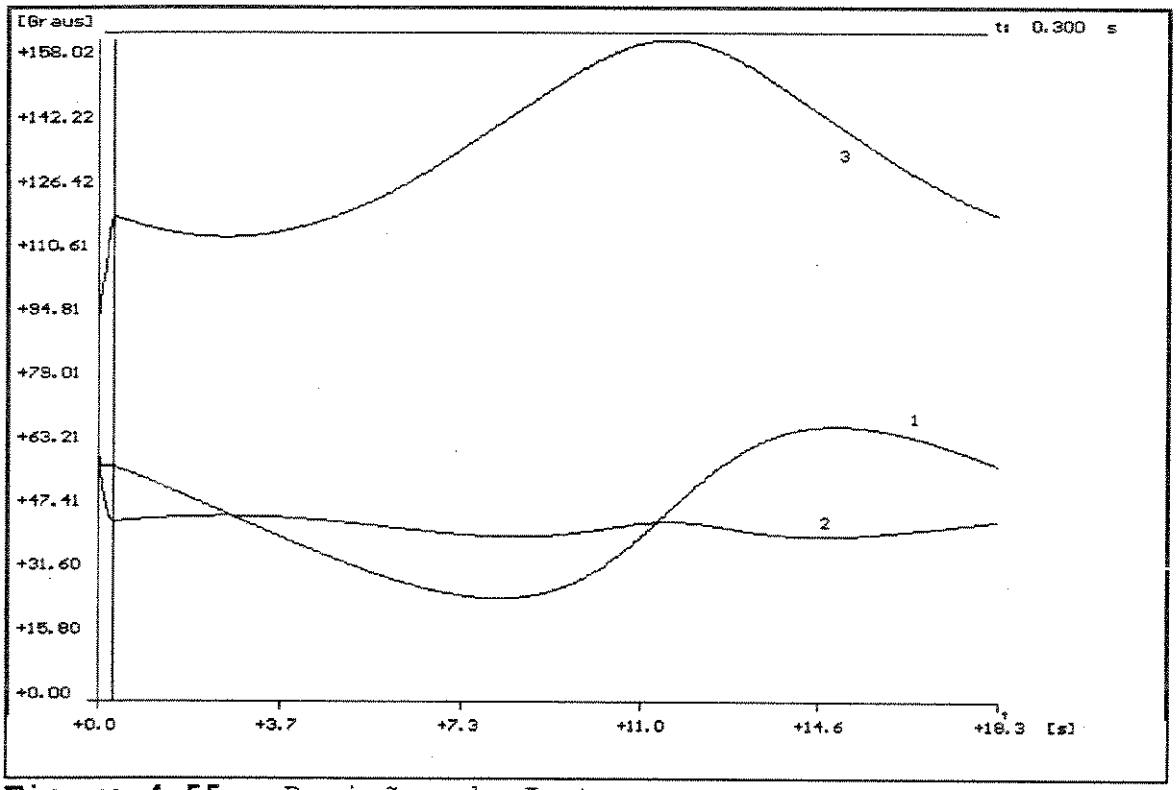


Figura 4.55 - Posições de Juntas.

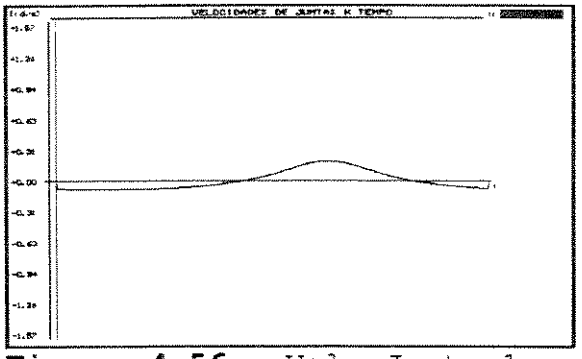


Figura 4.56 - Vel. Junta 1.

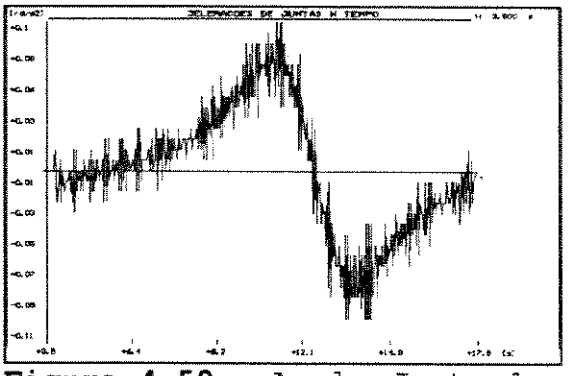


Figura 4.59 - Acel. Junta 1.

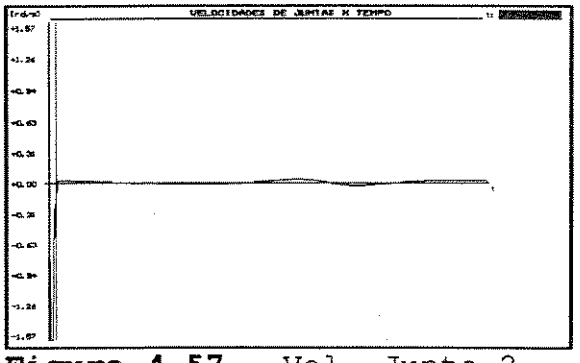


Figura 4.57 - Vel. Junta 2.

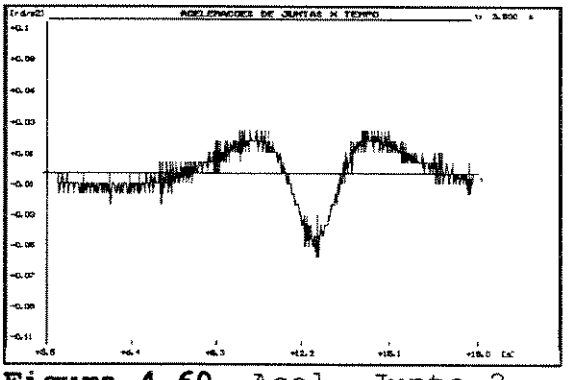


Figura 4.60. Acel. Junta 2.

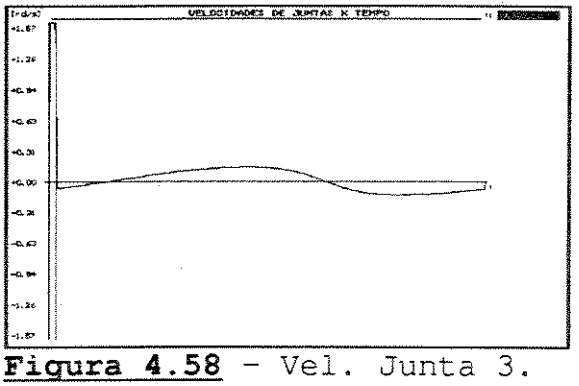


Figura 4.58 - Vel. Junta 3.

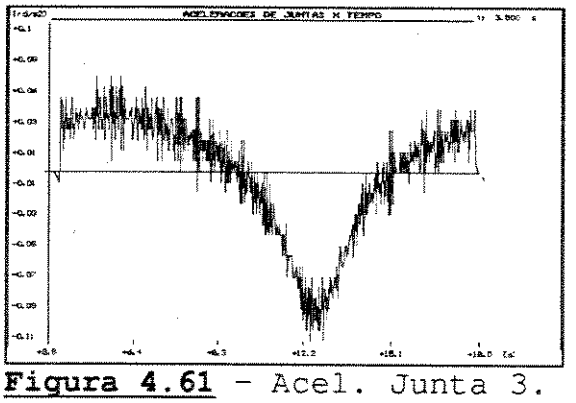


Figura 4.61 - Acel. Junta 3.

Atendo-nos a **figura 4.62** podemos verificar que o erro de rastreamento ficou em média menor do que 10% do erro permitido, ou exigido pela precisão imposta pela trajetória.

Apresentamos também na **figura 4.63** o gráfico da evolução das posições de juntas quando o algoritmo de controle da trajetória encontra soluções intermediárias utilizando o modelo cinemático inverso do robô, a fim de fornecer informações para fazer-se comparações. Note que em relação ao gráfico apresentado na **figura 4.55** houve muita semelhança. Para a obtenção deste gráfico da **figura 4.63** atribuímos à variável COTOVELO o coeficiente que indica ACIMA, e para a variável BASE, o coeficiente que indica DIREITA.

Note também, que inicialmente, como não há critério de máxima velocidade, com a utilização do modelo cinemático inverso (MGI), sempre o próximo ponto pertencente a trajetória é atingido imediatamente após o anterior, fato que pode não ser muito realista do ponto de vista temporal, mas que fornece a solução cinemática para cada ponto com precisão total, se supusermos que o modelo cinemático é correto.

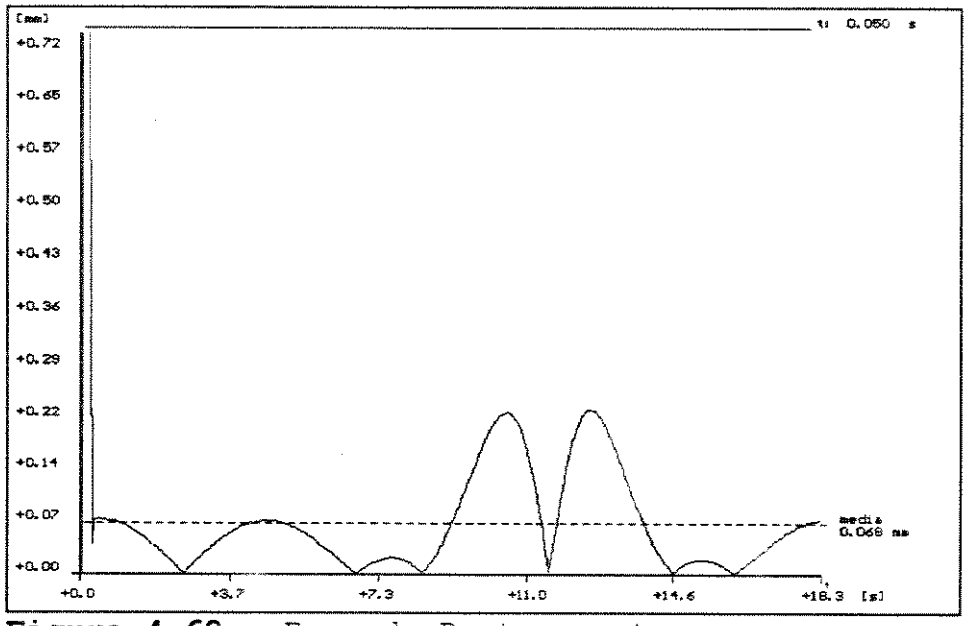


Figura 4.62 - Erro de Rastreamento.

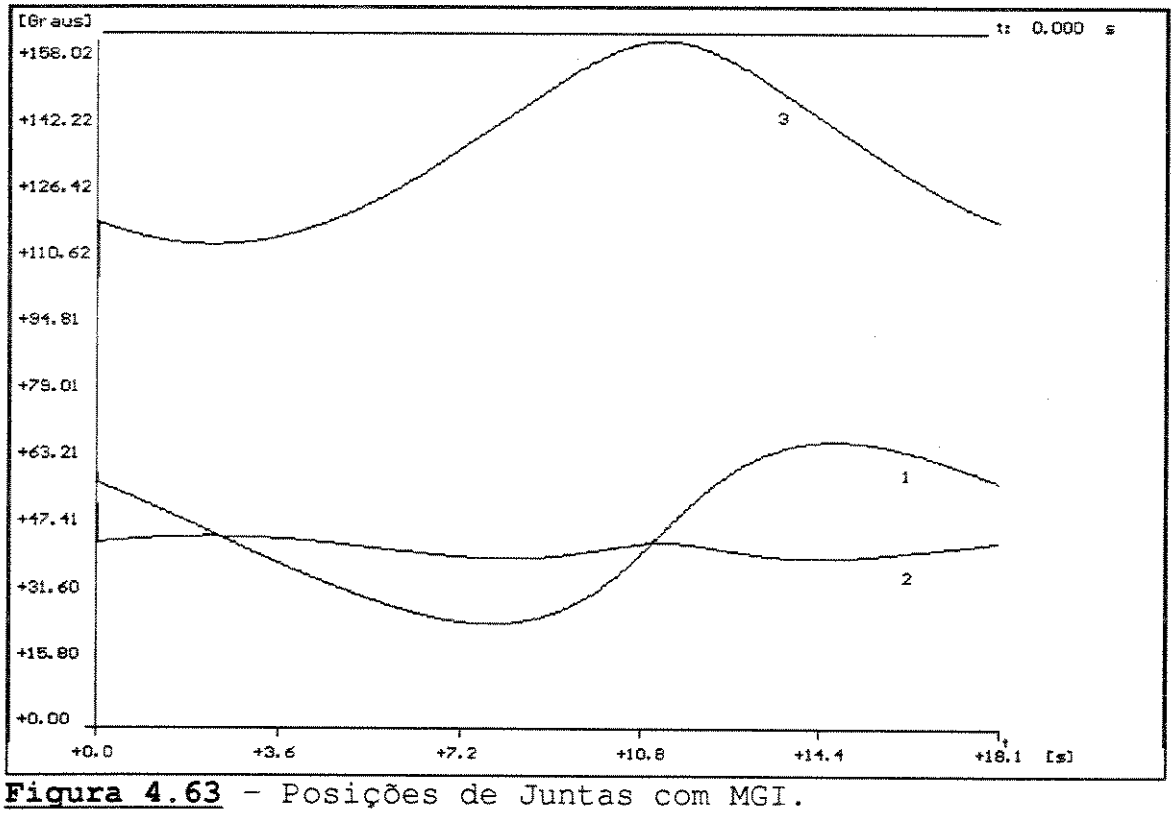


Figura 4.63 - Posições de Juntas com MGI.

Também a título de ilustração, a **figura 4.64** mostra em um só gráfico o comportamento das velocidades de juntas utilizando a modelagem inversa no algoritmo, e supondo o mesmo período de 50ms para a malha de controle de trajetórias.

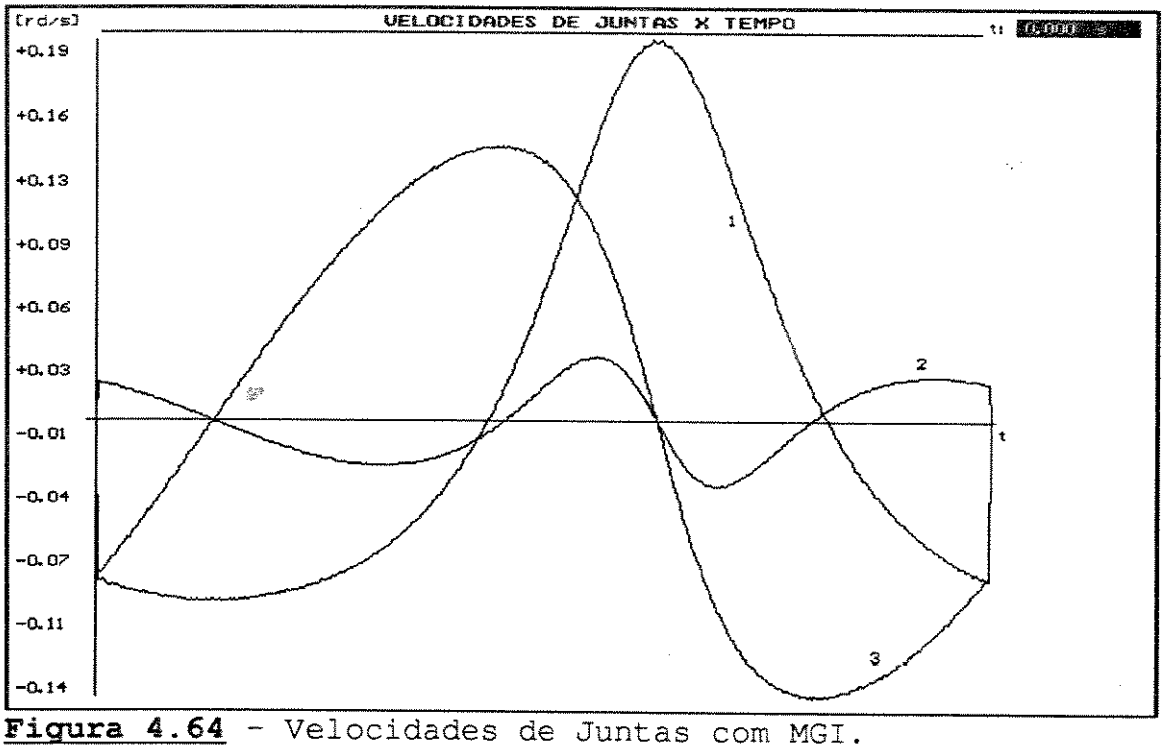


Figura 4.64 - Velocidades de Juntas com MGI.

A seguir fazemos a suposição de que os Escravos não atuam de forma perfeita, e para isso injetamos ruídos aleatórios nas saídas de posição de cada junta, que significam erros de resposta ao seguimento dos níveis de ajustes exigidos em cada período de amostragem pelo algoritmo de controle executado pelo Mestre. Mostramos os resultados obtidos com a utilização do 3º modo da técnica, e permitimos que as amplitudes destes ruídos sejam de até meio grau em cada junta (que conjuntamente significam perturbações bastante acentuadas no elemento terminal do robô). Ver gráficos das **figuras de 4.65 a 4.75**.

● *Através desses gráficos podemos constatar, que o método também sempre conduziu o robô razoavelmente bem até o final do rastreio. Ainda consideramos a mesma trajetória, e as mesmas restrições apresentadas anteriormente. As diferenças mais significativas são que agora o tempo total de movimentos foi maior, o que era de se esperar, porque somente considerando aquilo que propuzemos para a técnica até o momento, não existem critérios para fazer movimentos com maior ou menor aceleração para compensar estes avanços e atrasos provocados pelas perturbações, mas como dissemos, é perfeitamente possível anexar critérios assim, desde que se disponha de máquinas com maior capacidade computacional para realizarem outros cálculos e decisões necessárias para tal objetivo.*

Como pode-se constatar nos gráficos onde são apresentadas as evoluções das posições, velocidades e acelerações da estrutura, houve reações mais intensas por parte do controle para manter o rastreamento dentro da precisão.

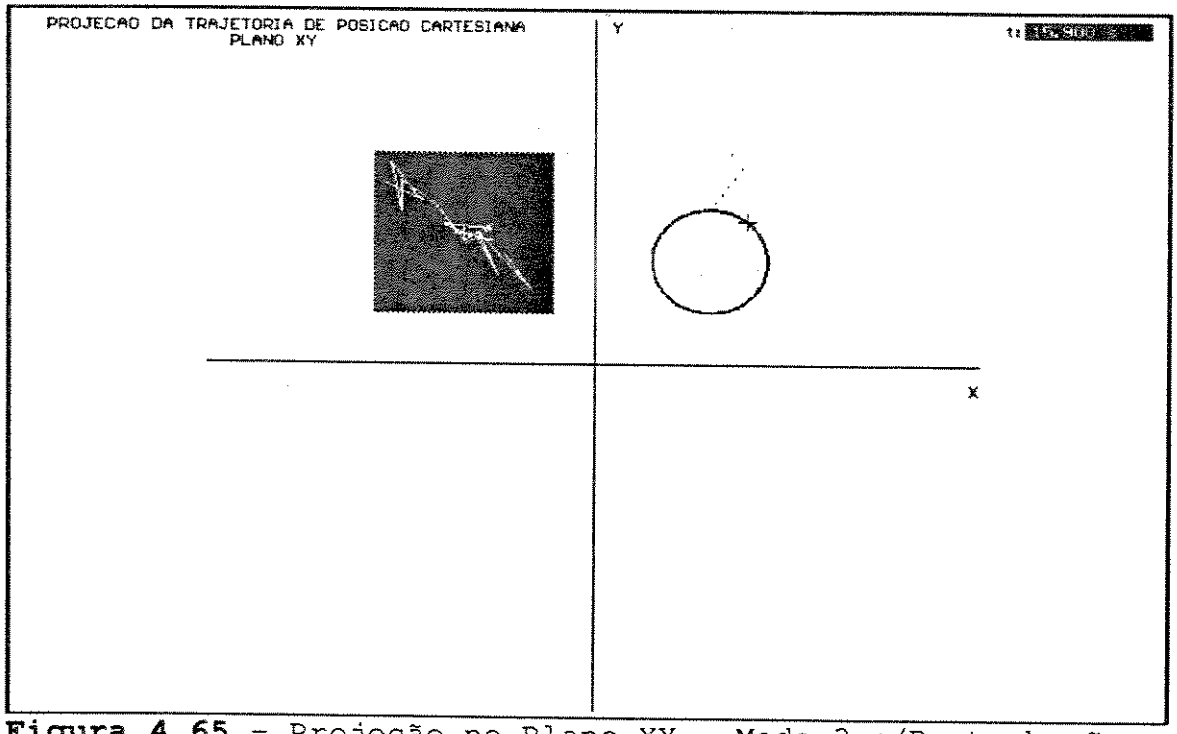


Figura 4.65 - Projeção no Plano XY - Modo 3 c/Perturbações.

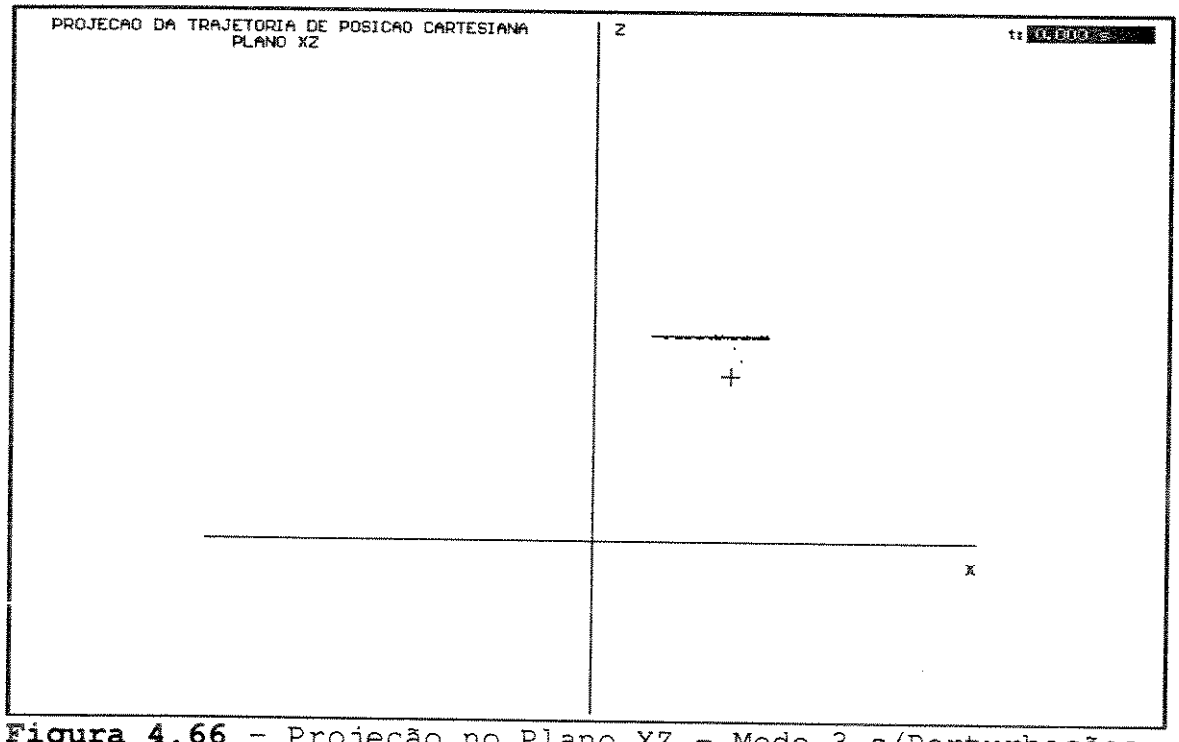


Figura 4.66 - Projeção no Plano XZ - Modo 3 c/Perturbações.

Os gráficos das **figuras 4.65 e 4.66** mostram as projeções respectivamente nos planos XY e XZ, da evolução do rastreamento, e neles pode-se ver que houve um pequeno alargamento da espessura da curva produzida em torno da curva que seria

seguida se não houvesse nenhuma perturbação, ou se os Escravos atuassem de maneira perfeita, fato este que dependendo da aplicação, é perfeitamente admissível.

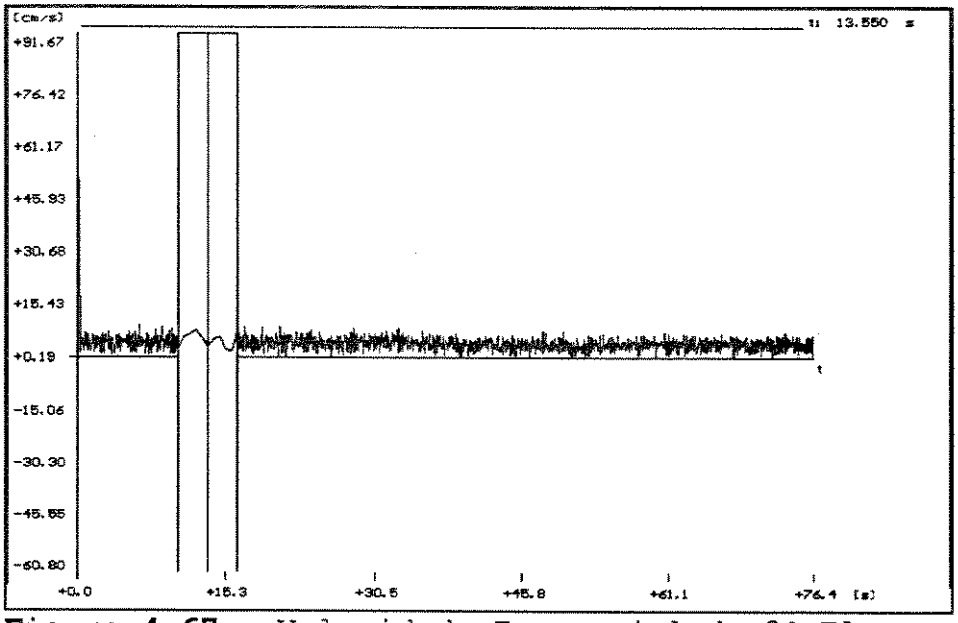


Figura 4.67 - Velocidade Tangencial do 3º Elo.

É interessante verificar que a etapa de aproximação (movimentos de busca do primeiro ponto pertencente a trajetória) exigiu um tempo muito pouco superior ao mesmo movimento sem perturbações. O primeiro ponto da trajetória foi atingido em 0,5s, e o tempo total para o rastreamento completo da trajetória se elevou para 76,4s.

Verifique também na janela de amplificação incluída na **figura 4.65**, que os movimentos da etapa de precisão, quando o robô já está percorrendo a trajetória, ficam com oscilações bem mais acentuadas, mas de pequenas amplitudes, que é claro, são função dos desempenhos dos Escravos.

A velocidade tangencial do extremo do terceiro Elo manteve aproximadamente o mesmo valor médio, porém com um desvio padrão mais acentuado (ver **figura 4.67**).

A **figura 4.68** mostra a evolução das posições de juntas, onde se pode ver que houve uma menor suavidade de movimentos. Este gráfico está mostrando todo o movimento.

As **figuras de 4.69 a 4.74** mostram as respectivas velocidades e acelerações das juntas da cintura, braço, e ante-braço, onde se pode constatar que houveram maiores oscilações e maior reação do sistema de controle.

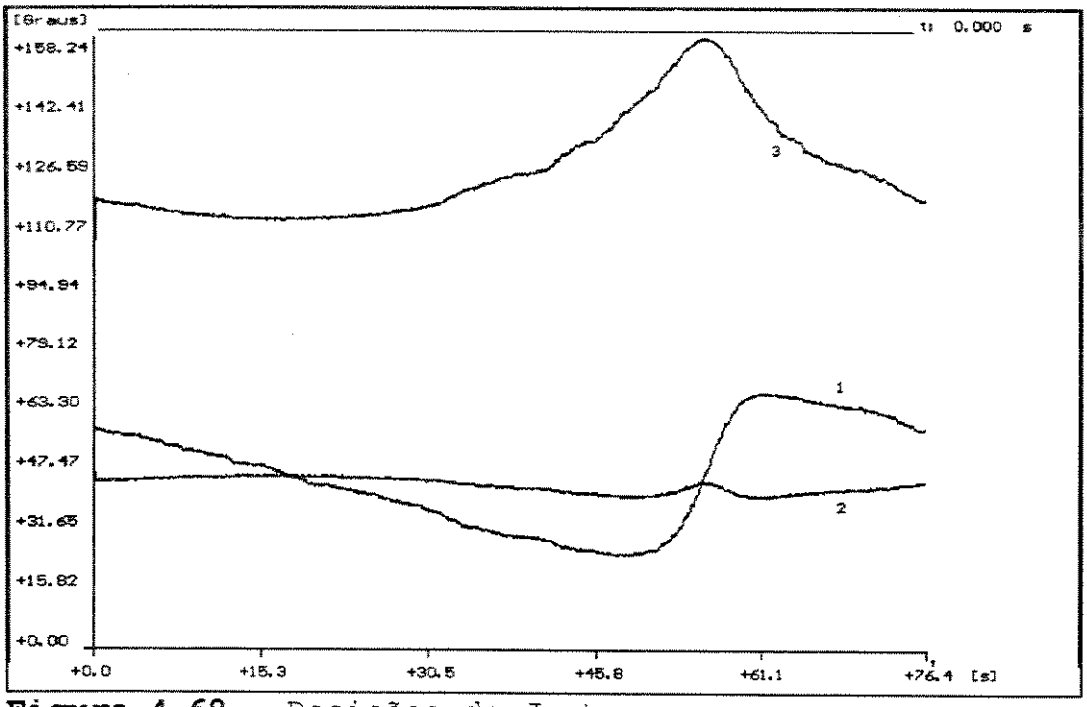


Figura 4.68 - Posições de Juntas.

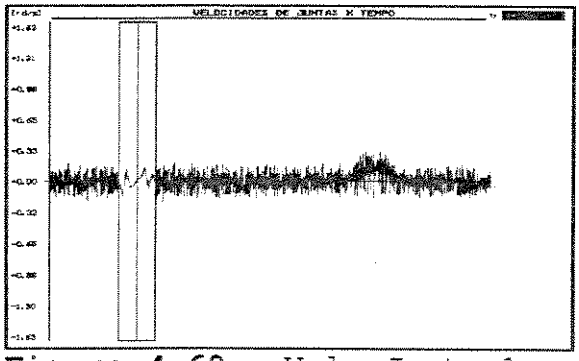


Figura 4.69 - Vel. Junta 1.

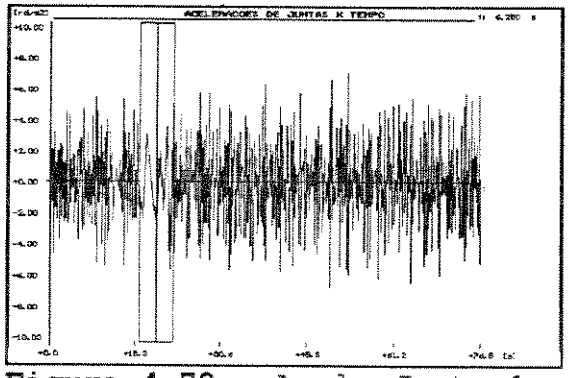


Figura 4.72 - Acel. Junta 1.

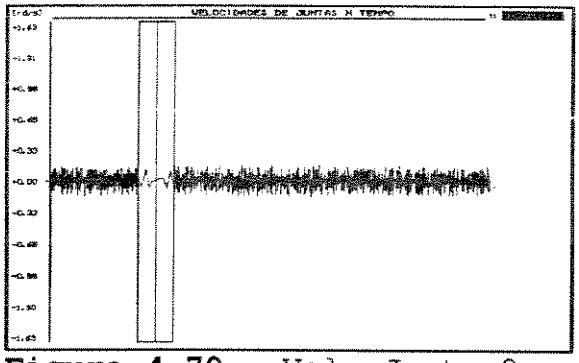


Figura 4.70 - Vel. Junta 2.

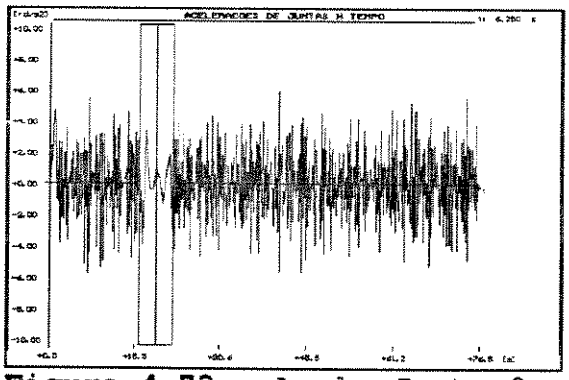


Figura 4.73 - Acel. Junta 2.

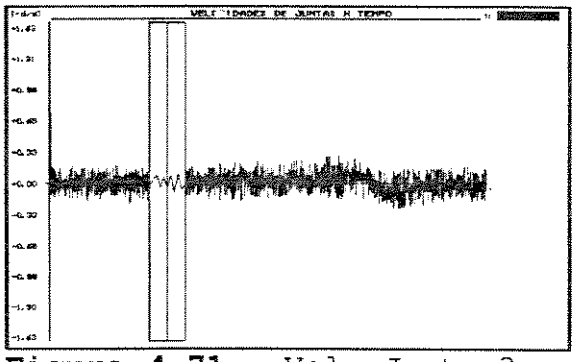


Figura 4.71 - Vel. Junta 3.

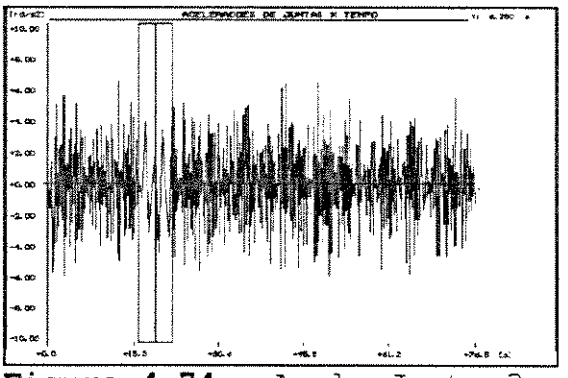


Figura 4.74 - Acel. Junta 3.

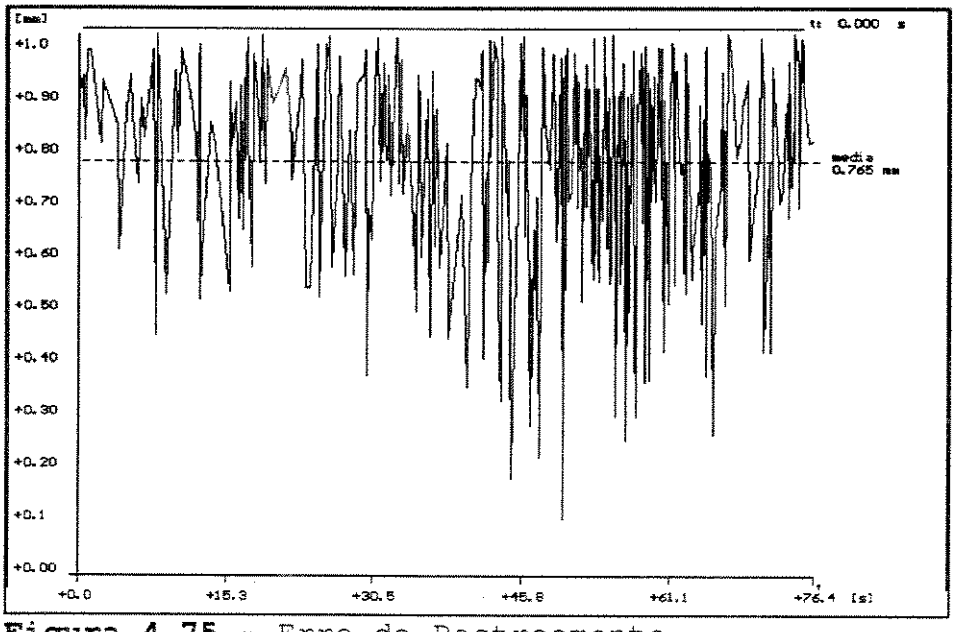


Figura 4.75 - Erro de Rastreamento.

Na **figura 4.75** pode-se ver que a média do erro de rastreamento aumentou, mas os valores máximos desses erros permaneceram dentro da precisão de 1mm.

Citamos também que realizamos experimentos supondo desvios constantes por parte da resposta dos Escravos aos níveis de ajustes enviados pelo Mestre em cada período de amostragem, e constatamos que ambos os modos da técnica também realizam um controle eficiente desde que hajam combinações dos movimentos elementares das juntas capazes de produzir movimentos do extremo do terceiro Elo com amplitude menor ou igual à precisão exigida pela tarefa.

● *Neste último experimento a utilização puramente da modelagem cinemática inversa fracassa, porque as perturbações, ou erros nas respostas dos Escravos não são previstas, ou seja, não existe capacidade de reação, ou de recuperação.*

A seguir mostramos os resultados de um experimento realizado considerando uma trajetória helicoidal cujas equações paramétricas são:

$$X = 200 + 100.sen(3.k)$$

$$Y = 200 + 100.\cos(3.k) \qquad \textbf{(4.21)}$$

$$Z = k$$

com $0 \leq k \leq 360$, e passo de discretização 1 (360 pontos).

Fazemos isso através da mesma seqüência de gráficos utilizada para mostrar os resultados das simulações anteriores. Neste caso, exigimos uma precisão de 0,02mm (algo que exigiria uma precisão extremamente elevada por parte da estrutura mecânica do robô, mas que nos serve para mostrar como a técnica também é eficiente, mesmo para exigências muito severas de precisão). Consideramos também a execução do terceiro modo proposto, e que os Escravos atuem de forma a rastrear com precisão os níveis de ajustes gerados pela técnica durante o rastreamento.

Também inicializamos o robô **JECAII** com $\theta1=\theta2=\theta3=0°$ e velocidades iniciais zero em todas as juntas; posiçao esta que em coordenadas cartesianas do extremo do terceiro Elo equivalem a $P_{(X,Y,Z)}=(0;0;990)$mm.

O primeiro ponto da trajetória é atingido em 1,2s e o tempo total de rastreamento, considerando um período de amostragem da malha de trajetórias de 50ms, é de 19,2s.

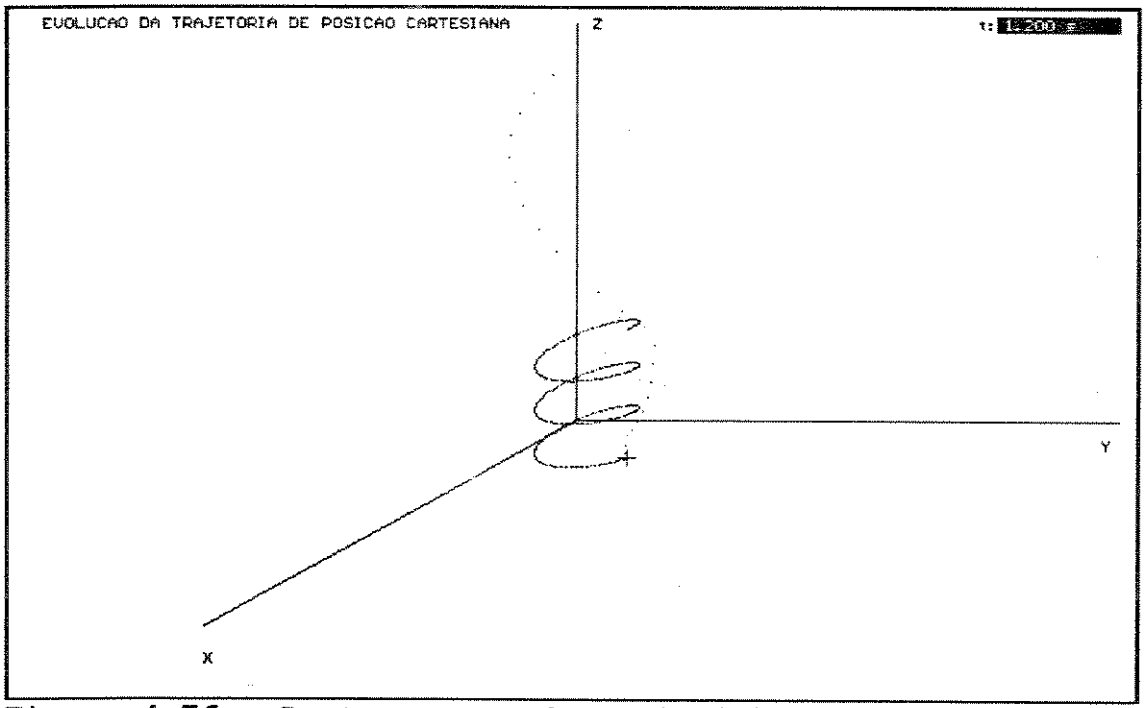


Figura 4.76 - Rastreamento da Trajetória em 3D - 3º Modo.

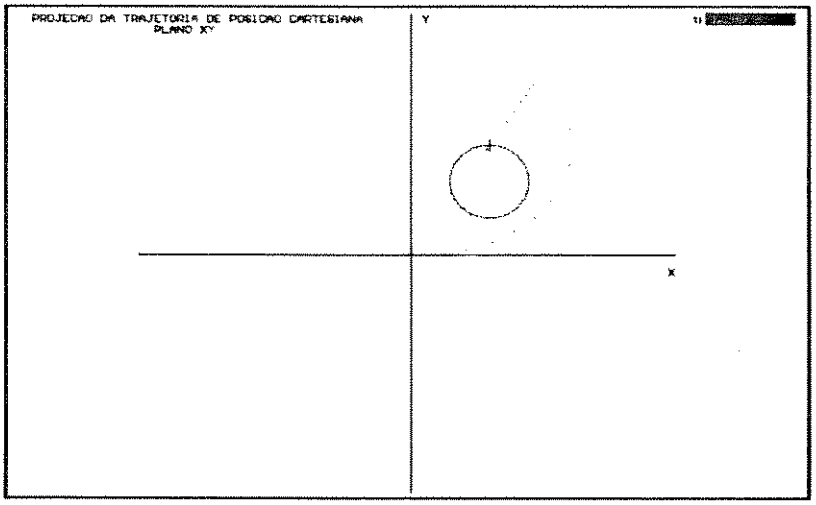


Figura 4.77 - Projeção no Plano XY.

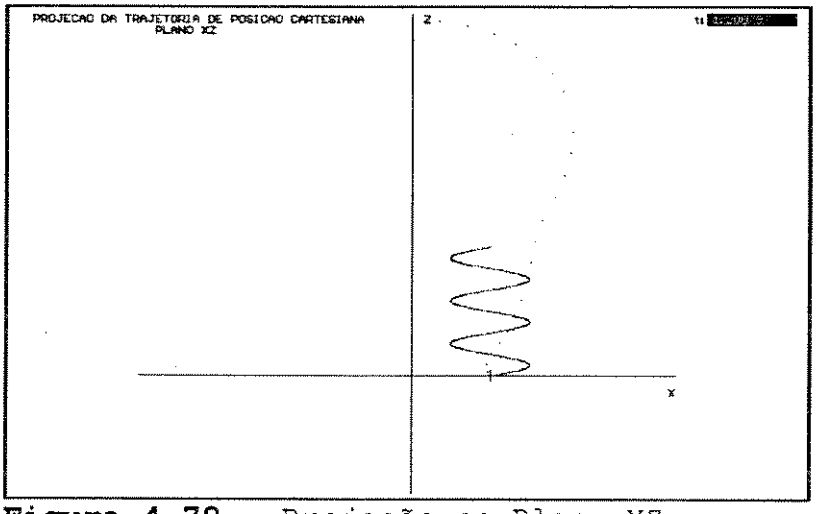


Figura 4.78 - Projeção no Plano XZ.

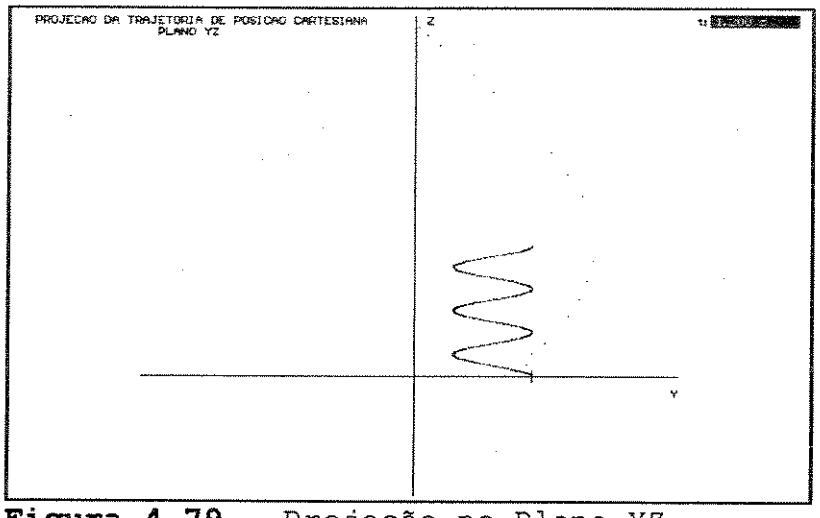


Figura 4.79 - Projeção no Plano YZ.

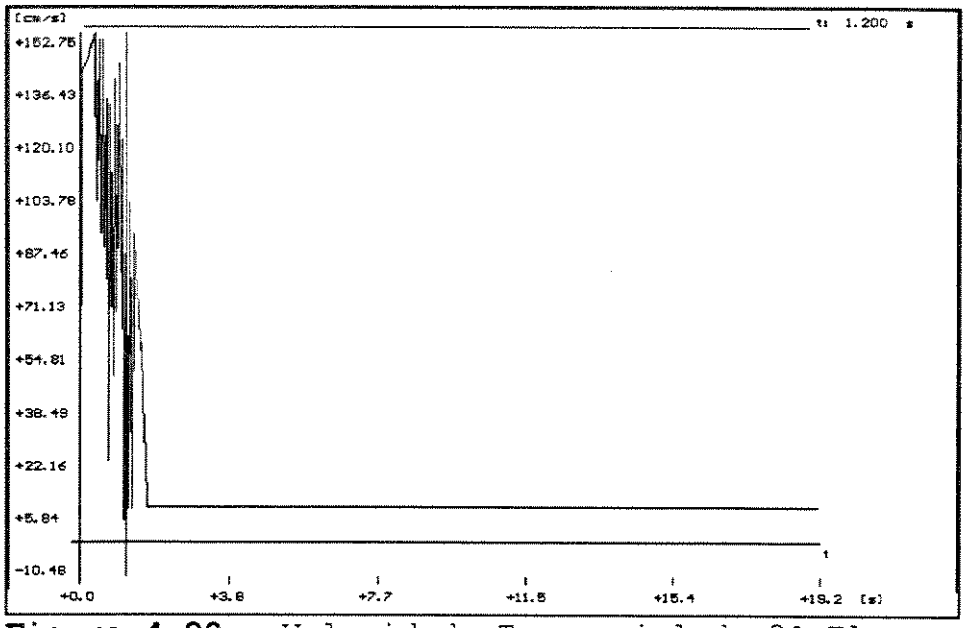


Figura 4.80 - Velocidade Tangencial do 3º Elo.

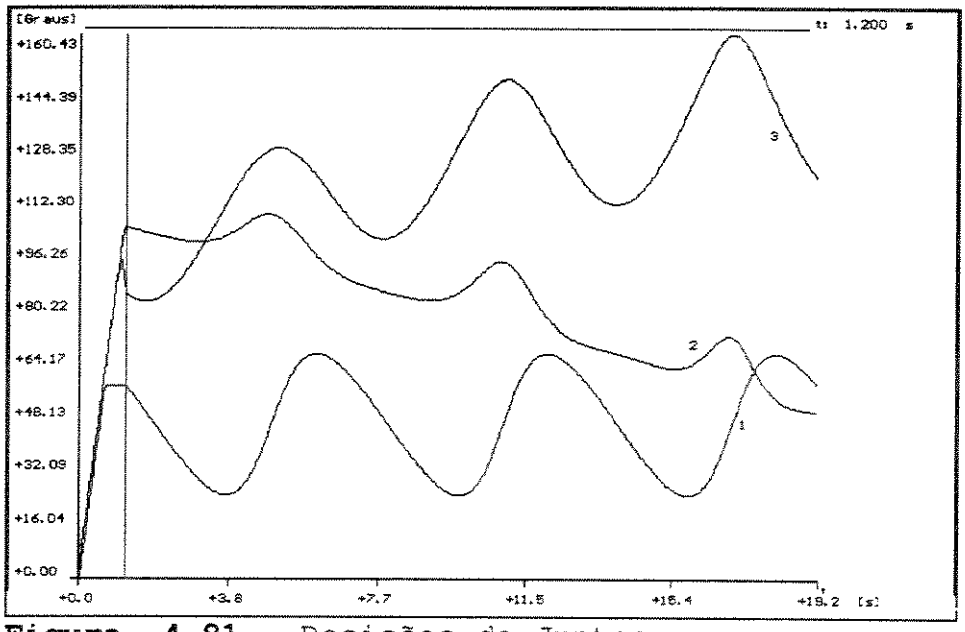


Figura 4.81 - Posições de Juntas.

No gráfico da **figura 4.80** pode-se verificar que durante a etapa de aproximação ao primeiro ponto da trajetória, existem variações bastante acentuadas na velocidade tangencial do extremo do terceiro Elo, e em conseqüência pouca suavidade nos movimentos, e isso se deve ao fato que existem limitações impostas para as velocidades de juntas, e os passos ótimos encontrados através do algoritmo de controle podem não ser

possíveis por causa destas limitações, e como comentamos anteriormente, nestes casos elegemos o passo máximo possível naquele sentido indicado pelo algoritmo para cada junta em cada período de amostragem da malha de controle de trajetórias. Mas note que quando o robô está rastreando a trajetória, as respostas de velocidades tangenciais ficam enormemente melhoradas.

● *Essa constatação nos leva a concluir que o passo de discretização da trajetória a ser rastreada é um fator muito importante a ser considerado pelo projetista, se a intenção é fazer com que os movimentos sejam suaves. Por exemplo, pode-se desenvolver um algoritmo que interpole automaticamente uma curva entre onde o robô se encontra, e o primeiro ponto pertencente a trajetória de forma a fazer estas melhoras na suavidade dos movimentos (Sempre que é necessário fazer saturações nos sinais de ajustes gerados pelo algoritmo de controle, a suavidade dos movimentos fica prejudicada).*

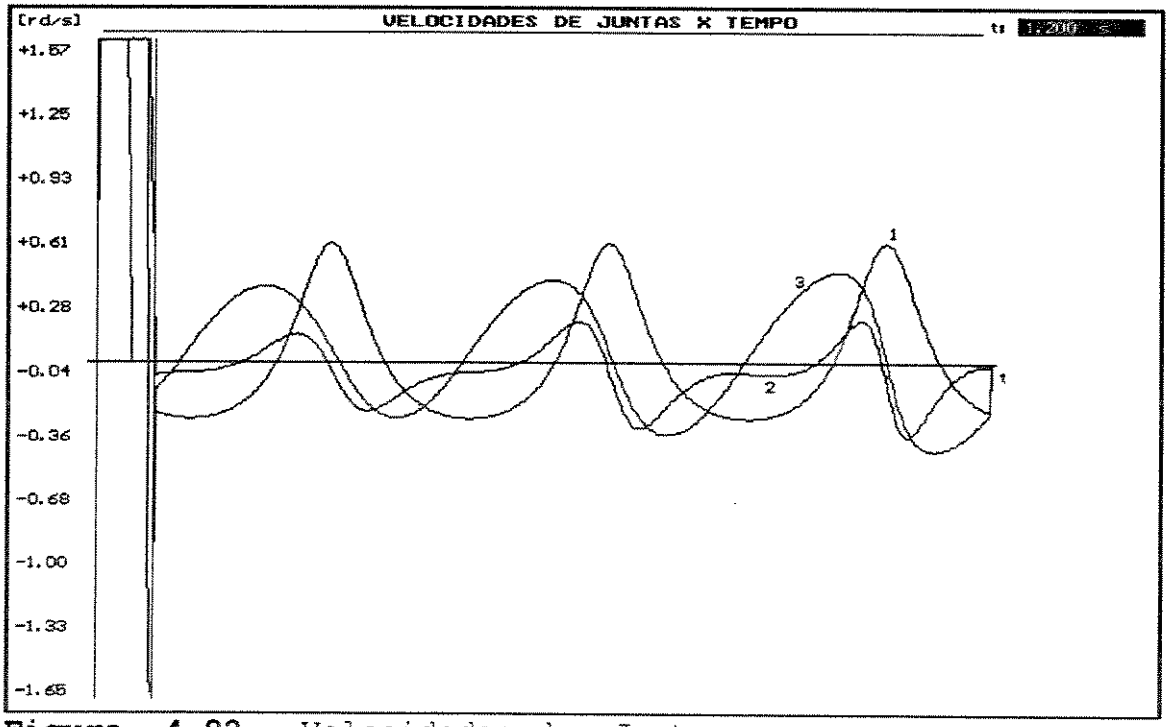


Figura 4.82 - Velocidades das Juntas.

O gráfico da **figura 4.82** mostra que as respostas em velocidades das juntas são relativamente muito suaves durante toda a etapa de rastreamento da trajetória, e esse fato surpreende um pouco, porque não estamos considerando nenhum critério de controle sobre as velocidades. Talvez essa seja uma característica particular da aplicação da técnica para o controle de robôs com o tipo de estrutura mecânica RRR para posicionamento espacial conforme a estrutura do **JECAII**. Também não devemos esquecer o fato que estamos considerando que os Escravos estejam atuando de modo perfeito, o que é difícil de ocorrer na prática.

● *De qualquer forma, como vimos através dos experimentos anteriores, se ocorrerem pequenas falhas nas suas atuações, apesar dos movimentos serem menos suaves, a técnica oferece a capacidade de realizar correções, e mesmo assim pode resultar um rastreamento bastante eficaz. Acreditamos que controladores do tipo PID bem otimizados para*

cada junta podem aproximar muito os resultados práticos a estes que apresentamos através de simulações, e mesmo que não se consiga um bom desempenho dos Escravos utilizando estes tipos de controladores, pode-se recorrer a outros mais sofisticados, (veja ***Capítulo II****), ou seja, o problema passa a ser de projeto de controladores de juntas.*

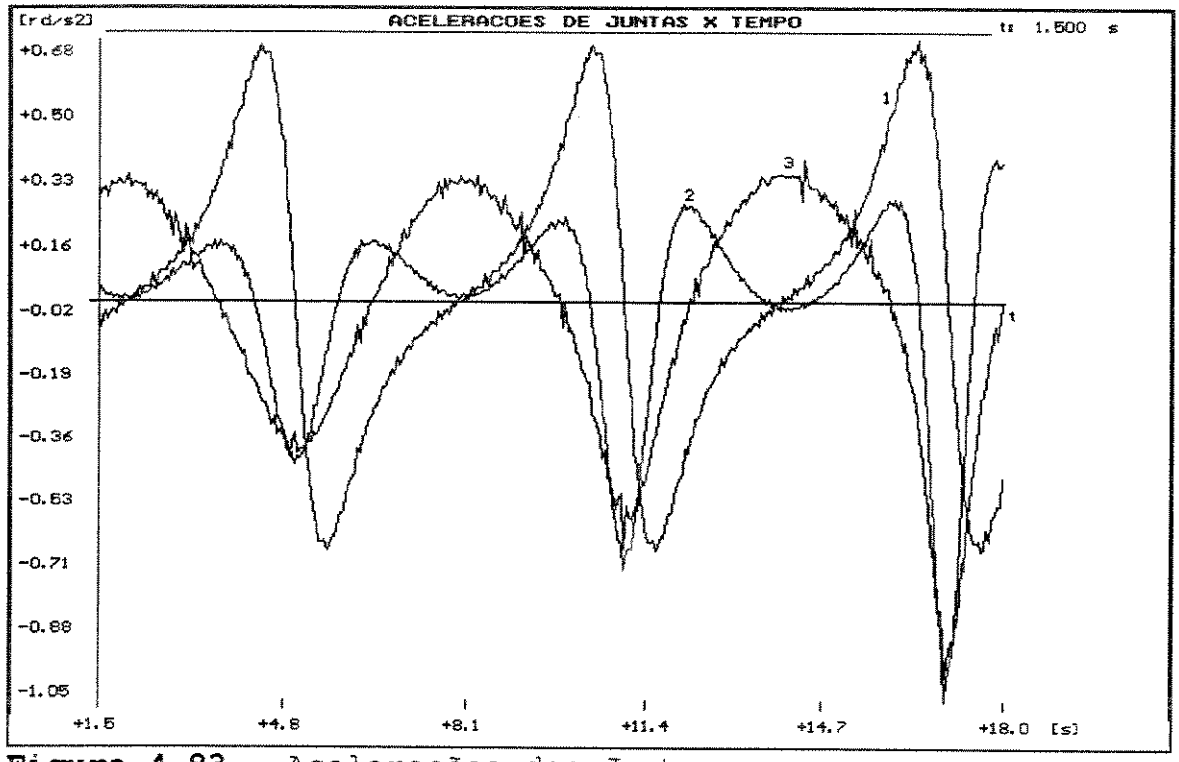


Figura 4.83 - Acelerações das Juntas.

No gráfico da **figura 4.83** onde estão apresentadas as evoluções das acelerações das juntas, fixamos o tempo de apresentação em 1,5s e 18s, para mostrar com mais detalhes a evolução durante o tempo de rastreamento. Ao início e final do movimento, entretanto, existem acelerações com amplitudes maiores (da ordem de 30rd/s²).

Note, observando a **figura 4.84**, que o erro de rastreamento esteve sempre dentro da precisão imposta. E para fazer mais uma comparação de desempenho da técnica, apresentamos na **figura 4.85** a evolução das respostas das juntas em posição, quando aplicamos a solução dada pela aplicação da modelagem cinemática inversa do robô. Veja que durante a etapa de rastreamento da trajetória, as respostas obtidas com a aplicação da técnica (**figura 4.81**) são praticamente idênticas. Nós notamos que conforme exigimos maiores precisões no rastreamento, mais os resultados se aproximam do ideal, que no caso é dado pela solução das equações cinemáticas inversas, ou seja:

● *Os resultados desta técnica, convergem para as soluções inversas analíticas, que podem ser uma outra aplicação possível. Essa conclusão é de extrema importância, porque, como vimos, ela pode ser aplicada para robôs com qualquer número de graus de liberdade (lembre que robôs com mais de seis graus de liberdade possuem necessariamente redundâncias, e suas soluções cinemáticas analíticas ficam muitíssimo mais complicadas; este é um agravante muito significativo, principalmente pela complexidade dos processos matemáticos*

envolvidos para determiná-las), e a técnica oferecería, pelo menos, uma solução.

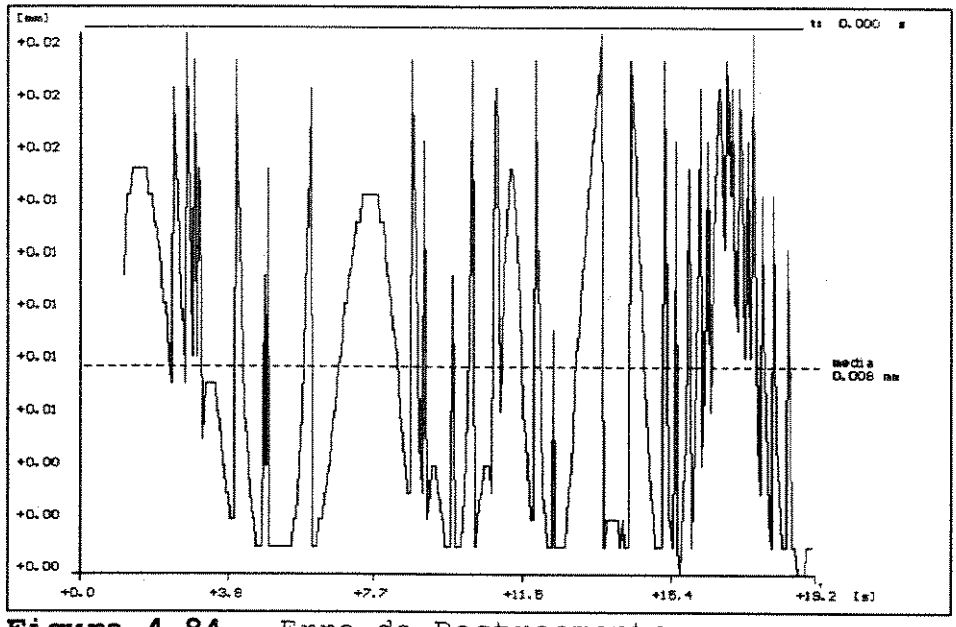


Figura 4.84 - Erro de Rastreamento.

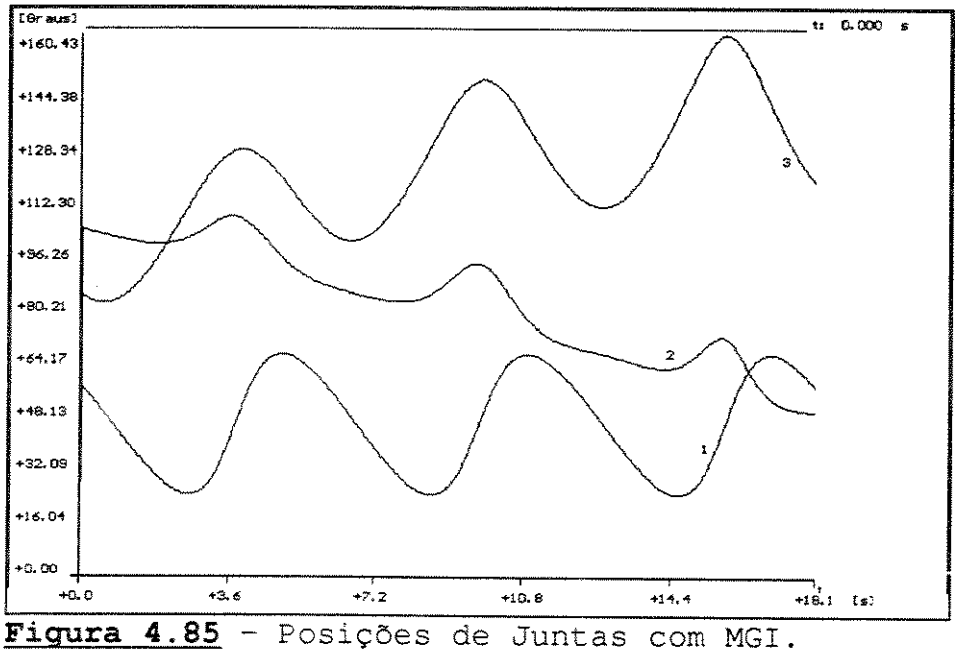


Figura 4.85 - Posições de Juntas com MGI.

Neste ponto também surge a idéia de incrementar os algoritmos com mais outros critérios heurísticos para selecionar entre soluções melhores ou piores, por exemplo, anexando critérios heurísticos de distâncias de aproximação/afastamento de pontos singulares da estrutura durante os movimentos, e procurando sempre realizar movimentos que evitem a passagem do robô por tais pontos; pode-se usar medidas de manipulabilidade

do robô para encontrar sua melhor postura frente ao rastreamento de cada trajetória **[87][88]**, incluir medidas de destreza (*dexterity measures*) para encontrar configurações ótimas da sua estrutura durante os movimentos **[36]**, medidas de mobilidade e de flexibilidade para encontrar soluções que permitam sempre a maior habilidade do robô trocar de direções de movimentos durante um rastreamento para aumentar sua capacidade de reação a fatores imprevistos como perturbações **[23]**, ou ainda utilizar critérios de movimentos com atenção para o controle das velocidades por análise dos elipsóides de velocidade obtidos através das posturas que o robô adota para rastrear as trajetórias **[16]**. Todas estas são possibilidades que esta técnica que estamos propondo pode permitir a fácil incorporação, e permitir que os robôs tenham desempenhos melhores, usando critérios para controlá-los tradicionalmente chamados de critérios inteligentes de controle de robôs.

♦ *CONCLUSÃO GERAL*

Após termos realizado uma grande quantidade de experimentos com a utilização desta técnica que estamos propondo para realizar o controle de trajetórias de robôs, verificamos que ela pode ser utilizada para fazê-lo em tempo-real, porque a quantidade de processamento computacional é mínima, e sem dúvida, pode dar enorme robustez para a questão do rastreamento de trajetórias que possuem equações matemáticas complexas, e mesmo que a estrutura física do robô sofra perturbações através da ação de forças externas durante os movimentos.

Constatamos que se os servomecanismos que controlam os movimentos de cada junta do robô realizarem a tarefa de seguimento adequado dos níveis de referências gerados por esta técnica, ela pode garantir muito boa suavidade nos movimentos, e excelente precisão de rastreamento. Chegamos a realizar experimentos exigindo precisões da ordem de centésimos de milímetros, e comprovamos que ela, mesmo assim, realiza um controle muito eficaz sobre a perseguição de pontos que compõe uma trajetória qualquer no espaço. E nestes experimentos onde fizemos exigência fortes de precisão, constatamos que as soluções encontradas pela técnica para rastrear as trajetórias, converge para a solução dada pela utilização de cálculos das equações cinemáticas inversas do robô. E esta técnica não utiliza estas equações em seus algoritmos. Além do mais, em testes utilizando o algoritmo mais complexo que propusemos processado por um computador tipo IBM-PC-AT 286, em um loop de controle, nunca ultrapassou os 60ms para precisões de centésimos de milímetros, e em um IBM-PC-AT 486 (50MHz) este tempo nunca ultrapassou os 4ms, fato que garante sua aplicação em tempo-real, para por exemplo, controlar uma enorme quantidade de motores de corrente contínua que podem ser utilizados para o acionamento dos servomecanismos das juntas.

A técnica baseia-se em procedimentos de busca heurística, como método de otimização dos movimentos, sempre visando alcançar posições espaciais que se acerquem tanto quanto possível de uma próxima posição objetivo a ser alcançada pelo ponto terminal do robô. Ela permite a anexação de vários critérios heurísticos que podem depender do tipo de tarefa a realizar, e do tipo de estrutura física do robô. Também concluímos que pode ser utilizada para o controle de trajetórias de robôs com qualquer número de graus de liberdade, inclusive para robôs com graus de liberdade redundantes, e se as posições objetivo a serem rastreadas

pelo robô estiverem dentro do seu volume de trabalho, existe a garantia de que a técnica conduza a pelo menos uma solução, para cada ponto que compõe a trajetória a ser rastreada. Verificamos também que para haver convergência a um ponto objetivo, não importa onde o robô se encontre posicionado, desde que o espaço onde a trajetória está definida esteja sem restrições (caso contrário, necessariamente deverão ser utilizadas outras medidas de sensores adequados para basear as tomadas de decisões).

Pelas características de ação da técnica durante os movimentos de rastreamento, resolvemos chamá-la de ***Técnica de Controle de Trajetórias Para Robôs Por Reflexos/Reação***.

CAPÍTULO V

PERSPECTIVAS FUTURAS

5.1 ☞ PLANO PARA A CONTINUAÇAO DA PESQUISA

Nas próximas etapas do projeto a ênfase será dada para a programação de tarefas, movimentos, planejamento de trajetórias, comunicação homem-máquina e organização de uma biblioteca de programas para um determinado conjunto de aplicações que requeiram mudança na estratégia de controle adotada.

Para a seqüência da pesquisa iremos instalar o planejador e controlador de trajetórias, integrando-lhe à SCHM para atuar em tempo-real, bem como desenvolver uma metodologia que permita classificar os campos de aplicações possíveis para o robô **JECAII**, definindo para cada campo discriminado a melhor estratégia de controle a ser adotada. Deverá se levar em conta que a estratégia a ser adotada para uma dada aplicação deve atender todos os requisitos de desempenho especificados e ao mesmo tempo deve ser a de menor custo de implementação, tanto no que diz respeito ao tempo de processamento dos algoritmos, como no *hardware* eletrônico e na estrutura mecânica a ser utilizada para cada caso. Pretende-se com isto que o robô **JECAII**, com característica modular, seja capaz de abranger a grande maioria das aplicações industriais.

A característica modular deste robô já vem sendo estudada pelos elementos da equipe e se classifica em três campos funcionais distintos:

- módulos de controle em *hardware* e *software*

- módulos de acionamento e atuadores

- módulos mecânicos

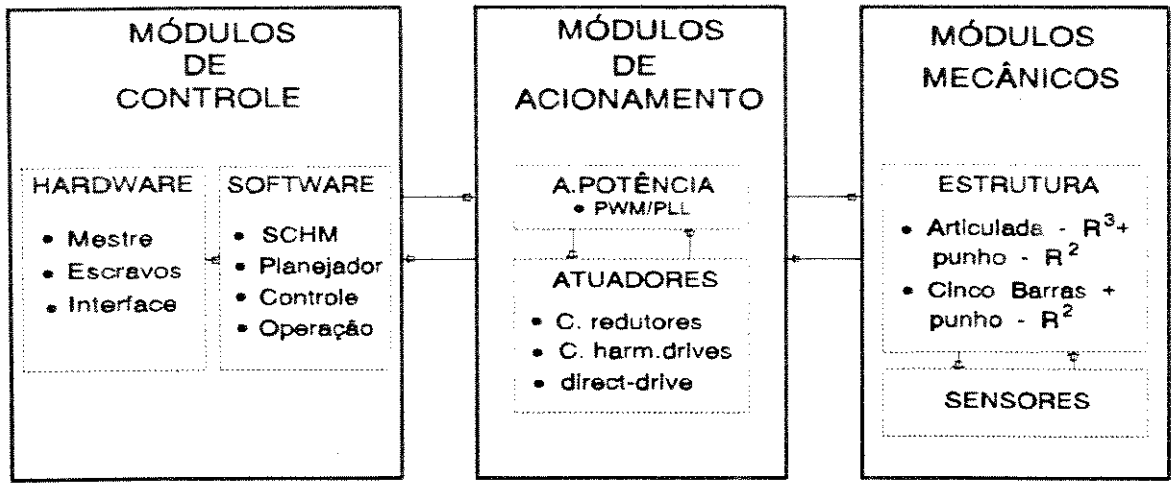


Figura 6.1 - Estrutura Modular do Robô **JECAII**.

5.1.1 ☞ Módulos de Controle Em *Software*

Iremos estruturar um sistema modular de *software* baseado em duas estratégias básicas de controle de trajetórias que utilizam o sistema de planejamento por busca heurística via inteligência artificial desenvolvido nesta tese de doutorado.

Levar-se-á em conta a característica multivariável, variante e não-linear do modelo dinâmico da estrutura a ser controlada para se estabelecer as estratégias a serem empregadas. Levar-se-á em conta também, que em alguns casos os acoplamentos e não linearidades do modelo multivariável podem ser vistos como servomecanismos de juntas isoladas com perturbações externas, e ainda, para a mesma estrutura e uma única aplicação, os graus de liberdade destinados ao posicionamento podem requerer uma estratégia de controle diferente daquelas requeridas para as juntas destinadas à orientação [40], supondo-se então que seja possível a adoção de uma estratégia para cada função na operação do rastreamento da trajetória.

Um estudo pormenorizado sobre as vantagens e desvantagens de uma estratégia sobre a outra, em cada campo de aplicação, deverá ser realizado. Criar-se-á um manual de auxílio que estará disponível no SCHM com todas as informações necessárias ao usuário a fim de que ele possa optar pela estratégia mais adequada para a operação que deverá realizar, informando inclusive se haverá necessidade de substituição de módulos do *hardware* para que o processamento da estratégia adotada seja bem sucedido.

5.1.2 ☞ Módulos de Controle em *Hardware*

Está em fase de projeto o desenvolvimento de controladores Escravos baseados nos microcontroladores *Intel-8051* que deverão permitir o aumento da capacidade computacional do sistema de controle do robô, e a diminuição do número de componentes

necessários, diminuindo o espaço físico necessário para fixá-los.

5.1.3 ☞ Módulos de Acionamento

- Acionamento *PLL/DUAL*: Em fase de projeto e estudo da viabilidade de sua aplicação em juntas que exigem rastreamento fino, com controle de fase **[50]**.

- Acionamento na forma *DIRECT-DRIVE* **[56]**: Através de um projeto de intercâmbio entre DSCE-UNICAMP e Escuela Técnica Superior de Ingenieros de Telecomunicación da UNIVERSIDAD POLITÉCNICA DE MADRID, aprovado e financiado pela CYTED-D do Ministério de Educación y Ciencia de España, estamos iniciando um estudo em conjunto sobre as influências no comportamento dinâmico do efeito dos acoplamentos dos servomecanismos de juntas isoladas acionadas por esta técnica. Estes módulos serão somente incluídos no sistema numa etapa futura do projeto, através de financiamento, pleiteado junto à FAPESP, para a compra dos motores de torque.

5.1.4 ☞ Módulos Mecânicos

- Projeto em desenvolvimento de uma estrutura para robô de cinco barras - R^3 + punho-R^2 + garra.

5.2 ☞ METODOLOGIA

A metodologia a ser adotada para a continuação da pesquisa prevê a realização de seis etapas conforme apresentamos a seguir.

Na primeira etapa se fará uma pesquisa bibliográfica para se obter um agrupamento, por campos, das aplicações que podem ser tratadas com a mesma estratégia de controle. Levar-se-á em consideração a demanda atual na indústria nacional, bem como na indústria internacional acompanhada de uma avaliação das tendências futuras no parque nacional, supondo-se uma retomada dos investimentos, ora estagnados em todos os setores do país. Nesta pesquisa será feito um levantamento nos setores com maiores chances de serem automatizados ou sofrerem um processo de renovação do sistema atual já automatizado. Dentre eles destacamos:

- Setor Automobilístico
- Setor Têxtil
- Setor Plástico
- Setor Eletro-Eletrônico
- Acreditamos também que o setor agrícola tenderá, num futuro próximo, a sofrer um processo de automatização que demandará a utilização da tecnologia proposta neste trabalho.

A relação custo/benefício merecerá atenção sobre dois aspectos fundamentais:

- Primeiro com relação aos aspectos sociais, que podem nos levar a concentrar esforços para selecionar aplicações que procuram a utilização do robô como um meio de liberar o homem das tarefas hostis e danosas para a sua saúde, não deixando de levar em cosideração que outras tarefas realizáveis pelo homem, sem ofender as suas limitações, também podem ser transferidas ao robô na medida em que as vantagens econômicas, tais como, redução do custo final do produto, melhoria da qualidade do produto, etc..., venham de encontro a uma melhoria da qualidade de vida da população, se isto não acarretar entretanto, aumento do desemprego. Uma reflexão sobre estes aspectos deverá ser feita levando-se em consideração que já existem mecanismos em países com parques industriais fortemente automatizados, que procuram disciplinar a questão com a preocupação de proporcionar treinamentos aos homens substituídos pelos robôs, para passarem a executar tarefas mais nobres.

- Segundo com relação aos aspectos puramente técnicos, onde levaremos em conta que tarefas simples devem ser executadas por máquinas simples, controladas com estratégias simples, e caso alguma junta do robô exija um tratamento especial, devido a operações em regiões críticas no seu espaço de parâmetros, o seu módulo de controle pode ser mais sofisticado que os das demais juntas, para que tenha a capacidade de processar algoritmos de controle mais sofisticados, como por exemplo: O Escravo de uma junta crítica pode ser baseado num microprocessador de 16bits e relógio de 33MHz para permitir o processamento de algoritmos mais poderosos que atendam especificações de alta resolução, precisão, robustez, etc., ao mesmo tempo que nas demais juntas, menos críticas, possam ser utilizados Escravos baseados em microprocessadores com 8bits e relógio de 8MHz, desde que o Mestre tenha configuração compatível e o barramento compartilhado seja projetado para permitir a comunicação entre o Mestre e os Escravos distintos, como os citados no exemplo.

Em seguida, já na segunda etapa, começa o trabalho de aplicação em laboratório, e incorporação dos conhecimentos obtidos no sistema real do robô **JECAII**. Esta seqüência de trabalho começa com a preparação dos módulos de *software* para a SCHM, com o objetivo de tornar o sistema global de controle amigável ao usuário, mesmo que este não tenha o conhecimento específico e detalhado de cada uma das partes que o compõem. O projeto do pacote de *software* continuará levando em consideração que o mesmo deverá oferecer riqueza de apresentações gráficas e que através dele seja possível ter-se acesso a utilização de todos os recursos de programação e de *hardware* que o sistema do robô ofereça, e que seja possível realizar simulações baseadas em dados reais de operação em plantas industriais e ainda auxilie o usuário a realizar a programação do sistema para novas aplicações.

Na terceira etapa, será posto em prática a teoria para o planejamento e a supervisão do rastreamento de trajetórias espaciais para robôs, baseada em técnicas de inteligência artificial e desenvolvida durante a evolução desta tese de doutoramento, já com o objetivo de compatibilizar a filosofia de construção de robôs modulares apresentada.

A quarta e a quinta etapas consistem da preparação dos módulos de *software*, para tornar viável a realização de diferentes estratégias de controle do robô, baseadas nas exigências de cada aplicação, e da modularidade da arquitetura. Em função destas estratégias serão criados os programas modulares de controle, de forma que sua

utilização possa ser feita através do acesso a bibliotecas padrões, que conterão os algoritmos básicos de cada técnica de controle disponível formando os módulos. Estes módulos estarão prontos para serem utilizados em qualquer momento, para uma dada aplicação, e as alterações em suas características variáveis serão feitas através da SCHM com auxílio de programação oferecida pelo mesmo, permitindo a fácil seleção de ítens, e entrada de dados pelo teclado do Mestre, e assim facilitando o trabalho do usuário.

Estando então todas estas partes integrantes do sistema do robô modular completas, será feito um manual detalhado de auxílio à programação e utilização de todos os recursos para sua operação, que deverá estar disponível ao usuário no SCHM para a escolha dos módulos adequados à aplicação requerida, bem como para a programação das tarefas a serem executadas.

5.3 ☞ IDÉIA PARA AUMENTAR AS POTENCIALIDADES DO MÉTODO DE BUSCA HEURÍSTICA - SEQÜÊNCIA DA PESQUISA

A técnica que propusemos para o controle de trajetórias de robôs, utilizando um método de busca heurística baseado no algoritmo A*, modificado para aplicação em tempo-real, demonstra-se muito eficiente para a solução do problema de posicionamento espacial, com excelente desempenho no rastreamento, mesmo para trajetórias definidas por equações de grande complexidade, inclusive proporcionando movimentos bastante suaves. E devido a característica inerente a este método de busca, permite recuperação de desvios de rastreamento provocados por eventuais perturbações na estrutura dos robôs durante seus movimentos.

O critério heurístico utilizado para os experimentos foi a distância euclidiana definida entre o ponto extremo do terceiro elo do robô e cada ponto que pertence a trajetória requisitada para o rastreamento, objetivando-se sempre que esta distância tenda ao mínimo possível, ou convirja para a precisão requisitada para tal rastreamento. No entanto, para que esta técnica possa ter sua aplicação ampliada ao problema global de rastreamento de trajetórias, é necessário adaptá-la ao problema conjunto de encontrar soluções tanto para posição como para orientação espacial.

Sabe-se que para um robô estar apto a perseguir uma trajetória qualquer no espaço (com orientação adequada), ele deve estar dotado de no mínimo 3 graus de liberdade para o posicionamento, e mais 3 graus para a orientação, com geometria estrutural adequada. Como o **JECAII** possui somente dois graus de liberdade para a orientação, logicamente existem algumas limitações para a solução geral deste problema, entretanto, com sua geometria é possível resolver uma enorme quantidade de problemas desta natureza.

Neste caso faz-se necessária a utilização do modelo cinemático direto completo do robô, porisso primeiramente apresentamos este modelo fazendo o desenvolvimento baseados na técnica de *Denavit* e *Hartenberg* **[20]** para a obtenção dos parâmetros das matrizes de transformações homogêneas entre os elos de sua estrutura.

5.3.1 ☞ Modelo Cinemático Direto Completo do JECAII - Método de *Denavit* e *Hartenberg*

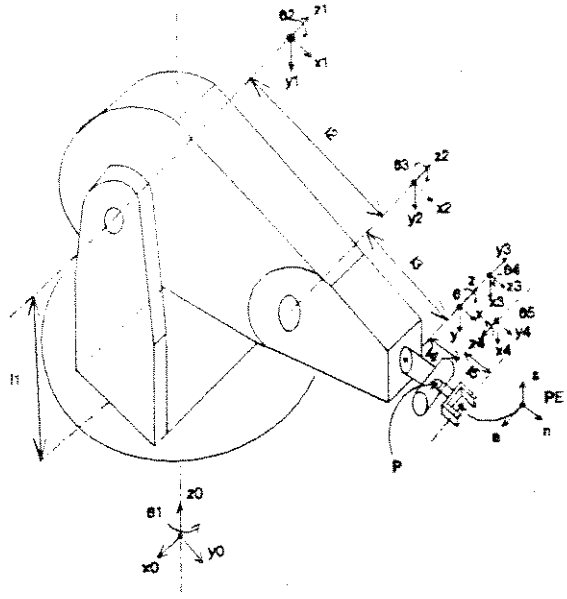


Figura 5.1 - Fixação dos Sistemas de Coordenadas Conforme *Denavit* e *Hartenberg*. Robô **JECAII**.

Observando a **figura 5.1**, podemos obter facilmente os quatro parâmetros de *Denavit* e *Hartenberg* para cada sistema de coordenadas fixados aos eixos de movimentos da estrutura. Estes parâmetros estão mostrados na **Tabela 5.1**.

Junta i	$\theta_{(i)}$	$\alpha_{(i)}$	$a_{(i)}$	$d_{(i)}$
1	θ1	-90°	0	l1
2	θ2	0°	l2	0
3	θ3	90°	0	0
4	θ4	90°	0	l3+l4
5	θ5	0°	l5	0

Tabela 5.1 - Parâmetros de *Denavit* e *Hartenberg* para Robô **JECAII**.

Substituindo estes parâmetros da **Tabela 5.1** na matriz **(5.1)** podemos obter todas as matrizes de transformações homogêneas entre os elos i-1 e i, i=1,...5.

$$ {}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i).sen(\theta_i) & sen(\alpha_i).sen(\theta_i) & a_i.\cos(\theta_i) \\ sen(\theta_i) & \cos(\alpha_i.\cos(\theta_i) & -sen(\alpha_i).\cos(\theta_i) & a_i.sen(\theta_i) \\ 0 & sen(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(5.1)} $$

Ou seja:

$$ {}^{0}T_{1} = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{1}T_{2} = \begin{bmatrix} S_2 & C_2 & 0 & 12.C_2 \\ -C_2 & S_2 & 0 & -12.S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}T_{3} = \begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}T_{4} = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & -C_4 & 0 \\ 0 & 1 & 0 & 13+14 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{4}T_{5} = \begin{bmatrix} C_5 & -S_5 & 0 & 15.C_5 \\ S_5 & C_5 & 0 & 15.S_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

onde: S_n= sen(θn) e C_n= cos(θn).

Assim pode-se ter conhecimento da posição e orientação do elemento terminal do **JECAII** em relação ao sistema de coordenadas de referência fixado na sua base, apenas realizando a multiplicação sequêncial destas matrizes homogêneas intermediárias: $T = {}^{0}T_{1}.{}^{1}T_{2}.{}^{2}T_{3}.{}^{3}T_{4}.{}^{4}T_{5}$.

Então:

$$ T = \begin{bmatrix} (C_1S_{23}C_4 - S_1S_4)C_5 - C_1C_{23}S_5 & (S_1S_4 - C_1S_{23}C4)S_5 - C_1C_{23}C_5 & C_1S_{23}S_4 + S_1C_4 & ----- \\ (S_1S_{23}C_4 + C_1S_4)C_5 - S_1C_{23}S_5 & -(C_1S_4 + S_1S_{23}C_4)S_5 - S_1C_{23}C_5 & S_1S_{23}S_4 - C_1C_4 & ---- \\ C_{23}C_4C_5 + S_{23}S_5 & S_{23}C_5 - C_{23}C_4S_5 & C_{23}S_4 & ---- \\ 0 & 0 & 0 & ---- \end{bmatrix} $$

$$ \begin{bmatrix} ---- & 12.C_1C_2 - (13+14).C_1C_{23} - 15.C_1C_{23}S_5 + 15.(C_1S_{23}C_4 - S_1S_4)C_5 \\ ---- & 12.S_1C_2 - (13+14).S_1C_{23} - 15.S_1C_{23}S_5 + 15.(S_1S_{23}C_4 + C_1S_4)C_5 \\ ---- & 11 + 12.C_2 + (13+14).S_{23} + 15.S_{23}S_5 + 15.C_{23}C_4C_5 \\ ---- & 1 \end{bmatrix} $$

Se fizermos:

$$T = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} nx & sx & ax & px \\ ny & sy & ay & py \\ nz & sz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

onde:

n= vetor de ataque da garra (direção normal à palma da garra).
s= vetor normal da garra (ortogonal à palma da garra).
a= vetor transversal da garra (direção da abertura e fechamento da garra.
p= vetor de posição da garra.

então:

$$nx = (C_1 S_{23} C_4 - S_1 S_4) C_5 - C_1 C_{23} S_5$$
$$ny = (S_1 S_{23} C_4 + C_1 S_4) C_5 - S_1 C_{23} S_5$$
$$nz = C_{23} C_4 C_5 + S_{23} S_5$$

$$sx = (S_1 S_4 - C_1 S_{23} C4) S_5 - C_1 C_{23} C_5$$
$$sy = -(C_1 S_4 + S_1 S_{23} C_4) S_5 - S_1 C_{23} C_5$$
$$sz = S_{23} C_5 - C_{23} C_4 S_5$$

$$ax = C_1 S_{23} S_4 + S_1 C_4$$
$$ay = S_1 S_{23} S_4 - C_1 C_4$$
$$az = C_{23} S_4$$

$$px = 12.C_1 C_2 - (13+14).C_1 C_{23} - 15.C_1 C_{23} S_5 + 15.(C_1 S_{23} C_4 - S_1 S_4) C_5$$
$$py = 12.S_1 C_2 - (13+14).S_1 C_{23} - 15.S_1 C_{23} S_5 + 15.(S_1 S_{23} C_4 + C_1 S_4) C_5$$
$$pz = 11 + 12.C_2 + (13+14).S_{23} + 15.S_{23} S_5 + 15.C_{23} C_4 C_5$$

Para resolver somente o problema do rastreamento de trajetórias em posição, vimos que utilizando o critério heurístico de busca baseado no objetivo de eleger combinações de movimentos de juntas que sempre provoquem uma diminuição da distância euclidiana entre o lugar atual em que se encontra o elemento terminal do robô e o próximo ponto que deve ser atingido em função da exigência feita pela trajetória a ser rastreada, no tempo, pode-se conseguir que o robô faça movimentos muito eficientes. Mas para resolver conjuntamente os problemas de rastreamento em posição e orientação, existe uma dificuldade maior para escolher um critério heurístico adequado para servir como base para as decisões inteligentes que o sistema de controle de trajetórias deve fazer. Isso se deve ao fato que neste último caso, o critério heurístico de diminuição de distâncias euclidianas não pode ser obtido diretamente através do cálculo de distâncias entre vetores, mas sim de distâncias entre matrizes.

Neste caso, a trajetória que deve ser rastreada também deve ser definida em termos de posição e orientação que o robô deverá atingi-la. E isso pode ser feito através de sua digitalização em pontos definidos por matrizes de transformações homogêneas.

A matriz de transformação homogênea que representa as transformações de coordenadas da cadeia completa formada pela estrutura do robô, define completamente como progridem as posições e orientações do seu elemento terminal em função dos movimentos das juntas, e referenciadas ao sistema de coordenadas fixado como o sistema de base inercial do robô.

Se a trajetória a ser rastreada for definida sob estas condições e também referenciada ao sistema de base inercial do robô diretamente, ou por alguma transformação de coordenadas conveniente, então podemos criar um critério heurístico que permita produzir um algoritmo de controle de trajetórias nas mesmas bases que o fizemos para o controle de rastreamento em posição, e que ao ser executado produza ações (movimentos de juntas) no sentido de provocar que a matriz homogênea que define a posição e orientação nas quais se encontra o elemento terminal do robô, sempre se aproxime de cada matriz homogênea definida pela discretização da trajetória a ser rastreada em função da evolução temporal.

Como é conhecido da teoria de matrizes, sempre é possível calcular normas para elas em função dos elementos que a compõem. E com este conceito pode-se obter um modo de verificar "semelhanças/diferenças", ou mesmo conseguir-se obter medidas de "distâncias" entre elas.

Existem algumas normas de matrizes mais usuais que podemos utilizar para isso, como por exemplo estas que apresentamos a seguir:

Uma função $\| \ \|$ em $C^{m \times n}$ é chamada de norma matricial se:

$$\|A\| \geq 0, \quad \|A\| = 0 \quad somente\ se\ A = 0 \qquad \textbf{(p1)}$$

$$\|\alpha A\| = |\alpha| \|A\| \qquad \textbf{(p2)}$$

$$\|A + B\| \leq \|A\| + \|B\|, \quad para\ todo\ A, B \in C^{m \times n}, \quad \alpha \in C \qquad \textbf{(p3)}$$

Se ainda:

$$\|AB\| \leq \|A\|\|B\| \qquad \textbf{(p4)}$$

quando o produto matricial AB é definido, então $\| \ \|$ é chamada de ***norma Multiplicativa***

A seguir apresentamos algumas outras definições de normas que possuem as propriedades **(p1)** a **(p4)**.

1. $\left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\right)^{1/2} = (\text{traço } A^TA)^{1/2}$ ***norma Euclidiana***

3. $\|A\| = \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|$

2. $\text{máx}\{|a_{ij}|: i = 1,\ldots,m;\ j = 1,\ldots,n\}$

4. $\|A\| = \text{máx}\left\{\sqrt{\lambda} : \lambda \text{ autovalor de } A^TA\right\}$ ***norma Espectral***

Em alguns experimentos preliminares de simulação, utilizando estas idéias, e as normas euclidianas de matrizes, obtivemos resultados promissores. Estes experimentos foram baseados nos seguintes procedimentos:

Passo 1. Localizamos a posição e a orientação atual do elemento terminal do robô e atualizamos a respectiva matriz de transformação homogênea que faz esta representação matemática (matriz *A*).

Passo 2. Atualizamos uma matriz (matriz *B*) com os coeficientes que representam a posição e a orientação que o robô deve atingir a próxima meta exigida pela trajetória a ser rastreada.

Passo 3. Calculamos $d_{\pi/2}$, $d_{-\pi/2}$, e d_{π} para cada junta, supondo esses seus respectivos movimentos de $\pi/2$, $-\pi/2$, e π, relativos à posição na qual se encontra. Onde:

$$d_\theta = \left(\sum_{i=1}^{4}\sum_{j=1}^{4}|a_{ij} - b_{ij}|^2\right)^{1/2}$$

Passo 4. Encontramos o "melhor" passo para o movimento de cada junta (α^*) com o auxílio da **equação (3.14)** (veja **Capítulo III**), no sentido dela dar sua "melhor" contribuição para diminuir *d*.

Passo 5. Uma vez encontrados os "melhores" passos para cada junta, conforme *passo 4*, colocamos em prática estes movimentos atualizando a matriz *A*, através da utilização do modelo cinemático direto e dos novos ângulos assumidos por cada uma.

Passo 6. Se com esta nova atualização feita no *passo 5*, o heurístico *d* ainda tiver maior amplitude que a precisão exigida pela trajetória, então retornamos ao *passo 1*. Senão, atualizamos a matriz *B* com os coeficientes que representam a próxima meta por ela definida, voltamos ao *passo 1*, e assim por diante, até que o robô tenha feito o seu completo rastreamento.

Como o **JECAII** possui somente dois graus de liberdade para a orientação da sua garra, sempre um dos vetores *n*, *s* ou *a* deverá ser ortogonal ao plano definido pelos outros dois para que estes procedimentos funcionem. Lembrar que para posicionar e orientar completamente o elemento terminal de um robô, no espaço livre, são necessários, no mínimo três graus de liberdade para posicioná-lo, e mais três para orientá-lo.

5.4 ☞ PERSPECTIVAS POR IDÉIAS POUCO ORTODOXAS

Levando esta restrição em consideração, verificamos que o processo converge para uma solução, ou seja, o elemento terminal do robô tende a buscar as metas exigidas pelas trajetórias. Entretanto, a maneira como progride esta convergência ainda não nos está clara. Fatores como as relações de grandeza entre os elementos das matrizes *A* e *B* e as dimensões dos elos da estrutura, influem de forma muito acentuada na velocidade de convergência das buscas e nas posturas adotadas por ela.

Acreditamos que possam existir formas de normalização entre os elementos destas matrizes que possam ser utilizadas como outros critérios heurísticos para que os movimentos tendam a fornecer soluções ótimas. E que inclusive possam ser "aprendidos" pelo sistema de controle em função das "observações" sobre como a estrutura está se comportando ao realizar os movimentos. Este "aprendizado" seria conseguido através da atualização de valores de ponderação associados com cada elemento das matrizes *A* e *B*. Se for realmente possível encontrar este modo de ponderar seus elementos em função dos dados obtidos através do funcionamento do robô e de critérios heurísticos como estes que acabamos de abordar, faremos com que ele seja controlado muito eficientemente através de um equacionamento matemático extremamente simples, que poderá permitir por exemplo, que ao "saber" como deve se "comportar" ao rastrear uma trajetória reta, "saberá", ou terá base, para como se "comportar" ao rastrear outra trajetória reta com parâmetros diferentes.

Aparentemente existe regularidade de escala dentro do volume de trabalho do robô, e sua geometria de construção funciona como uma transformada entre espaços de dimensões diferentes → Se representarmos a evolução de suas variáveis de juntas em gráficos com eixos de progressão não-linear, tipo sen, cos, etc..., conforme alguma conveniência, em função da geometria da sua estrutura, ao ser posto para rastrear trajetórias não-lineares de mesma família de equacionamento, e com escalas variadas,

pode ser possível diminuir o grau das não-linearidades destas curvas, e até mesmo linearizá-las. Esse fato pode contribuir enormemente para a tarefa de "aprendizagem", e em função da simplicidade da abordagem, poder permitir que a capacidade de processamento computacional do sistema de controle tenha maior liberdade para atuar em tempo-real.

Com este modo de abordar, estamos tentando resolver o problema de controle de trajetórias através de uma "*intuição geométrica*" afastando-nos um pouco da conotação matemática formal e tradicional que o problema pode ter. Se observarmos um pouco mais atentos o modo como este tipo de técnica conduz os movimentos do robô, notaremos que seu elemento terminal se move submetido a ações de "*zig-zag*" (veja *zoom's* das **figuras 4.24, 4.37, 4.50, e 4.65**), esses movimentos, é claro, podem ser infinitesimalmente pequenos, ou muito grandes, dependendo da intensidade da ação, da geometria da sua estrutura, etc... Portanto, para avaliar a dimensão das distâncias percorridas, ou a percorrer entre tramos, o uso da geometria euclidiana pode não ser o mais conveniente. É possível que o interesse na trajetória não seja diretamente sua direção, senão a distribuição dos seus ziguezagues. Essa outra idéia faz perceber que pode ser interessante pesquisar e utilizar a geometria dos fractais para encontrar critérios heurísticos que aumentem mais a eficiência desta técnica.

ANEXO A

DETALHAMENTO DO HARDWARE ELETRÔNICO DO JECAII

A.1 ☞ MICRO MESTRE

O Micro Mestre utilizado possui uma CPU tipo 80386DX com co-processador 80387DX, com controladora de monitor padrão EGA, portas serial e paralela, teclado *enhancement*, disco rígido de 40*Mbytes* e flexíveis de 1,2*Mbytes* e 1,44*Mbytes*.

Ele forma o nível hierarquicamente superior da arquitetura proposta, responsável pela supervisão geral do sistema do robô. Nele ficam residentes os pacotes de *software* responsáveis pelo Sistema de Comunicação Homem-Máquina (SCHM), que incluem as facilidades de programação dadas aos usuários, como permissão de desenvolvimento de programas para controle, simulações, acesso via teclado aos recursos disponíveis do sistema, instruções de ajudas para a programação, e controle de saídas gráficas. Durante as execuções de tarefas é ele quem coordena o funcionamento dos Escravos, fazendo as devidas sincronizações e tranferências de dados para que o robô se movimente de forma correta.

Ele possui uma memória base de 640*Kbytes* disponível para usuários, mas em nosso projeto liberamos os 128*Kbytes* superiores desta memória para utilizar como mémoria para comunicação com os Escravos. Permitindo uma grande facilidade para as operações de leitura e escrita em suas memórias, pois para fazer este acesso basta proceder como se a intenção fosse ler ou escrever na memória do próprio Mestre, com instruções simples e rápidas de programação. O que fazemos é chavear esta memória base do Mestre para 512*Kbytes* sempre que o sistema é posto para operar o controle do robô, e o demais é feito pelo circuito de interface do PC com os Escravos, que está projetado para reconhecer os dados que são enviados para estes 128*Kbytes* entre os 512*Kbytes* e os 640*Kbytes* da memória base. O circuito de interface citado é colocado num dos *slots* padrão AT da placa *Mother Board* do Mestre.

A utilização de um microcomputador com estas características para a função de Mestre dada foi intencional, porque como existe uma padronização universal, existe garantia de que os novos avanços dados nesta tecnologia poderão ser facilmente adaptados ao robô, tanto em *software* como em *hardware*, poder-se-á substituir esta CPU por outra com processamento mais potente. É evidente que sempre ao realizar câmbios em um projeto existe a necessidade de reavaliação de todas as partes que o compõe, e provavelmente uma substituição assim exigiria também a adaptação dos demais componentes da arquitetura para trabalharem com capacidade compatível, mas isso não impede que este projeto possa ser sempre atualizado com os avanços tecnológicos conseguidos.

Ainda sobre a arquitetura de *hardware* destes tipos de máquinas citamos a referência **[25]** onde é feita uma descrição minuciosa de suas características e potencialidades. E sobre os aspectos de programação veja o **Anexo C**.

A.2 ☞ INTERFACE DE COMUNICAÇÃO ENTRE O MESTRE E OS ESCRAVOS

Este circuito de interface permite ao Mestre realizar o acesso completo aos Escravos, foi dada a possibilidade de acesso direto a 32*Kbytes* de memória de cada Escravo, e pode-se utilizar dois modos de operação distintos para isso, um modo que chamamos de real, onde o Mestre opera no modo de endereçamento real da CPU, e os endereços são formados pela união do endereço que cada Escravo tem no sistema e os 16*bits* menos significativos do barramento de endereços do Mestre. Como destinamos os 128*Kbytes* superiores da memória base para usuários do Mestre, os endereços de memória válidos para esta finalidade estão compreendidos entre Ø8ØØØØh e Ø9FFFFh. E o outro modo possível é o avançado, onde o Mestre opera no modo de endereçamento protegido da CPU utilizando a faixa de endereços entre 5ØØØØØh e 5FFFFF. Este circuito de interface também permite que o Mestre acesse os dispositivos de entradas e saídas (E/S) dos Escravos, para isso basta gerar o endereço do Escravo ao qual se deseja acessar, esta geração automaticamente provoca uma interrupção no processamento do circuito em questão, com a liberação dos seus barramentos de dados e endereços, e o Mestre pode realizar as operações de E/S como se o fizesse com seus próprios dispositivos de E/S **[21]**.

Devido ao fato deste circuito possuir muitas funções importantes dentro da arquitetura que propusemos, seu projeto tem bastante complexidade. Nós o dividimos em sete blocos que juntos compõem uma montagem em uma placa padronizada para ser inserida num dos *slots* do Mestre, e chamamos estes blocos respectivamente por módulo decodificador de endereços de E/S, módulo amplificador de barramento, módulo gerador de habilitação de memória e *reset*, módulo gerador de estado de espera, módulo temporizador, módulo gerador de BUSREQ e endereços das placas, e módulo habilitador do barramento de dados.

Nas explicações a seguir utilizamos algumas siglas para referir-nos às linhas físicas do *hardware* e aos sinais digitais presentes nelas, procuramos seguir a padronização universal utilizada em circuitos eletrônicos baseados em microprocessadores da família *8080*. O esclarecimento minuscioso dos seus significados pode ser encontrado na referência **[25]**.

O módulo decodificador de endereços de E/S é responsável pela geração dos sinais de leitura e escrita utilizados pelos Escravos, e dos sinais de habilitação de seus próprios dispositivos de E/S. Os sinais de escrita e leitura para os dispositivos de E/S dos Escravos no barramento são gerados quando houver, respectivamente, um sinal IOW ou IOR, e um endereço entre 12Øh e 13Fh no barramento do Mestre.

O módulo amplificador de barramento destina-se a amplificação dos sinais de endereços e dados originados no barramento do Mestre, este bloco foi inserido como medida de proteção contra sobrecargas para não danificar o Mestre por causa de eventuais problemas com os circuitos dos Escravos. Os *bits* de endereços SAØ e SA15 são amplificados através de amplificadores permanentemente habilitados. As linhas SA16 até SA19 são habilitadas através de um sinal gerado por programação, os *bits* que transitam por estas linhas formam os endereços dos Escravos no barramento, e somente podem ser transmitidos para a placa de interface se o Mestre estiver operando no modo de endereçamento protegido. No modo de endereçamento real, as linhas SA16 até SA19 são

geradas pela própria placa de interface, então somente os endereços formados pelas linhas de SAØ até SA15 são gerados pelo Mestre. Os dados formados pelas linhas SDØ até SD7 são amplificados através de um amplificador bidirecional, e a direção do fluxo de dados é controlada pelo sinal IOR do barramento do Mestre e pelo módulo habilitador do barramento de dados.

O módulo gerador de habilitação de memória e *reset* é responsável pela geração dos sinais de MEMRD e MEMWR necessários para as operações de leitura e escrita nos Escravos, e de RESET, a partir dos sinais das linhas de SMEMR, SMEMW e RESET do Mestre quando este módulo estiver habilitado. Existe uma inversão do sinal de RESET porque este sinal é habilitado em nível alto pelo Mestre e em nível baixo nos Escravos. Se o Mestre estiver operando no modo de endereçamento real, os sinais de MEMRD e MEMWR são gerados quando ele enviar um sinal de SMEMR ou SMEMW e o modo real for habilitado na placa de interface via programação, entretanto, se o Mestre estiver operando no modo de endereçamento protegido, é necessário que além dos sinais SMEMR ou SMEMW, o sinal da linha SA22 que é o vigésimo terceiro *bit* de endereços do Mestre esteja ativo, pois neste caso é utilizado o quinto *megabyte* do modo protegido da sua memória, e além disso, o sinal BALE deve estar ativo e o sinal AEN deve estar em nível lógico baixo para indicar que o endereço atual é válido.

O módulo gerador de estado de espera foi inserido para provocar um aumento do número de ciclos de máquina para as instruções de E/S geradas pelo Mestre, este procedimento permite que se possa realizar comunicação com dispositivos mais lentos, e foi dada a possibilidade de gerar até seis ciclos adicionais de relógio por instrução, apenas fazendo-se a seleção adequada de *jumpers* colocados no circuito para esta função. Isso pode garantir que possamos utilizar uma CPU com maior capacidade de processamento, com relógio de mais freqüência, mantendo os mesmos Escravos, apenas adequando os tempos de espera.

O módulo temporizador tem a função de gerar sinais de temporização para o barramento de comunicação. Com ele incorpora-se a possibilidade de sincronizar a operação de todos os Escravos, e é composto por um gerador de relógio, um temporizador e um amplificador, sendo que o gerador de relógio é baseado em um circuito oscilador com cristal de 4MHz, e em sua saída o sinal tem a freqüência dividida através de um *flip-flop* tipo JK operando em modo *toggle*; o temporizador pode ser programado através dos sinais de três linhas presentes no barramento de comunicação especialmente colocadas para isso, chamadas de TEMP1, TEMP2 e TEMP3. Este circuito de temporização é composto por três temporizadores, e para programá-lo existe um procedimento seqüêncial de quatro etapas, onde deve-se primeiramente escolher qual dos três temporizadores será programado, injetar a palavra de controle informando sobre que tipo de programação ele receberá, e após carregar o *byte* menos significativo e o mais significativo, respectivamente nesta ordem. E o amplificador fica permanentemente habilitado para garantir que os sinais de saída sejam capazes de manter os níveis corretos no barramento de comunicação.

O módulo gerador de BUSREQ e endereços dos Escravos tem a função de gerar até 10 sinais de requisição de barramento que denominados de BUSREQ1 até BUSRE1Ø. Estes sinais são utilizados para que o Mestre possa interromper momentaneamente o processamento dos Escravos para realizar transferências de dados para suas memórias, ou ler dados de suas memórias; Os seis primeiros desses sinais são respectivamente, um para cada Escravo, e os demais estão disponíveis para que possa-se eventualmente aumentar a capacidade do sistema inserindo outros circuitos no mesmo barramento de comunicação, que também possam ser controlados com o auxílio destes

sinais. Existe um artifício colocado neste módulo para que estes dez sinais possam ser gerados corretamente, isso foi necessário, porque para operação do Mestre no modo de endereçamento real os sinais de SA16 até SA19 são utilizados na decodificação dos endereços válidos para instruções de memórias dos Escravos, mas a quantidade da memória que pode ser acessada utilizando-se somente os endereços válidos, ou seja 128*Kbytes*, é inferior ao necessário para dez Escravos com 32*Kbytes* de memória cada (320*Kbytes* de memória total), assim utilizamos também os *bits* de endereços de ADR16 até ADR19 e os sinais que indicam em qual modo o Mestre está operando para resolver o problema. Este circuito é baseado num *chip* PIA 8255, onde todas suas três portas são programadas como saídas, e assim todos estes sinais podem ser gerados convenientemente, podendo ter seus níveis variados através de programação de alto nível.

O módulo habilitador do barramento de dados é responsável pela geração do sinal de habilitação do barramento bidirecional de dados. Este sinal é ativado em nível lógico baixo, e deve estar presente sempre que o Mestre necessita acessar os endereços de memória ou de E/S dos Escravos. Para o acesso aos endereços de E/S dos Escravos presentes no barramento de comunicação, o Mestre suspende a operação das CPU's destes subsistemas e controla estes endereços como se fossem seus próprios dispositivos.

Os sinais entre esta placa de interface de comunicação e o circuito físico onde está o barramento de comunicação com os conectores para conexão dos Escravos, são transmitidos por meio de linhas físicas especiais tipo *flat cable*, e como os sinais podem ser distorcidos devido a perdas em sua extensão que é de aproximadamente dois metros, também projetamos um sistema de amplificação que garante a integridade dos sinais que aparecem nesses conectores. Cada conector possui 70 vias onde estão presentes os sinais de alimentação com nove linhas de terra, quatro de 5v, quatro de 12v, e quatro de -12v; Vinte linhas que compõem os endereços ADRØ até ADR19; Oito linhas de dados DADØ até DAD7; Dez linhas de requisição de barramento BUSREQ1 até BUSREQ1Ø; Três linhas de temporização TEMP1 até TEMP3; Uma linha de relógio do Mestre; Quatro linhas para os sinais de controle IORD, IOWR, MEMRD e MEMWR; E uma linha de RESET.

A.3 ☞ CONTROLADORES ESCRAVOS

Os controladores Escravos são circuitos microprocessados que têm por finalidade realizar o controle digital de cada junta do robô. Eles recebem os sinais dos sensores e enviam os sinais de comando aos acionadores, de acordo com programas de controle que são colocados em suas memórias. Por possuírem uma arquitetura de *hardware* relativamente complexa, também fizemos a divisão de seus circuitos em módulos, e cada um ficou composto por 11 módulos, que independem do tipo de CPU que será utilizada para tal finalidade, chamados por módulo do microprocessador, módulo de geração do relógio, módulo de *reset* manual/automático e externo, módulo amplificador de barramentos da CPU, módulo amplificador do barramento de comunicação com o Mestre, módulo decodificador de endereço, módulo de temporizações e Interrupções, módulo de memória, módulo seletor de E/S, módulo dos circuitos de E/S, e módulo de conversão analógica/digital.

Para o módulo do microprocessador utilizamos como CPU os *chip's* Z-80H que tem por finalidade realizar o controle dos componentes do circuito que compõem o Escravo. Existe uma constate monitoração dos sinais de BUSREQ e BUSACK presentes nos respectivos pinos do microprocessador, isso é feito através de *leds* estrategicamente colocados no painel frontal onde os circuitos estão alojados, para que o operador possa acompanhar o processo de comunicação entre o Mestre e cada Escravo, e avaliar em qualquer momento se os procedimentos estão sendo realizados corretamente. Sempre que o Mestre selecionar um determinado Escravo para transferência de dados, automaticamente é gerado um sinal de BUSREQ provindo do circuito de interface do PC e direcionado para o Escravo em questão. Após receber um sinal desses, o microprocessador termina de realizar a última operação que estava processando, e entra em estado de espera, colocando suas linhas de endereços e dados em alta impedância, e avisando através da linha de BUSACK que já está esperando para que o Mestre assuma o controle do seu subsistema de junta.

O módulo de geração do relógio é responsável por gerar a freqüência de operação para a CPU, para o conversor *A/D* e para os temporizadores programáveis que integram o subsistema de controle.

No módulo de *reset* manual/automático e externo, tem-se a geração de um sinal de reinicialização para a CPU cada vez que o circuito é energizado, ou possibilidade de geração deste sinal de modo manual através de uma chave colocada estrategicamente no painel frontal onde os circuitos são alojados, e ainda cada vez que o Mestre sofrer um pedido de reinicialização.

O módulo amplificador de barramentos da CPU é composto por amplificadores de três estados em todas suas linhas de dados, de endereços e de controle, com a principal finalidade de fazer um desacoplamento entre a CPU e o restante do subsistema integrado a ela, quando o Mestre fazer a requisição do seu barramento por meio de um pedido de BUSREQ.

O módulo amplificador do barramento de comunicação com o Mestre também é composto por amplificadores de três estados ligados aos barramentos de endereços, dados e controle do barramento de comunicação. O controle sobre esta etapa de amplificação é feito pelos sinais de BUSACK da CPU e ADR16 até ADR19, quando o Escravo em questão estiver sendo endereçado pelo Mestre.

O módulo decodificador de endereços é responsável pela geração do sinal apropriado de habilitação para os amplificadores da CPU e amplificadores do barramento de comunicação.

Para fazermos o controle digital de cada servomecanismo que compõe o robô, necessitamos de uma temporização adequada que nos permita contar uma freqüência fixa de amostragem, e ter controle sobre em que períodos cada rotina de controle será executada. Para isso, colocamos o módulo das temporizações e interrupções. Eles são compostos de três canais de temporização acoplados a um controlador de interrupções, com finalidade de coordenar os ciclos de execução de rotinas do programa principal de controle que está posto para ser executado na memória do subsistema. O controlador de interrupções utilizado apresenta oito canais de interrupções controlados por *software*. Alocamos os temporizadores nos canais de 0 até 2 deste controlador, e o sinal de final de conversão de um conversor *A/D* no seu canal 6.

No módulo de memória estão contidos 16Kbytes de memória EPROM e 16Kbytes

de memória *RAM*. A memória *EPROM* existe para armazenarmos as rotinas de inicialização, de interrupções, tabelas de dados, programa monitor (cuja finalidade principal é coordenar a recepção e execução do programa de controle enviado pelo Micro Mestre) e eventuais programas executáveis de controle de junta que tenham sua estrutura aprovada para este controle e que necessitem somente utilizar poucas variáveis armazenáveis, para manipulação em memória *RAM*. Sugerimos que a memória *RAM* seja utilizada deixando uma área de reserva de 3Kbytes para armazenamento do programa de controle a ser transmitido pelo Mestre, quatro áreas de 3Kbytes cada para armazenamento de variáveis durante a execução de uma tarefa (ex. criação de tabelas de dados) e 1Kbyte para *flags*. Mas esta alocação deve ficar a critério do projetista de controladores para as necessidades criadas pelo modo de programar e pelas características de cada algoritmo de controle.

No módulo seletor de E/S, cada dispositivo de E/S possui um endereço fixo ao qual deve ser relacionado quando um deles for selecionado, e este módulo faz esta seleção interpretando e direcionando os sinais de acordo com as requisições da CPU.

O módulo dos circuitos de E/S é composto por portas de E/S com finalidades genéricas (*chip's 8255*, *74F244*, e *74F373*), que podem ser utilizadas para leitura dos dados provindos do circuito que que conta e registra os dados do sensor *encoder* presente em cada junta, e ler dados convenientemente convertido de outros sensores que podem fazer parte do servomecanismo da junta.

O módulo de conversão analógica/digital é baseado no conversor *CA3310* com tempo máximo por conversão de 6μs, e foi projetado para dar capacidade para cada subsistema Escravo poder receber até oito sinais analógicos externos e convertê-los para palavras digitais de 10bits. Estes sinais de entrada analógicos devem ter uma faixa de variação entre -5v e +5v e a seleção temporal de qual das oito entradas será convertida, pode ser realizada por programação, apenas endereçando-a corretamente de acordo com a definição de portas dadas pelo projeto do *hardware*.

Os circuitos eletrônicos de cada um desses módulos que compõe os Escravos, e os respectivos endereços de acesso a cada setor dentro do sistema está apresentado detalhadamente na referência **[22]**.

A seguir detalhamos com maior profundidade o funcionamento dos módulos de temporização e interrupções dos Escravos cujo entendimento é fundamental para o desenvolvimento dos programas de controle do robô.

A.3.1 ☞ Sistema de Temporização por Interrupções dos Escravos do JECAI

O sistema de controle de um robô necessita executar muitos procedimentos de atualização de dados, em tempos sincronizados, ao processar as rotinas de comunicação entre os circuitos que o integram e as rotinas de controle dos servomecanismos de suas juntas. Ele deve poder controlar com precisão os períodos das malhas de controle para que não ocorram instabilidades ocasionadas por esta causa. Estes períodos são determinados em função da estrutura da junta e dos movimentos requisitados, possuindo limites que devem ser respeitados. O programa de controle pode gerar o tempo necessário para cada

período, mas deve-se considerar que isso exige tempo de processamento, e em muitos casos esse tempo é crítico. Como as juntas do manipulador possuem constantes de tempo da ordem de milissegundos, e os processadores que utilizamos processam palavras digitais de 8bits, e possuem frequência de relógio de 8MHz, esse tempo de processamento pode vir a ser muito crítico dependendo da complexidade dos algoritmos de controle a utilizar. Fazendo um cálculo estimativo, considerando que um período de controle fosse de 100ms, cada processador executaria 800.000 ciclos de relógio, se considerarmos uma média de 5 ciclos de relógio por instrução, ele teria capacidade para executar 160.000 instruções neste período. Este número de instruções aparentemente é muito expressivo, mas em testes utilizando um algoritmo PID em ponto fixo para controle de uma única variável executável por um Escravo de uma das juntas do robô, e já bastante otimizado, foram necessárias 210 instruções. Portanto, se considerarmos o controle de mais variáveis, e a utilização de notação em ponto flutuante para aumentar a precisão, esse número máximo de instruções possíveis pode limitar a complexidade do algoritmo de controle a ser utilizado, e se além disso, ainda considerarmos que o mesmo programa de controle deva ser responsável pelas temporizações dos períodos de controle de cada variável, estas limitações ainda serão maiores.

Considerando os fatos citados acima, optamos por liberar cada Escravo da responsabilidade por essas temporizações, inserindo no *hardware* de seu subsistema um circuito com três temporizadores programáveis. Esses circuitos são integrados em dois circuitos integrados, encontráveis no mercado eletrônico especializado, respectivamente com os nomes *8253* (temporizador programável de intervalos), e *8259A* (controlador programável de interrupções).

- ***Temporizador Programável de Intervalos (8253)***

O circuito *8253* é organizado com três contadores independentes de 16bits cada um, cada um com operação indepedente dos demais, programável por *software*, e com contagem baseada em uma freqüência de 2MHz. Com atenção para os tempos necessários para os períodos de controle das juntas do manipulador, projetamos estes contadores para trabalharem entre uma faixa de contagem que vai de 27ms até 131ms, faixa esta adequada para os períodos de controle de todas as juntas. Por isso a base de contagem utilizada é de 500KHz, cada contador é programado para ser um contador binário divisor por N, onde N pode variar de 1 até 65536.

Foram utilizados três contadores, porque é de interesse controlar a posição, a velocidade e a aceleração de cada junta, e assim um contador é alocado para sincronizar os períodos de controle de cada uma destas variáveis.

- **Programação do Circuito Temporizador**

A **figura A.1** mostra o diagrama de blocos do *CI 8253*.

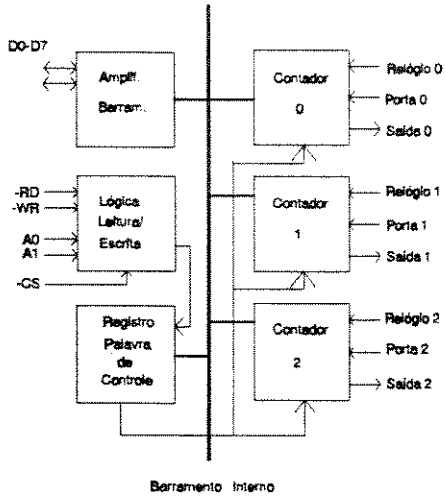


Figura A.1 - Diagrama de Blocos dos Temporizadores Programáveis.

A programação por *software* deve respeitar as situações da **tabela A.1**:

-CS	-RD	-WR	A1	AØ	
Ø	1	Ø	Ø	Ø	Programar Cont Ø
Ø	1	Ø	Ø	1	Programar Cont 1
Ø	1	Ø	1	Ø	Programar Cont 2
Ø	1	Ø	1	1	Modo de Escrita
Ø	Ø	1	Ø	Ø	Leitura Cont Ø
Ø	Ø	1	Ø	1	Leitura Cont 1
Ø	Ø	1	1	Ø	Leitura Cont 2
Ø	Ø	1	1	1	Sem Operação 3-*State*
Ø	X	X	X	X	Desabilita 3-*State*
Ø	1	1	X	X	Sem Operação 3-*State*

X - Não Importa

Tabela A.1 - Programação do *Chip 8253*.

- **Registro da Palavra de Controle**

O registro da palavra de controle é selecionado quando AØ=1 e A1=1. Então o circuito aceita a informação do barramento de dados e armazena no registrador. A informação contida neste registrador controla o modo de operação de cada contador,

seleção de contagem binária ou BCD, e o carregamento de cada registro de contagem.

o **Intefaceamento com os Sistemas Escravos de Controle das Juntas**

Basicamente, as entradas de seleção AØ e A1, são conectadas aos sinais AØ e A1 do barramento de endereços da CPU. A entrada -CS de seleção do circuito está conectada pela porta (ØCh) obtida da saída de um circuito eletrônico multiplexador, assim, para o contador Ø ser selecionado, basta que o programa executado pela CPU acesse a porta (ØCh) para o contador 1 basta que acesse a porta (ØDh), e para o contador 2 basta que acesse a porta (ØEh).

Para o acesso e a programação serem feitos corretamente, as linhas de -RD e -WR, e as linhas do barramento de dados também devem estar conectadas às respectivas linhas da CPU.

o **Operações de Programação e Modos de Contagem**

Cada contador do *CI 8253* é programado individualmente pela escrita da palavra de controle no respectivo registro. Para esta operação as linhas de entrada AØ e A1 devem ser colocadas em nível alto (AØ=1, A1=1).

O formato da palavra de controle deve ser o seguinte:

D7	D6	D5	D4	D3	D2	D1	DØ
SC1	SCØ	RL1	RLØ	M2	M1	MØ	BCD

Onde:

SC1	SCØ	
Ø	Ø	Seleciona Contador Ø
Ø	1	Seleciona Contador 1
1	Ø	Seleciona Contador 2
1	1	Ilegal

Tabela A.2 - Selecionamento dos Contadores.

RL1	RLØ	
Ø	Ø	Operação segundo contagem em circuito *Latch*
1	Ø	Leitura/Carregamento somente do *byte* mais significativo
Ø	1	Leitura/Carregamento somente do *byte* menos significativo
1	1	Leitura/Carregamento primeiro do *byte* menos significativo depois do *byte* mais significativo

Tabela A.3 - Programação Por Partes dos Contadores.

M2	M1	MØ	
Ø	Ø	Ø	Modo Ø
Ø	Ø	1	Modo 1
X	1	Ø	Modo 2
X	1	1	Modo 3
1	Ø	Ø	Modo 4
1	Ø	1	Modo 5

Tabela A.4 - Programação do Tipo de Modo de Contagem.

BCD	
Ø	Como contador binário de 16bits
1	Como contador BCD (4 décadas)

Tabela A.5 - Programação de Contagem Binária ou BCD.

Programamos cada contador como sendo binário de 16bits, e como divisores por N. Então para o contador Ø a palavra de controle é 34h (**figura A.2**).

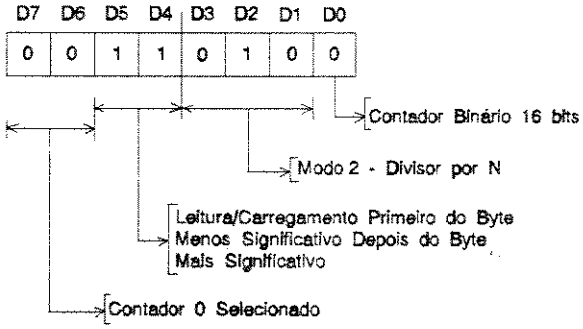


Figura A.2 - Formação de Palavra de Controle para o Contador Ø.

Para os contadores 1 e 2 estas palavras são respectivamente 74h e B4h.

o **Modo 2 de Contagem**

Neste modo de contagem o contador fica programado para ser um contador divisor por N. A saída será baixa por um período de entrada de relógio. O período entre um pulso de saída e o próximo é igual ao número de contagem de entrada que está no registrador de contagem. Se o registrador de contagem for reprogramado entre pulsos de saída, o presente período não é alterado, mas o período subsequente refletirá o novo valor.

As portas de entrada (Porta Ø, Porta 1, e Porta 2), quando baixas, forçam a saída para nível alto. Quando as portas de entrada estão em nível alto, os contadores começam a contagem inicial. Como nos interessa esta segunda situação, conectamos estas entradas diretamente a +5v.

Quando este modo é selecionado, a saída permanecerá alta até que o registrador de contagem seja carregado.

o **Procedimento de Leitura e Escrita**

Como foi explicado, cada contador deve ser programado com o modo, e a quantidade desejada. Como optamos pelo carregamento e leitura através de dois *bytes* (primeiro o menos significativo, depois o mais significativo), através de duas instruções de saída ou de entrada podemos programar ou verificar a atual contagem de cada contador.

A programação dos contadores pode ser feita em qualquer seqüência, não necessita ser numa ordem determinada, isso permite que modifiquemos ou leiamos somente um dos contadores no momento que desejarmos. O Contador Ø não precisa ser o primeiro, nem o contador 3 ser o último. Se quiséssemos programar o contador Ø para contar um período de 10ms, com o *hardware* que projetamos, uma seqüência de instruções que a CPU deveria processar escrita em linguagem *Assembler* do Z-80H seria:

```
LD   A,34h      ;
OUT  (ØFh),A    ; Programa palavra de Controle para Contador 0
LD   A,88h      ;
OUT  (ØCh),A    ; Programa byte menos significativo de contagem para contador 0,
LD   A,13h      ;
OUT  (ØCh),A    ; Programa byte mais significativo de contagem para contador 0
```

Como a frequência de relógio é de 500KHz, então:

$$Tempo\ de\ Contagem = \frac{1}{500x10^3}\ x\ 5000 = 10x10^{-3} = 10ms$$

Então após esta seqüência de instruções a saída do contador Ø irá dar um pulso a cada 10ms que, por exemplo, pode gerar o período da malha de controle da variável de posição.

Se ao invés de utilizarmos instruções de saída, forem utilizadas instruções de entrada, a CPU poderá fazer a leitura atual de contagem de qualquer um dos contadores, por um processo semelhante. Apenas deve-se respeitar a **tabela A.6**, que informa o que deve ocorrer para se obter as leituras desejadas:

A1	AØ	-RD	
Ø	Ø	Ø	Leitura do Contador Ø
Ø	1	Ø	Leitura do Contador 1
1	Ø	Ø	Leitura do Contador 2
1	1	Ø	Ilegal

Tabela A.6 - Programação para Leitura dos Contadores.

Todo procedimento de leitura que inicia deve ser terminado, isto é, a CPU deve executar duas instruções de leitura para a leitura da contagem de cada contador. Na primeira operação de leitura o *CI 8253* fornece o *byte* menos significativo da atual contagem, e na segunda operação ele fornece o *byte* mais significativo. Estes procedimentos não afetam a contagem normal do contador selecionado.

A.3.2 ☞ Sistema de Interrupções Mascaráveis Adotado

Com o *CI 8253* conseguimos gerar períodos bem determinados de tempo para indicar à CPU os instantes de início de um novo ciclo de controle, mas ainda é necessário a inclusão de outro circuito para auxiliar a CPU a interpretar que tipo de ciclo está iniciando, já que há a possibilidade da geração de até 3 ciclos distintos para cada controlador escravo, e cada ciclo deve executar rotinas de controle distintas. Este circuito, irá interpretar, em tempo-real, qual ciclo deve ser inicializado, e informar a CPU para que ela proceda a execução da respectiva rotina de controle. Optamos por utilizar o sistema de interrupção mascarável do microprocessador (-INT) para reconhecimento dos pedidos de inicialização dos ciclos de controle, e um circuito controlador de interrupções programável (*CI 8259A*) para interpretar estes pedidos, selecionar prioridades, e gerar as interrupções mascaráveis no microprocessador.

Cada microprocessador Z-80H dos Escravos possui uma linha de interrupção em seu *hardware*, mascarável por *software* denominada (-INT). Um sinal de interrupção -INT é amostrado pela CPU na transição positiva do último ciclo de relógio de qualquer instrução. A aceitação deste sinal é condicionada por registros internos de habilitação de interrupção (controlados por *software*), e pela linha de requisição de barramento (-BUSREQ). Se estes registros internos estiverem habilitados, a linha de (-BUSREQ) não estiver ativa, e o microprocessador receber um sinal -INT, ele irá aceitar o pedido, fazendo sua linha -IORQ ficar ativa, para indicar que o dispositivo que o interrompeu pode colocar, no barramento de dados, um vetor de 8bits para se identificar com a finalidade de ser atendendido pela sua respectiva rotina de tratamento. Assim, a função do circuito controlador de interruções *8259A* será justamente esta: Gerar o sinal de -INT, e uma vez que o microprocessador o aceite, informar através de uma palavra de 8bits, qual é o

dispositivo que está requisitando um novo ciclo de controle. O microprocessador Z-80H possui três modos de executar uma interrupção mascarada, designadas por: Modo Ø, Modo 1, e Modo 2. Os registros internos responsáveis pela habilitação ou não de uma interrupção -INT são chamados IFF1 e IFF2. O IFF1 determina se pode haver pedido de interrupção, e o IFF2 é uma cópia do IFF1, sendo usado para guardar o *Status* do IFF1. E as instruções em *Assembler* que ativam e desativam estes registros são respectivamente, DI e EI. O Modo Ø de interrupção é automaticamente selecionado pela CPU quando ela recebe um -RESET, ou quando executar uma instrução *Assembler* do tipo IMØ. Nós optamos por utilizar o Modo 1 de interrupção, por melhor se adaptar a nossa filosofia de programação. Porisso passamos a descrever como é o funcionamento neste modo. Para se obter maiores informações sobre os outros modos de operação pode-se recorrer a literatura especializada neste tipo de microprocessador. Para que o Z-80H seja selecionado para processar o Modo 1 de interrupção, toda vez que receber um -RESET ele deve executar a instrução IM1. Neste modo, antes de uma interrupção -INT ser aceita, é verificado se o registro IFF1 está habilitado. Em caso positivo, o registrador contador de programa (*Program Counter - PC*) sempre se modificará para ØØ38h. Portanto, a rotina que irá tratar das interrupções deve ser colocada a partir desta posição de memória. Sempre que ocorrer uma operação deste tipo, o microprocessador executará a rotina que começa na posição de memória ØØ38h, desviando-se da rotina que estava executando, e processará esta rotina de interrupção até encontrar uma instrução RETI (retorno de interrupção mascarável), a qual o fará retornar ao ponto que estava executando antes de ser interrompido.

- ***Controlador Programável de Interrupções (8259A)***

O circuito *8259A* é especificamente projetado para uso em tempo-real, em circuitos digitais microprocessados que operam com sistemas complexos de interrupções. Ele supervisiona até oito níveis de interrupção, com programação de prioridades. A **figura A.3** mostra a configuração em diagrama de blocos *CI*.

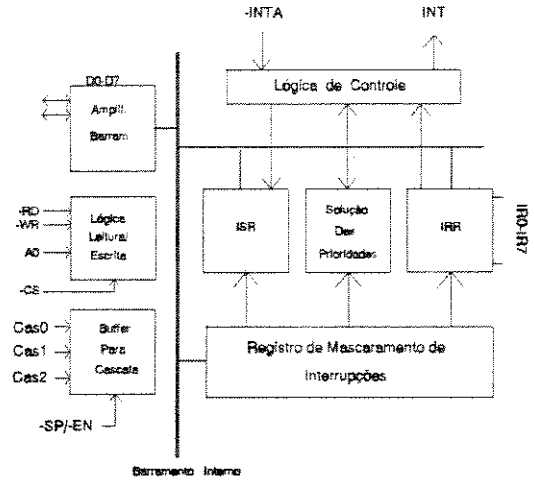


Figura A.3 - Diagrama de Blocos do Controlador Programável de Interrupções.

- **Registrador de Requisição de Interrupção (IRR) e Registrador de Serviço (ISR)**

As interrupções nas linhas IRn de entrada são tratadas por dois registradores em cascata, o registrador de requisição de interrupção (IRR) e o registrador de serviço (ISR). O IRR é usado para armazenar todos os níveis de interrupção requisitados, e o ISR é usado para armazenar todos os níveis de interrupção que estão sendo servidos.

- **Solucionador de Prioridades**

Este bloco determina as prioridades dos *bits* setados no IRR. A mais alta prioridade é selecionada e colocada no *bit* correspondente do ISR durante pulsos de -INTA.

- **Registros de Mascaramento de Interrupção (IMR)**

O IMR armazena os *bits* os quais mascaram as linhas de interrupção. Ele opera no IRR. E o mascaramento da entrada com prioridade mais elevada não afeta as linhas de interrupção de menor prioridade.

- **Lógica de Controle de Leitura e Escrita**

A função deste bloco é aceitar comandos de saída da CPU. Ele contém os registradores das palavras de comando de inicialização (ICW) e os registradores de palavras de comando de operação (OCW) os quais armazenam os vários formatos de controle para a operação do circuito. Este bloco também permite a transferência do *status* do *8259A* para o barramento de dados.

- **Amplificador/Comparador para Cascata**

Este bloco é utilizado para fazer comparações e verificar prioridades quando são utilizados mais de um *CI 8259A* para compor o controlador de interrupções. O qual não é utilizado em nosso projeto, porque um *CI* destes é suficiente por placa controladora Escrava.

o **Seqüência de Interrupção**

Utilizamos o *CI 8259A* de modo como se estivesse operando em um sistema compatível com microprocessadores *8086* ou *8088*, porque ao estudar seu completo funcionamento concluímos que este modo se adapta melhor aos nossos subsistemas de controle de juntas do robô. Neste caso os seguintes eventos ocorrem durante uma seqüência de interrupção.

1) Uma ou mais linhas de requisição de interrupção (IRØ-IR7) são levadas a nível alto (por pulsos gerados pelos contadores programáveis CONTØ, CONT1 e CONT2, por indicação de final de conversão *A/D* pelo respectivo conversor, ou por pedidos externos TEMP1, TEMP2 e TEMP3), setando os correspondentes *bits* do IRR.

2) O *8259A* avalia estas requisições, e envia um pedido de -INT para a CPU através de sua respectiva linha de saída (note que a saída do *CI* é não negada (INT), mas como o Z-80H interpreta a transição do pulso, este fato não é problema, e as linhas podem ser diretamente conectadas).

3) A CPU reconhece o sinal de -INT e responde com um pulso de -INTA.

4) Após recebido um pulso de -INTA da CPU, o *bit* de mais alta prioridade do ISR é setado, e o correspondente *bit* de IRR é resetado, mas o *8259A* não coloca nada no barramento de dados durante este ciclo.

5) A CPU iniciará então um segundo pulso -INTA. Durante este pulso, o *8259A* coloca um apontador de 8 *bits* no barramento de dados que pode ser lido pela CPU, e que indica qual a interrupção que deve ser atendida.

6) Isto completa o ciclo de interrupção. No modo AEOI, que pode ser programado por *software*, o *bit* ISR é resetado no final do segundo pulso de (-INTA), por outro lado, o *bit* ISR permanece setado até que um comando apropriado de EOI terminará o ciclo.

Se nenhuma requisição de interrupção estiver presente durante o **passo 4** da seqüência acima (isto é, se a duração da requisição de interrupção foi muito curta), o *8259A* gerará uma interrupção de nível 7. Isso pode ser utilizado para tornar o sistema imune a eventuais ruídos indesejados.

o **Programação do CI 8259A**

O *8259A* aceita dois tipos de palavras de comando geradas pela CPU:

1) Palavras de comando de inicialização (ICW's): Antes que a operação normal

possa iniciar, a *8259A* deve ser colocada em uma condição inicial por uma seqüência de 2 a 4 *bytes* através de pulsos de -WR.

2) Palavras de comando de operação (OCW's). Estas são as palavras de comando que fazem o *8259A* operar em vários modos de interrupção. Estes modos podem ser:

a) Modo totalmente aninhado
b) Modo com prioridade de rotação
c) Modo com mascaramento especial
d) Modo por votação

As OCW's podem ser escritas na *8259A* em qualquer tempo após a inicialização.

Palavras de Comando de Inicialização (ICW's)

Quando um comando da CPU produz A0=Ø e D4=1, ele é interpretado como palavra 1 de comando de inicialização da *8259A* (ICW1). A palavra ICW1 começa a seqüência de inicialização durante a qual ocorrem automaticamente as seguintes operações:

a) O circuito sensor de bordas é resetado, significando que está se seguindo uma inicialização, uma entrada de requisição de interrupção (IRn) deve fazer uma transição de baixo para alto para gerar uma interrupção.

b) O registrador de mascaramento de interrupção é limpo.

c) A entrada IR7 é colocada em prioridade 7 (mais baixa prioridade).

d) O endereço de modo escravo é setado para 7 (não interessa para o sistema porque só existe um *CI 8259A* por placa escrava, e ele é considerado mestre em cada placa).

e) O modo de mascaramento especial é limpo, e o estado de leitura é setado para IRR.

f) Se IC4=Ø então todas as funções selecionadas pela palavra 4 de comando de inicialização, aplicada mais adiante, são setadas para zero.

Palavras 1 e 2 de Comando de Inicialização (ICW1) (ICW2)

A palavra 1 de comando de inicialização (ICW1) deve ter o seguinte formato:

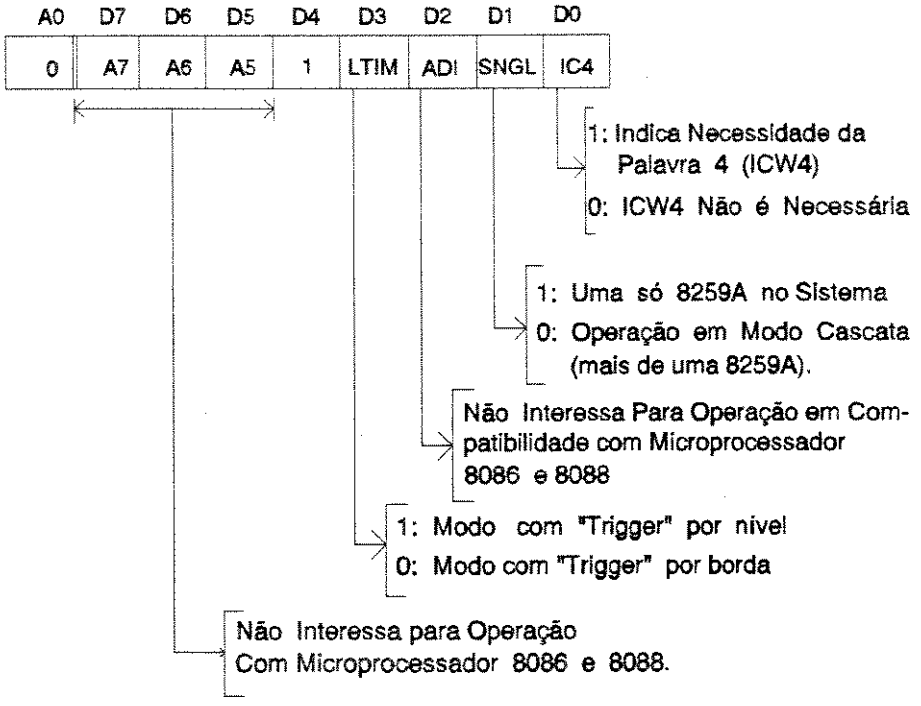


Figura A.4 - Palavra 1 de Comando de Inicialização.

A palavra 2 de comando de inicialização (ICW2) deve ter o seguinte formato:

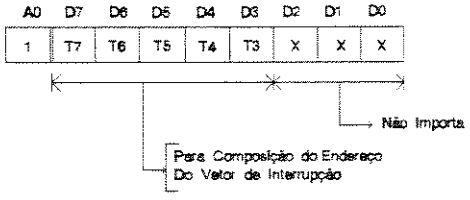


Figura A.5 - Formação da Palavra 2 de Comando de Inicialização.

Como o *8259A* controla 8 níveis de interrupções, ele necessita de apenas três *bits* (os menos significativos) para indicar estes respectivos níveis, mas a palavra digital com a qual ele indicará para a CPU sobre cada nível, possui 8 *bits*. Então através da palavra 2 de comando de inicialização, pode-se programar como estarão os *bits* restantes (D3-D7).

O formato da palavra que o *8529A* envia para a CPU após ter sido requisitado para informar qual a interrupção que deve ser atendida pode ser visto na **tabela A.7**:

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	T7	T6	T5	T4	T3	1	1	1
IR6	T7	T6	T5	T4	T3	1	1	0
IR5	T7	T6	T5	T4	T3	1	0	1
IR4	T7	T6	T5	T4	T3	1	0	0
IR3	T7	T6	T5	T4	T3	0	1	1
IR2	T7	T6	T5	T4	T3	0	1	0
IR1	T7	T6	T5	T4	T3	0	0	1
IR0	T7	T6	T5	T4	T3	0	0	0

Tabela A.7 - Formato da Palavra Digital Para Informar
A CPU Sobre Qual é a Interrupção Requisitada.

Palavras 3 e 4 de Comando de Inicialização (ICW3) (ICW4)

A palavra 3 de comando de inicialização só é necessária quando existe mais de um *CI 8259A* no circuito ligados em cascata, como os controladores só possuem um em cada, esta palavra não é usada.

A palavra 4 de comando de inicialização deve ter o seguinte formato:

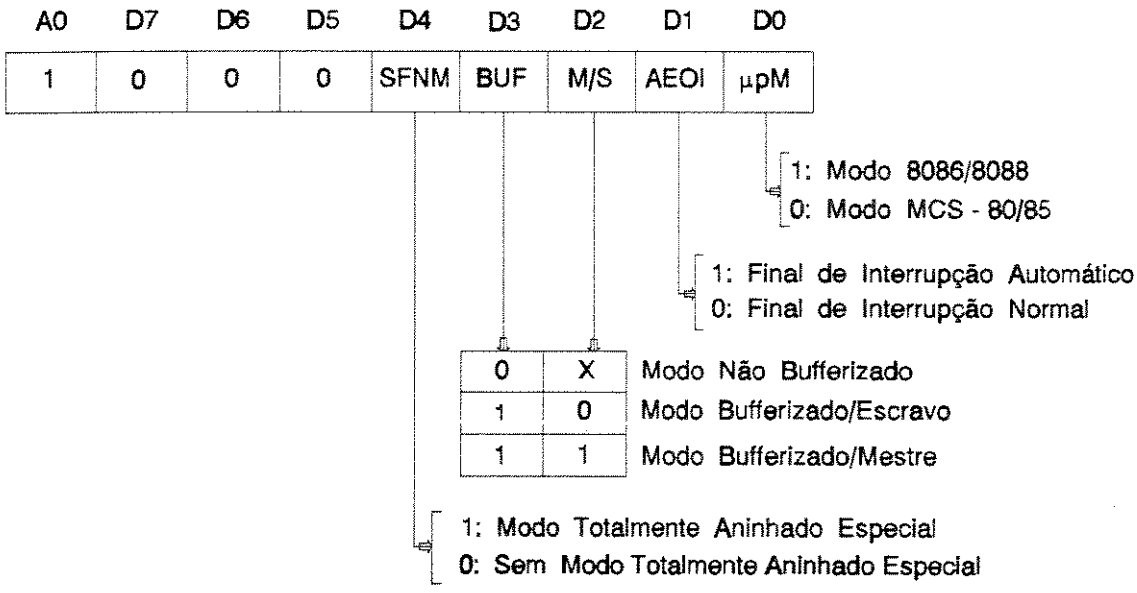


Figura A.6 - Formação da Palavra 4 de Comando de Inicialização.

Palavras de Comando de Operação (OCWS)

Após a programação das palavras de comando de inicialização, o *CI* está pronto para aceitar requisições de interrupção em suas linhas de entrada, entretanto, durante sua operação, a seleção de algoritmos pode comandá-lo para operar em vários modos distintos através das OCWS.

Como estamos interessados simplesmente na opção de podermos mascarar as entradas de interrupção em momentos oportunos, fazemos uso somente da palavra 1 de comando de operação (OCW1) que tem o seguinte formato.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

Figura A.7 - Formato da Palavra 1 de Comando de Operação.

Esta palavra altera o conteúdo do registrador (IMR). MØ até M7 representam os oito *bits* mascaráveis. M=1 indica que o canal será mascarado ou inibido, M=Ø indica que o canal estará habilitado.

As palavras 2 e 3 de comando de operação são úteis quando o *8259A* é programado para alteração de prioridades de interrupções e modo de mascaramento especial (dinâmico), e como não estamos utilizando estes modos, suas explicações serão omitidas. É possível obter informações sobre estes modos em manuais INTEL do *chip*.

Como exemplo de programação da inicialização e operação do *8259A* em nossos controladores de juntas, está colocada abaixo uma seqüência de instruções que deve ser processadas pela CPU (Z-80H):

```
LD   A,13h ;
OUT  (1Øh),A        ; Envia Palavra (ICW1)
LD   A,7Øh ;
OUT  (11h),A        ; Envia Palavra (ICW2)
LD   A,ØFh ;
OUT  (11h),A        ; Envia Palavra (ICW4)
LD   A,ØBEh         ;
OUT (11h),A         ; Envia palavra (OCW1)
```

Atentos para as ligações do *hardware* dos controladores de juntas e às explicações dadas, podemos constatar que ao ser executado o programa anterior pela CPU, o *8259A* será programado como:

- Modo simples (um único *chip* *8259A* no circuito).
- Modo *trigger* por borda.
- Palavra de endereço de interrupção composta por (7x)h, onde x depende da interrupção selecionada pela *8259A*).
- Modo compatível com microprocessadores *8086* e *8088*.
- Final automático de Interrupção.
- Modo *bufferizado*/Mestre.

- Sem Modo totalmente aninhado especial.
- Linhas de entrada MØ e M6 habilitadas.
- Linhas de entrada M1, M2, M3, M4, M5, e M7 desabilitadas.

Portanto, de acordo com as ligações do *hardware*, neste caso a CPU poderá atender a pedidos de interrupções do Contador Ø do *CI 8253*, e de finalização de conversão *A/D* do *CI 3310*. Para o primeiro caso, o *CI 8259A* responderá com um endereço de interrupção 7Øh, para o segundo caso responderá com um endereço de interrupção 76h.

Observe que para essa resposta ocorrer, de acordo com a programação inserida no *8259A*, a CPU deverá gerar dois pulsos de -INTA. Isso pode, por exemplo, ser feito através da execução de duas instruções como a seguir:

```
OUT   (1Ch),A        ; 1º -INTA
IN    A,(1Ch)        ; 2º -INTA
```

De acordo com as ligações do *hardware*, a porta 1Ch acessa a entrada -INTA do *8259A*, e através do segundo acesso, a própria instrução que o gerou já lê o endereço da interrupção requisitada, que é colocado no barramento de dados pelo *8259A*.

ANEXO B

DETALHES DO SISTEMA DE ACIONAMENTO DO JECAII

B.1 ☞ ACIONAMENTO DO ROBÔ

Cada junta que movimenta o **JECAII** possui um motor do tipo CC, com imã-permanente, isto é, campo magnético constante, e que permite seu acionamento através da corrente elétrica que passa por seu circuito de armadura.

Como é conhecido, e pode ser encontrado em bibliografias que tratam deste assunto, o circuito equivalente para um motor CC controlado por corrente de armadura pode ser esquematizado conforme mostrado na **figura B.1**.

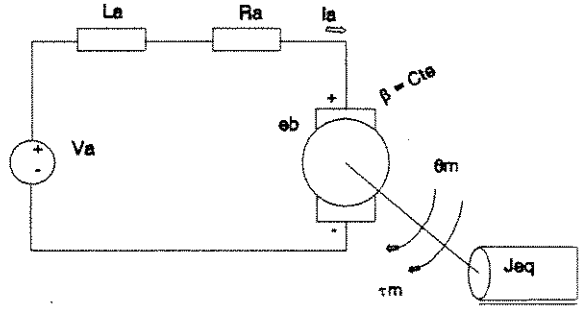


Figura B.1 - Circuito Equivalente de Motor CC Controlado por Armadura.

onde:
- i_a é a corrente de armadura
- e_b é a força contra-eletromotriz
- τ_m é o torque eletromagnético do motor
- θ_m é o deslocamento angular do eixo
- J_{eq} é o momento de inércia do motor + o momento de inércia da carga refletida ao eixo do motor.

Este tipo de motor tem a característica de que o torque entregado em seu eixo varia linearmente com a variação da corrente de armadura, independentemente da velocidade e da posição angular. Isso pode ser expressado pela seguinte equação:

$$\tau_m = K_a \cdot i_a(t) \qquad \textbf{(B.1)}$$

onde K_a é conhecida como a constante de proporcionalidade de torque do motor.

Aplicando-se a lei das tensões de *Kirchhoff* ao circuito elétrico da **figura B.1** obtém-se:

$$V_a(t) = R_a.i_a(t) + L_a.\frac{\delta i_a(t)}{\delta t} + e_b(t) \qquad \textbf{(B.2)}$$

e_b, por construção, é proporcional à velocidade angular do eixo do motor, e pode ser escrita a seguinte equação para ele:

$$e_b(t) = K_b.\frac{\delta \theta_m}{\delta t} \quad ou \quad e_b = K_b.\theta m \qquad \textbf{(B.3)}$$

onde K_b é a constante de proporcionalidade

então:

$$V_a(t) = R_a.i_a(t) + L_a.\frac{\delta i_a(t)}{\delta t} + K_b.\frac{\delta \theta_m}{\delta t} \qquad \textbf{(B.4)}$$

Aplicando-se a transformada de *Laplace* sobre esta última equação tem-se:

$$V_a(s) = R_a I_a(s) + sL_a I_a(s) + sK_b\theta_m(s) \qquad \textbf{(B.5)}$$

e isolando $I_a(s)$, a equação tem a seguinte forma:

$$I_a(s) = \frac{V_a(s) - sK_b\theta_m(s)}{R_a + sL_a} \qquad \textbf{(B.6)}$$

Aplicando-se também a transformada de *Laplace* à **equação B.1** tem-se:

$$\tau(s) = K_a.I_a(s) \qquad \textbf{(B.7)}$$

então,

$$\tau(s) = K_a.\left(\frac{V_a(s) - sK_b\theta_m(s)}{R_a + sL_a}\right) \qquad \textbf{(B.8)}$$

Da parte mecânica do motor, quando conectado a uma carga, também é conhecido que:

$$\tau L(t) = J_c.\ddot{\theta}_c(t) + \beta_c.\dot{\theta}_c(t) \qquad \textbf{(B.9)}$$

e

$$\tau_m(t) = J_m.\ddot{\theta}_m(t) + \beta_m.\dot{\theta}_m(t) \qquad \textbf{(B.10)}$$

onde:

- τL é o torque da carga referido à carga.
- J_c é o momento de inércia da carga referido à carga.
- $\ddot{\theta}_c$ é a aceleração da carga.
- βc é o atrito viscoso da carga referido à carga.
- $\dot{\theta}_c$ é a velocidade da carga.

Para a **equação B.10** estes coeficientes possuem os mesmos atributos, mas referidos ao motor.

Se entre o eixo do motor e a carga existir um sistema de redução de velocidade (amplificação de torque) Caso de cada junta do **JECAII**, com uma relação igual a η (η<1).

então:

$$\theta_c(t) = \eta.\theta_m(t) \qquad \textbf{(B.11)}$$

e

$$\dot{\theta}_c(t) = \eta.\dot{\theta}_m(t) \qquad \textbf{(B.12)}$$

$$\ddot{\theta}_c(t) = \eta.\ddot{\theta}_m(t) \qquad \textbf{(B.13)}$$

Como estamos interessados no torque total visto do lado do eixo do motor, lembrando que existe conservação de trabalho entre a carga e o motor, podemos escrever que:

$$\tau_c^*.\theta_m(t) = \tau_c.\theta_c(t)$$

ou

$$\tau_c^* = \frac{\tau_c.\theta_c(t)}{\theta_m(t)} = \eta.\tau_c(t) \qquad \textbf{(B.14)}$$

usando as **equações B.10, B.12, B.13 e B.14** podemos concluir que:

$$\tau_c^* = \eta^2.\ (J_c.\ddot{\theta}_m(t) + \beta_c.\dot{\theta}_m(t)) \qquad \textbf{(B.15)}$$

Portanto o torque mecânico total visto no eixo do motor fica:

$$\tau(t) = \tau_m(t) + \tau^*L(t) = (J_m+\eta^2J_c).\ddot{\theta}_m(t) + (\beta_m+\eta^2\beta_c).\dot{\theta}_m(t)) \qquad \textbf{(B.16)}$$

ou

$$\tau(t) = J_{eq}.\ddot{\theta}_m(t) + \beta_{eq}.\dot{\theta}_m(t)) \qquad \textbf{(B.17)}$$

Vamos considerar apenas $J = J_{eq}$ e $\beta = \beta_{eq}$

Aplicando a transformada de *Laplace* também à **equação B.17** tem-se:

$$\tau(s) = s^2J\theta_m(s) + s\beta\theta_m(s) \qquad \textbf{(B.18)}$$

Agora com as **equações B.8 e B.18** podemos facilmente encontrar a função de transferência da tensão de armadura para o delocamento angular do motor, apenas igualando e fazendo manipulações algébricas, obtém-se que:

$$\frac{\theta_m(s)}{V_a(s)} = \frac{K_a}{s(s^2JL_a+(L_a\beta+R_aJ)s+R_a\beta+K_aK_b)} \qquad \textbf{(B.19)}$$

Portanto, agora temos uma equação que nos pode indicar como se comporta a posição angular do motor em relação a tensão que está presente em seus bornes. Nesta **equação B.19** estão considerados as constantes de tempo elétrica e mecânica, mas muitas vezes podemos ignorar a constante elétrica, por ser muito menor que a mecânica (dependendo da construção do motor), e daí esta função de transferência fica muito mais simples, ou seja:

$$\frac{\theta_m(s)}{V_a(s)} = \frac{K}{s(T_ms+1)} \qquad \textbf{(B.20)}$$

onde

$$K = \frac{K_a}{R_a.\beta+K_a.K_b} \qquad \textit{ganho do motor}$$

$$T_m = \frac{R_a.J}{R_a.\beta+K_a.K_b} \qquad \textit{constante de tempo do motor}$$

Para conseguir variações adequadas para V_a, nós elegemos o método de

controlar o valor médio de uma onda quadrada de período constante, e amplitude constante, fazendo variações no ciclo útil da onda em cada período (Modulção por Largura de Pulsos). É bem conhecido que este é um método eficiente para ser utilizado neste tipo de aplicação. E através de estudos e experimentos práticos em nosso laboratório, adquirimos uma boa experiência sobre sua utilização em acionamento de motores. Esse raciocínio de obter a função de transferência do motor, acaba por deixar implícito os efeitos da corrente de armadura, que é a variável a ser controlada. Na realidade a etapa do circuito que acionará o motor, deve ser capaz de controlar a potência a ser fornecida ou retirada dele, de acordo com as exigências de torque no seu eixo, e de suas características de construção.

B.1.1 ☞ Acionamento

O acionamento em cada junta isolada, consciste em se produzir tensão e corrente, nos terminais de alimentação do motor, para que ele forneça um determinado torque a cada instante de tempo. Podemos classificar o controle de acionamento segundo o fluxo de potência entre um circuito de alimentação e uma carga (Ex. *PWM* e Motor). Para isso, vejamos o gráfico geral de trabalho Tensão x Corrente sobre uma carga genérica (**figura B.2**):

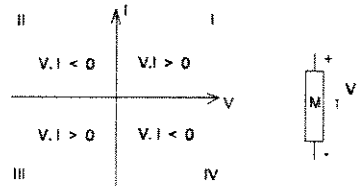


Figura B.2 - Gráfico de Tensão X Corrente Sobre Carga Genérica.

Verifica-se que há quatro regiões distintas de trabalho:

I) $V > 0, i > 0,$	a carga está recebendo potência; o motor estaria sendo acelerado.
II) $V < 0, i > 0,$	a carga está fornecendo potência; o motor estaria sendo frenado.
III) $V < 0, i < 0,$	novamente a carga recebe potência; o motor estaria sendo acelerado no sentido contrário.
IV) $V > 0, i < 0,$	a carga está fornecendo potência; o motor estaria sendo frenado em sentido contrário.

Dependendo do objetivo, pode-se fazer o projeto para atuação em um só dos quadrantes, mais de um, ou mesmo nos quatro. Como temos interesse nestes quatro modos de operação para o controle de cada junta do **JECAII**, fizemos o projeto para que o sistema possa atuar em todos esses modos.

O circuito que propusemos para esta etapa está basicamente composto por dois blocos em cascata, como pode ser visto na **figura B.3**.

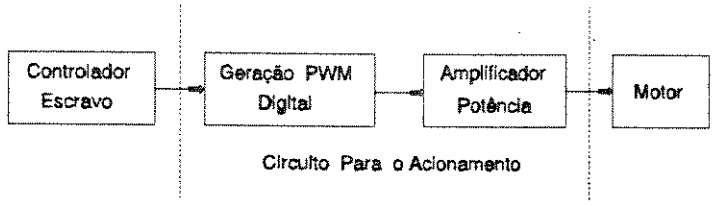


Figura B.3 - Diagrama de Blocos do Circuito de Acionamento do **JECAII**.

A função deste circuito é produzir uma forma de onda, mais aproximadamente quadrada quando possível, com uma freqüência fixa, cuja escolha é função das características do motor + carga imposta pela sua respectiva junta, e dos componentes ativos de potência do amplificador. Fizemos vários testes e simulações, com alguns dos motores que estamos utilizando no projeto, e concluímos que entre uma faixa de freqüências desde ~1250Hz até ~5KHz existe uma resposta adequada destes motores à injeção de sinais *PWM*.

Este é um assunto muito pesquisado (escolhas inadequadas para as freqüências de operação podem gerar instabilidades e fenômenos caóticos **[9][13]**, podem causar problemas de ressonância com a estrutura mecânica do robô, e para freqüências muito elevadas são necessários componentes ativos de alta tecnologia, dependendo das potências envolvidas.

B.1.2 ☞ Amplificador de Potência - Tipo Ponte H

Projetamos os amplificadores de potências que hoje estão sendo usados para acionar cada junta do robô, baseados na freqüência de operação de 5KHz, e para controlar uma potência de ~360VA (24v x 15A), potência esta suficiente para controlar cada um dos motores que está sendo utilizado no robô **JECAII**.

Para estas características de operação encontramos os transistores de potência TIP35 e TIP36, que operam bem e até com uma certa folga **[77]**. Os demais componentes que compõem este amplificador têm a função de controlar adequadamente a ação destes transistores para o controle de cada motor.

A filosofia básica do amplificador *PWM* em topologia Ponte H é fazer com que a partir de uma fonte de alimentação com tensão fixa +V, seja possível injetar na carga, tensões +V e -V. Isto é conseguido através do circuito esquematizado na **figura B.4**.

É fácil verificar que, com o circuito da **figura B.4** quando S=1, as chaves S1 e S3 fecham, e S2 e S4 abrem, então aparece a tensão +V do lado esquerdo da carga, e quando S=Ø, S1 e S3 abrem, e S2 e S4 fecham, fazendo com que apareça a tensão +V do lado direito da carga.

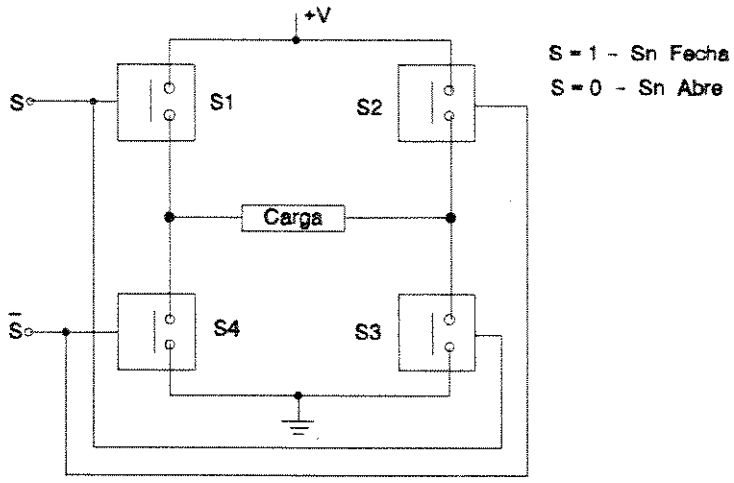


Figura B.4 - Esquema do Amplificador *PWM* em Topologia Ponte H.

Apesar de extremamente simples, este circuito tem limitações práticas para ser implementado, porque principalmente não se conseguem chaves ideais, isto é, as chaves não são instantâneas, e um pequeno retardo em sua abertura ou fechamento pode causar problemas graves no funcionamento do circuito.

Se na transição de nível alto para baixo da entrada S, S2 e S4 fecharem mais rapidamente do que o tempo necessário para S1 e S3 abrirem, então, neste intervalo de tempo, ocorrerá um curto circuito entre +V e terra, que passa por S1 e S4, e outro por S2 e S3, isso poderá ocasionar a inutilização das chaves, ou danos na fonte de alimentação. E este problema se agrava proporcionalmente à freqüência exigida para as chaves comutarem, portanto, existe a necessidade de se acrescentar mais um circuito que garanta a não ocorrência de situações assim.

Para resolver este problema, utilizamos os transistores TIP35 e TIP36 fazendo a polarização para funcionarem como chaves.

O circuito global para a função de amplificação do sinal de *PWM* que projetamos, pode ser visto na **figura B.5**.

Ele possui 4 módulos internos:

- Ponte H de Acionamento (Acionamento Superior e Inferior)
- Circuito de Controle de Acionamento Superior
- Circuito de Controle de Acionamento Inferior
- Acoplamento Óptico.

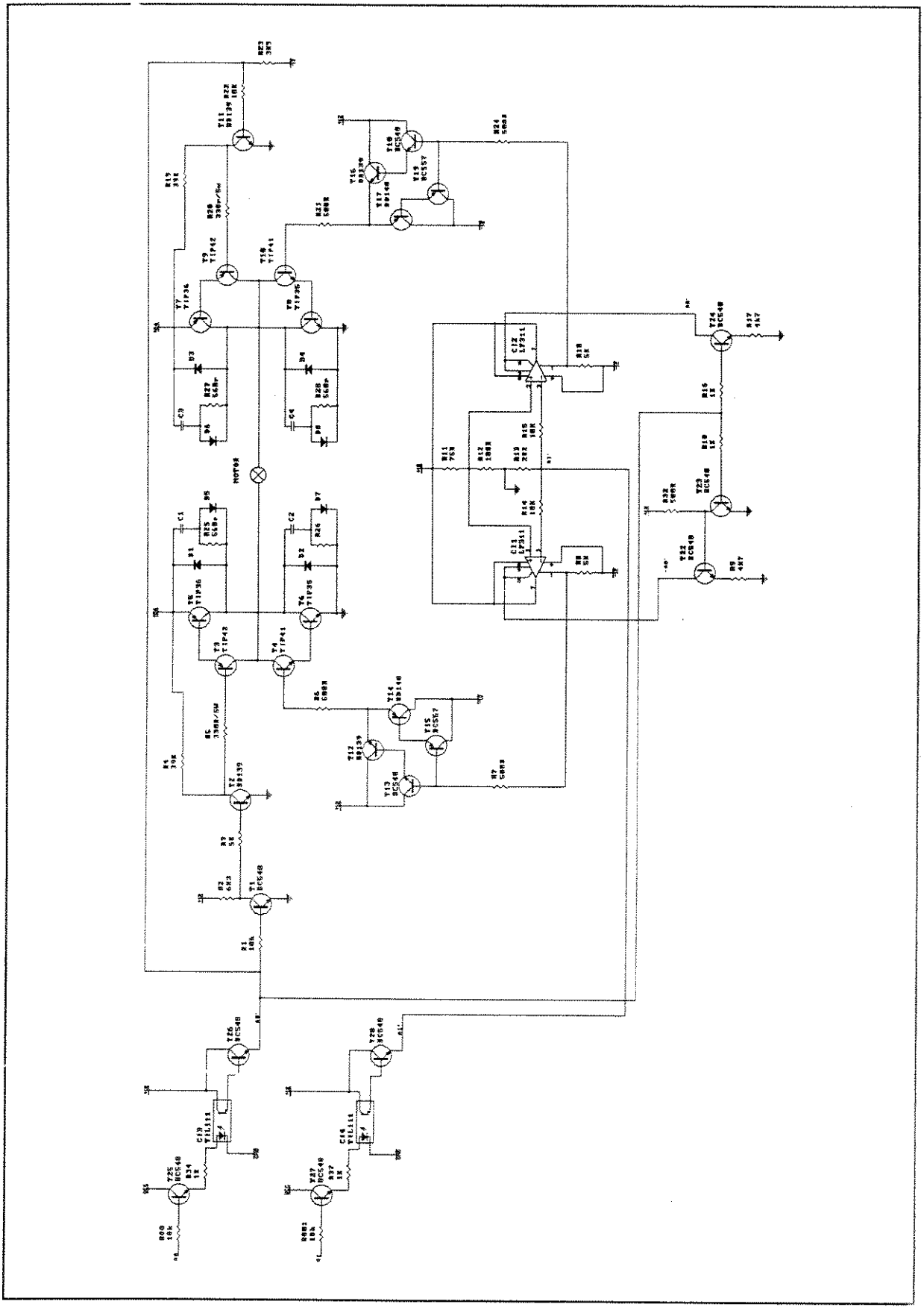


Figura B.5 - Esquema Eletrônico dos Amplificadores *PWM* de Potência do **JECAII**.

♦ ***Ponte H de Acionamento***

Os transistores T3 e T5, T4 e T6, T7 e T9, T8 e T10 formam pares *Darlington*, tendo exclusivamente o objetivo de elevar o ganho β de cada par-chave da ponte H, para não sobrecarregar as outras etapas de menor potência que a controlam. O fator β de amplificação de cada par *Darlington* utilizado, é de aproximadamente 450.

Na parte superior da ponte existem os resistores R5 e R20 que servem para limitar a corrente que passa pelos transistores T3, T5 e T7, T9, respectivamente em ~31A (que poderá ser atingida durante o arranque do motor).

A potência dissipada nos resistores R5 e R20, em caso de máxima exigência do circuito, pode ser calculada como:

$$P_R = I^2.R = (68{,}5mA)^2.330\Omega \cong 1{,}55W$$

Porisso especificamos a potência destes resistores em 5W, como critério prático para não trabalharem super aquecidos.

Os transistores T5 e T7 são do tipo TIP 36C para que além de suportarem estes níveis de correntes, também tenham uma tensão de polarização reversa elevada, para suportar transientes elevados que ocorrem pricipalmente durante o chaveamento. E escolhemos os transistores tipo TIP 42 para compor cada par *Darlington*, porque para os níveis de exigências de correntes nas bases dos TIP's 36, eles suportam bem (~2,1 $A_{máx}$) na freqüência exigida.

Na parte inferior da ponte existem os resistores R6 e R21 que também servem como proteção para os pares *Darlington* T4, T6 e T8, T10, e para permitir que haja corrente suficiente para suas saturações e cortes rápidos.

Os diodos D1, D2, D3, e D4, servem para auxiliar na passagem da corrente no sentido do motor para a fonte de alimentação, quando o sistema está operando no modo de frenagem do motor (Cada vez que o motor está girando em um sentido, a corrente também tem um determinado sentido, e existe energia acumulada nele; quando o sistema necessita reverter o sentido de giro, faz-se necessário retirar rapidamente esta energia acumulada nele, e deve-se promover um caminho para que a corrente possa fluir para a fonte.

E o circuito composto pelos componentes R, C e D, servem para melhorar a resposta transitória de cada transistor durante seu chaveamento (Para cada freqüência de trabalho existe uma combinação melhor). Nós fizemos experimentos práticos variando a freqüência de operação da ponte H entre os limites para os quais fizemos o projeto, e os valores que estão propostos melhoram adequadamente a resposta transitória para estas freqüências, conforme informações obtidas com auxílio de osciloscópio.

♦ *Circuito de Controle de Acionamento Superior*

Chamamos este circuito por tal nome por ele ser responsável pelo acionamento dos transistores superiores da ponte H (PNP). Ele é muito simples, pois o próprio sinal A0' é quem provoca o corte ou a saturação dos transistores T1, T2 e T11, a única diferença entre um lado e outro que vão respectivamente aos transistores da ponte H, é que para o lado esquerdo superior da ponte aparece o sinal A0", que tem o mesmo estado lógico que A0', e para o lado direito superior da ponte aparece o sinal $\bar{A0}$" que é a negação do estado lógico A0'.

Os resistores R1, R2, R3 e R4, apenas servem para garantir as devidas polarizações de T1 e T2 para trabalharem como chaves, e este conjunto faz uma dupla inversão analógica (dupla negação de níveis lógicos), que, em lógica, equivale a não executar operação alguma, portanto, só serve como adaptador de sinais. O transistor T2 é do tipo BD139, para trabalhar com folga mesmo quando haja a máxima exigência de corrente da ponte H, e T1 é um transistor para uso geral tipo BC548 porque controla níveis pequenos de correntes (Icsat ~2mA).

O resistor R23 promove um caminho para o nível de tensão -12v, que é útil para contribuir com a rápida transição de saturação para corte do transistor T11, que também é do tipo BD139 pelas mesmas razões comentadas acima. No entanto, como só existe uma etapa de inversão analógica (negação de nível lógico), ela executa uma operação de negação de A0'.

♦ *Circuito de Controle de Acionamento Inferior*

Utilizamos componentes comparadores de tensões tipo LM311, principalmente porque possuem alimentação simétrica, presença de entradas *enable*, que permitem criar uma porta '&' analógica, e tem excelente rejeição à ruídos de modo comum.

Estes comparadores servem também para conseguir uma melhoria na quadratura da forma de onda vinda da etapa de acoplamento óptico. Através dos resistores R11 e R12 criamos um divisor de tensão, que faz aparecer ~+6,8v nos pinos 2 dos comparadores. Os resistores R13, R14 e R15 são para adequar o sinal A1' vindo do respectivo opto-acoplador, a outra entrada dos comparadores. Portanto, sempre que o nível de tensão de A1' ultrapassar +6,8v, suas respectivas saídas passam para +12v, desde que suas entradas de *enable* estejam habilitadas. Se, no entanto, o nível de tensão de A1' estiver menor que +6,8v, então estas saídas passam para -12v (R8 e R18 são para polarização dos componentes).

As respectivas entradas de *enable* são controlados com o auxílio do mesmo sinal AØ', que serve para controlar a parte superior da ponte, e a lógica de controle é feita pelos transistores T22, T23 e T24 devidamente polarizados para trabalharem como chaves, pelos resistores R9, R10, R16, R17, e R32. Este circuito faz com que quando AØ'= 1, o comparador da direita fique habilitado, e o da esquerda desabilitado, e quando A0'=0, o reverso. Como as saídas destes comparadores conduzem o sinal para cada respectivo transistor inferior da ponte H, então conseguimos sincronizar o funcionamento da ponte adequadamente, garantindo que não ocorra curto circuito entre um mesmo lado da ponte,

isto é, para qualquer combinação entre AØ' e A1', somente um dos transistores de cada lado da ponte conduzirá, e sempre em anti-simetria.

Ainda acrescentamos mais 8 transistores, sendo quatro para cada lado da ponte, para garantir que haja nível suficiente de corrente nas respectivas bases dos pares *Darlington* que formam a parte inferior da ponte H, de tal forma que para a saturação apareça +12v e para o corte -12v, nas respectivas bases, o que garante uma excelente transição entre estes dois estados. Estes transistores são T12, T13, T14, e T15 para o lado esquerdo da ponte, e T16, T17, T18 e T19 para o lado direito. Os resistores R7 e R24 servem apenas para proteção desta parte do circuito.

- ***Acoplamento Óptico***

É sabido que sempre que se unem circuitos onde existe elevados câmbios de potência envolvida nos sinais, a melhor forma de proteção é fazer um isolamento óptico entre tais partes. Então, com este objetivo, na entrada do circuito de amplificação dos sinais de modulação *PWM*, colocamos um circuito opto-acoplador para cada um dos sinais AØ e A1, que provêm da etapa de suas gerações.

Utilizamos componentes TIL111, que são facilmente encontráveis no mercado eletrônico, e que possuem respostas razoáveis para a faixa de freqüências que estamos utilizando.

Esse fato gera também a necessidade de se ter fontes de alimentação independentes entre o primário e o secundário dos opto-acopladores, porisso, necessariamente deve-se construir fontes independentes para a geração de +12v, -12v, e +24v utilizadas nestes amplificadores.

Agora podemos construir uma tabela-verdade (**tabela B.1**) para os sinais de entrada AØ e A1, para facilitar a visualização do que ocorre com cada transistor da ponte H, e a tensão sobre os bornes do motor.

A0	A1	T5	T6	T7	T8	Motor
0	0	C	NC	NC	NC	0
0	1	NC	C	C	NC	-V
1	0	NC	NC	C	NC	0
1	1	C	C	NC	C	+V

onde C : Conduz e NC: Não Conduz

Tabela B.1 - Modos de Operação dos Amplificadores *PWM*.

Ou ainda graficamente:

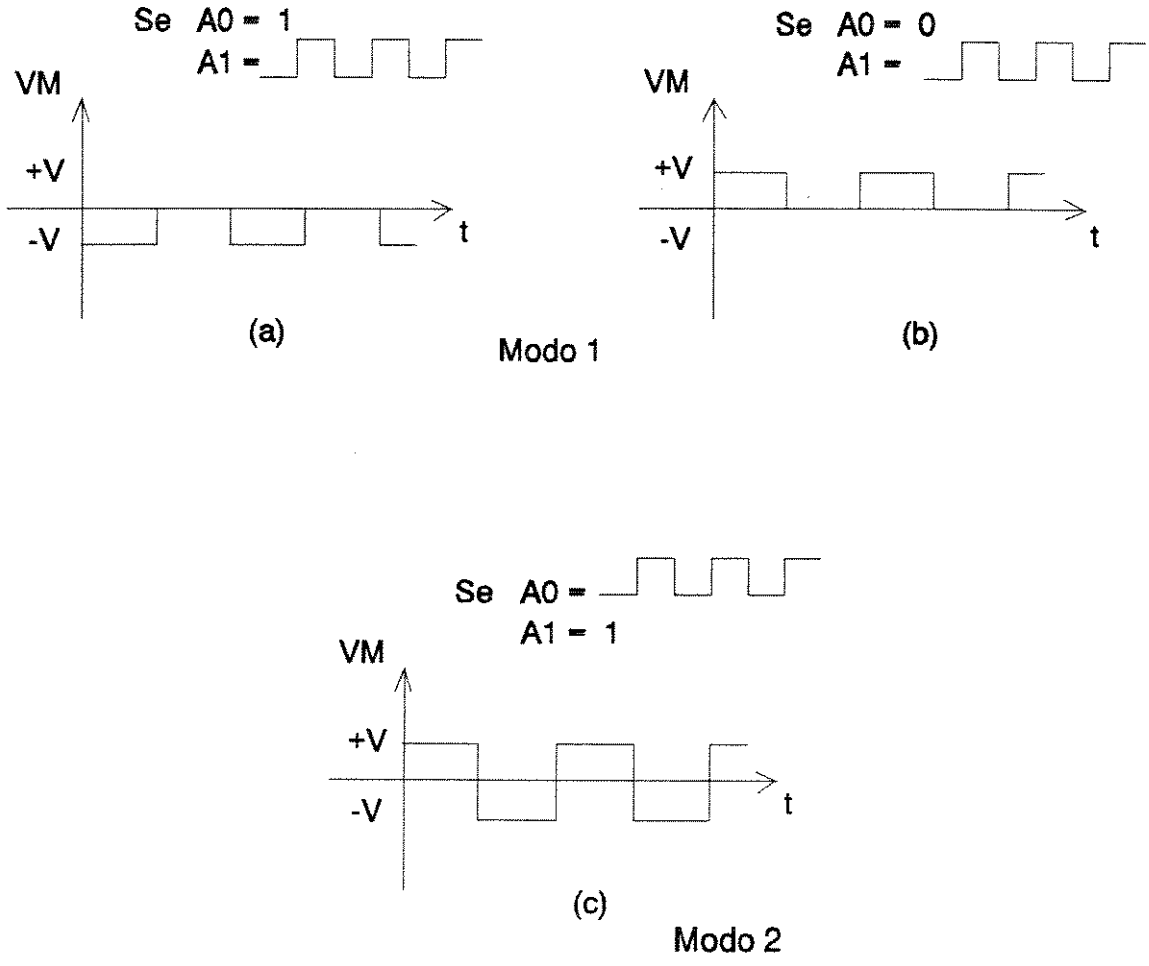


Figura B.6 - Modo de Operação *PWM*.

Se consideramos uma forma de onda quadrada com uma frequência fixa, por exemplo com valor médio zero, cada transistor da ponte H conduz durante um tempo T/2 (dois à dois em anti-simetria). Isso significa que cada transistor deve ser capaz de chavear (saturação para corte, ou corte para saturação), com o dobro dessa freqüência. No entanto, nossa intenção é continuar mantendo a mesma freqüência, mas fazendo variar o valor médio da forma de onda.

Neste caso, se por exemplo, utilizamos o Modo 2 de operação do nosso amplificador, os transistores T5 e T8 deverão conduzir por um tempo muito menor do que os transistores T6 e T7, durante um período. Isso significa que deverão passar dos estados de corte para a saturação, e vice-versa com uma freqüência muito mais elevada do que a freqüência base da modulação, para que o esquema funcione corretamente.

Isso causa sérios problemas para a implementação de amplificadores para modulação por largura de pulsos que devem funcionar em freqüências mais elevadas (ex.

não audíveis), porque faz-se necessária a utilização de componentes extremamente rápidos para garantir seu bom funcionamento, e o problema se agrava mais quando é necessário o controle de potências elevadas.

Vamos fixar a freqüência base da modulação *PWM* em 1KHz, e imaginemos que estamos gerando esta modulação através de um circuito digital que permite gerar 256 ângulos diferentes de comutação entre um período desta onda. Imaginemos que o ângulo de comutação seja equivalente ao 1º destes níveis possíveis. Sabemos que o tempo de um período é de:

$$T = \frac{1}{f} = \frac{1}{1KHz} = 1ms \qquad (21)$$

Que equivale a um ângulo de comutação máximo (360° → 256 níveis)

Logo, o ângulo de comutação seria 360°/256 ≈1,41°, que equivale a um tempo de ≈3,91μs à partir do começo do período. Isto também equivale a que os respectivos transistores da ponte, neste tempo, passem do corte para a saturação, e voltem novamente ao corte, ou ainda, que devem fazê-lo numa freqüência de 256KHz.

Para o caso da freqüência base da modulação ser 5KHz, pelo mesmo raciocínio, será necessário que os transistores sejam capazes de chavear em 1,28MHz.

Podemos pensar em aumentar a precisão dos ângulos de comutação gerados digitalmente para T/4096 (12bits). Neste caso, para uma freqüência base de 1KHz, os transistores devem ser capazes de chavear a uma freqüência de 4,096MHz (para 5KHz → 20,48MHz).

Aparentemente isso inviabiliza o projeto prático, porque componentes com estas características, se existirem, certamente elevariam muito o custo final do robô. Mas mesmo assim, decidimos por investir no projeto do circuito para geração dos sinais *PWM* com possibilidade de obtenção de 4096 ângulos de comutação dentro de um período, porque, em cada período temos uma faixa de atuação com uma precisão muito maior, e isso contribui muito para a melhora do desempenho do acionamento, e para os ângulos de comutação não sustentados pelos transistores de potência, simplesmente descartamos seu uso, utilizando o método de saturação.

Supondo que estamos utilizando um motor que pode ter a velocidade de seu eixo variável entre 0 e 3000rpm, quando variada a tensão aplicada em seus bornes desde 0 até 24v quando este motor funciona à vazio. Se discretizarmos estas 3000rpm em 128 níveis, que seriam os níveis conseguidos para que a tensão média aplicada ao motor varie desde 0 até 24v, utilizando o amplificador que projetamos no Modo 2 de operação, e geração *PWM* digital com 8bits (os restantes 128 níveis provocariam uma variação de 0 até -24v, neste mesmo modo de operação). Então para cada câmbio sucessivo de ângulo, estaríamos variando proporcionalmente ≈23,44rpm. Pode-se constatar que esse é um câmbio muito acentuado, entretanto, ao utilizarmos a geração *PWM* com 12bits, nas mesmas condições, estaremos variando proporcionalmente ≈1,5rpm a cada câmbio sucessivo de ângulo de comutação, o que é bastante mais razoável para fazer um controle melhor sobre o motor. Para os ângulos de comutação muito pequenos (próximos de zero graus), e muito grandes (próximos de 360°), como estamos limitados pela capacidade de chaveamento dos transistores, então simplesmente fazemos as transições desde 0 até o mínimo ângulo suportado, ignorando os ângulos intermediários entre esta faixa, da mesma

forma que na faixa entre o máximo ângulo suportado e 360°. Na prática, isso equivale a fazer com que a variação total dos valores médios de tensões aplicáveis ao motor, tenham como máximo um valor, em módulo, um pouco menor do que a tensão da fonte de alimentação.

B.1.3 ☞ Geração *PWM* Digital

Para que se possa implementar um circuito eletrônico digital que gere formas de onda conforme exigidas pela técnica de modulação por largura de pulsos, primeiramente deve-se considerar a necessidade de se conseguir gerar uma base de tempo precisa para manter a freqüência de modulação fixa. Isso pode ser conseguido com o uso de cristais osciladores.

Para a geração de sinais com 256 níveis de comutação (8bits), faz-se necessário projetar um contador, que conte desde 0 até 255 dentro de um período. Se desejarmos uma freqüência base de 1KHz, e usarmos tecnologia comum, fazendo um contador/divisor por 256, para que gere todos os valores possíveis de contagem de forma cíclica, com saídas conectadas a um comparador de amplitude, que compara estes valores com uma outra palavra digital imposta pelo Escravo de junta, e tenha como saída, pelo menos um sinal que indica se o dado de contagem é menor ou não, do que o fornecido pelo controlador. Então a freqüência de relógio para este contador/divisor por 256 deverá ser de 256KHz (Se a freqüência base exigida for 5KHz, assim a freqüência de relógio do contador deverá ser de 1,2MHz e assim por diante).

Entretanto, para a geração de 4096 ângulos de comutação, utilizando o mesmo tipo de tecnologia, a freqüência de relógio do contador/divisor por 4096, para uma freqüência base de 1KHz, deverá ser de 4,096MHz (para freqüência base de 5KHz, a freqüência de relógio do contador deverá ser de 20,48MHz).

Note que para o raciocínio exposto, se for necessária a geração de freqüências base *PWM* mais elevadas, é bem possível que para este tipo de tecnologia de geração, não haja tecnologia de componentes (ou estes componentes são de custos muito elevados). Existem formas de fazer contadores programáveis, que contam em freqüências muito mais elevadas, utilizando *flip-flop's* tipo D e portas lógicas para gerar sequências de números pseudo-aleatórios, que podem servir para o caso de quere-se fazer geração de sinais *PWM* em freqüências bem mais elevadas, caso haja tecnologia em transistores de potência disponível, ou outros componentes capazes de atuarem nestas freqüências e potências. Esse é um assunto em pesquisa em nosso laboratório.

O circuito que propusemos para fazer a geração *PWM* com 12bits, pode ser visto na **figura B.7**.

Pelo conector *Header* CN1 entra a palavra digital vinda da respectiva placa de controlador Escravo (D0 à DB) referente ao ângulo de comutação desejado para a onda de modulação *PWM*.

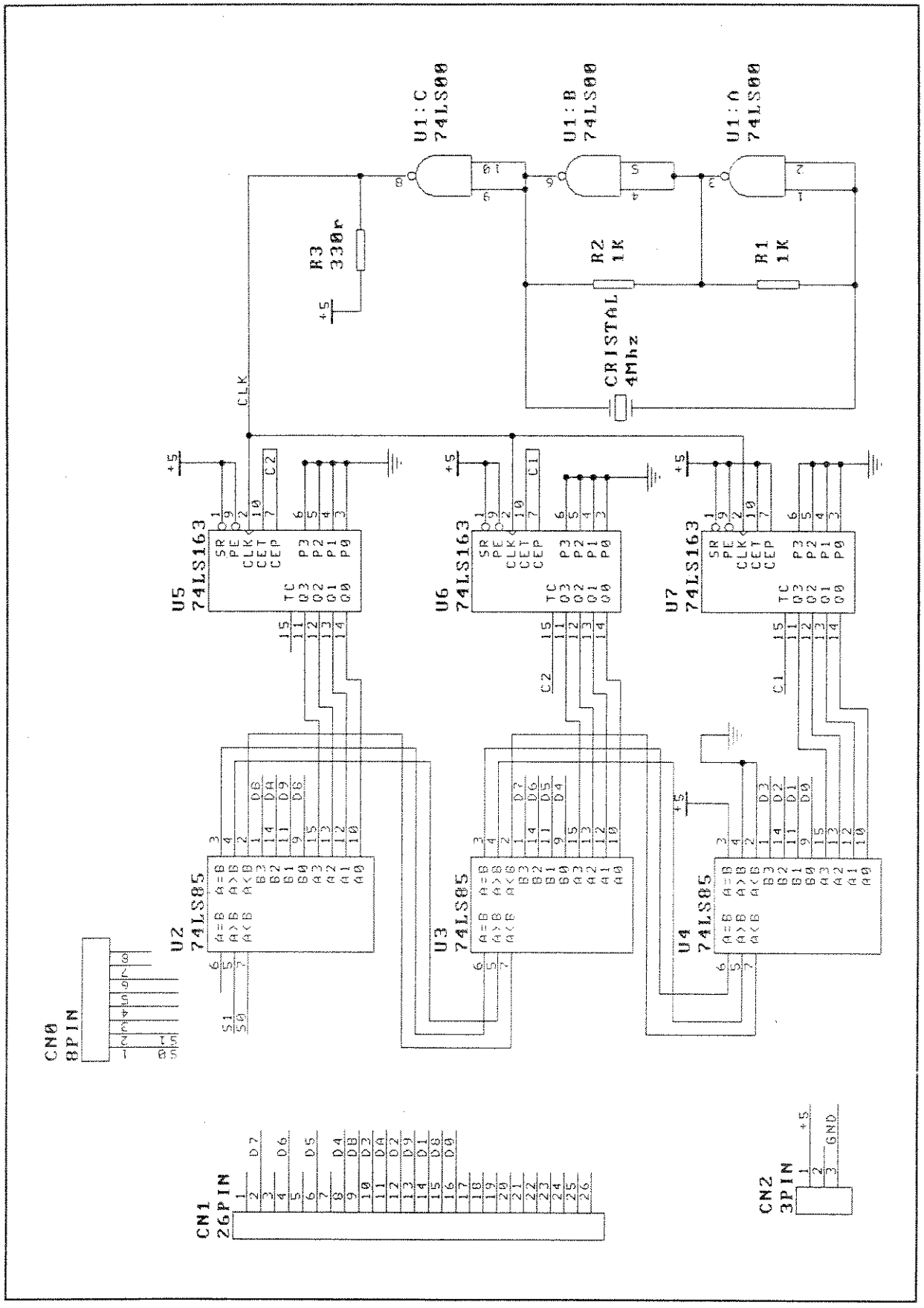


Figura B.7 - Circuito de Geração *PWM* Digital de 12bits do **JECAII**.

Os *Chip's* U2, U3 e U4 formam um comparador de magnitude 12, que compara a palavra digital vinda de CN1, com a última palavra digital de contagem do contador em cascata formado pelos *Chip's* U5, U6 e U7, que têm como relógio, o circuito tradicional de geração, através de inversores e cristal (4MHz para que o contador repita os ciclos de contagem com uma freqüência próxima a 1KHz, e a saída A<B do comparador de maior hierarquia é utilizada diretamente como a entrada AØ ou A1 do amplificador de potência, dependendo do modo que se quer utilizar (modo1 ou modo 2).

Estamos utilizando o modo 2 de operação, e portanto, este sinal de saída do circuito de geração *PWM* é utilizado como entrada A0 para controlar o amplificador de potência, e o sinal A1 simplesmente está conectado a +5v, para a operação nos quatro quadrantes conforme visto anteriormente.

ANEXO C

PROGRAMAÇÃO PARA OPERACIONALIZAR O JECAII

C.1 ☞ ALGORITMOS DE OPERAÇÃO

Ao ligar o sistema, cada controlador de juntas começa a executar o programa monitor que está gravado em sua *EPROM*, a partir da posição ØØØØh. No estágio atual de desenvolvimento do sistema sugerimos um programa em linguagem *Assembler* conforme abaixo:

```
$ALLPUBLIC

      DEFSEG DEFCODE,START=0
      SEG   DEFCODE
STTOP:  EQU    7FFFH

      DEFSEG IOSEG,START=0,CLASS=IOSPACE
      SEG   IOSEG
      ORG    00H

PORT_A: DS          1          ; Endereço da porta digital A de E/S
PORT_B: DS          1          ; Endereço da porta digital B de E/S
PORT_C: DS          1          ; Endereço do porta digital C de E/S
CONTES: DS          1          ; Endereço da Palavra de controle da PiA
CDASUP: DS          1          ;
CDAINF: DS          1          ;
EXPAN1: DS          1          ; Expansão
EXPAN2: DS          1          ; Expansão
CANADC: DS          1          ; Canais de entrada A/D
INICON: DS          1          ; Início de conversão
LADINF: DS          1          ; Byte inferior do A/D
LADSUP: DS          1          ; Byte superior do A/D
CANTE0: DS          1          ; Endereço do canal de temporização 0
CANTE1: DS          1          ; Endereço do canal de temporização 0
CANTE2: DS          1          ; Endereco do canal de temporização 0
CONTEP: DS          1          ; Endereço da Palavra de controle do 8253
ICW1TP: DS          1          ; Endereço da Palavra de controle 1 do 8259
ICW2TP: DS          1          ; Endereço da Palavra de controle 2 do 8259

          ORG       1CH
INTACP: DS          1          ; Simulação do sinal de INTA para o 8259

**********Inicialização de variáveis**********

INIRAM:   EQU       4000H      ;Endereço inicial da RAM
FIMRAM:   EQU       7FFFH      ;Endereço final da RAM
CONPIA:   EQU        80H       ;Seta P_A=E & P_B=S & P_C=S
TEMPO1:   EQU        01H       ;Variável do DELAY
TEMPO2:   EQU       0FFFH      ;Variável do DELAY
PCONT0:   EQU        34H       ;Palavra de controle do canal 0
PCONT1:   EQU        74H       ;Palavra de controle do canal 1
PCONT2:   EQU        0B4H      ;Palavra de controle do canal 2
ITAB1:    EQU       5000H      ;Início da tabela 1
ITAB2:    EQU       5800h      ;Início da tabela 2

**********CONT DE INTERRUPÇÃO **********

ICW1CI:   EQU 13H  ;
ICW2CI:   EQU 70H  ;
ICW4CI:   EQU 0FH  ;
CAN0AD:   EQU 00H  ; Canal Analógico 0
CAN1AD:   EQU 01H  ; Canal Analógico 1
CAN2AD:   EQU 02H  ; Canal Analógico 2
CAN3AD:   EQU 03H  ; Canal Analógico 3
CAN4AD:   EQU 04H  ; Canal Analógico 4
CAN5AD:   EQU 05H  ; Canal Analógico 5
CAN6AD:   EQU 06H  ; Canal Analógico 6
```

```
CAN7AD:   EQU  07H  ; Canal Analógico 7

      DEFSEG ZEROCODE,START=0
      SEG    ZEROCODE

**********Endereço de Reset **********

            ORG       0000H
            IM        1
            DI
            LD        HL,STTOP  ;Inicialização do Stack Pointer
            LD        SP,HL
            LD        HL,ITAB1
            LD        (PONT1),HL
            LD        HL,ITAB2
            LD        (PONT2),HL
            LD        A,00H
            LD        (EW),A
            LD        (HINT),A
            LD        (EW1),A
            JP        INICIO    ;Rotina de Inicialização

**********Endereco Inicial de Interrupção**********

            ORG       0038H
            DI
            PUSH      AF
            PUSH      BC
            PUSH      DE
            PUSH      HL
            PUSH      IX
            PUSH      IY
            LD        A,00H
            OUT       (INTACP),A
            IN        A,(INTACP)
            EI
VER0:       CP        70H
            JP        NZ,VER1
            CALL      4000H
            JP        NFIM
VER1:       CP        71H
            JP        NZ,VER2
            JP        NFIM
VER2:       CP        72H
            JP        NZ,VER3
            JP        NFIM
VER3:       CP        73H
            JP        NZ,VER4
            JP        NFIM
VER4:       CP        74H
            JP        NZ,VER5
            JP        NFIM
VER5:       CP        75H
            JP        NZ,VER6
            JP        NFIM
VER6:       CP        76H                ;Leitura do A/D
            JP        NZ,VER7
            IN        A,(LADINF)
            LD        (ADCINF),A
            IN        A,(LADSUP)
            LD        (ADCSUP),A
            JP        NFIM
VER7:       CP        77H
            JP        NZ,NFIM
            JP        NFIM
NFIM:       POP       IY
            POP   IX
```

```
        POP     HL
        POP     DE
        POP     BC
        POP     AF
        RETI

;*****************************************************
;*          ROTINA DE INICIALIZAÇÃO                  *
;*****************************************************
;****************** Inicialização Da PIA ***************
        ORG     0500H
INICIO: LD      A,CONPIA
        OUT     (CONTES),A
        LD      A,80H
        OUT     (PORT_A),A

;********** Inicialização do Cont. de Interrupção **********

        LD      A,ICW1CI
        OUT     (ICW1TP),A
        LD      A,ICW2CI
        OUT     (ICW2TP),A
        LD      A,ICW4CI
        OUT     (ICW2TP),A
        LD      A,0BEH
        OUT     (ICW2TP),A

;************ Inicialização dos Temporizadores ************

CANAL0: LD      A,PCONT0
        OUT     (CONTEP),A
        LD      A,080H
        OUT     (CANTE0),A
        LD      A,20H
        OUT     (CANTE0),A
CANAL1: LD      A,PCONT1
        OUT     (CONTEP),A
        LD      A,00H
        OUT     (CANTE1),A
        LD      A,00H
        OUT     (CANTE1),A
CANAL2: LD      A,PCONT2
        OUT     (CONTEP),A
        LD      A,00H
        OUT     (CANTE2),A
        LD      A,00H
        OUT     (CANTE2),A

;******************** PROGRAMA PRINCIPAL ****************

TEST:   LD      A,(HINT)
        CP      0FFH
        JP      Z,TINT
        JP      TEST
TINT:   EI
CONT:   LD      A,(HINT)
        CP      00H
        JP      NZ,CONT
        DI
        LD      A,80H
        OUT     (PORT_A),A
        JP      TEST

;*****************************************************
;*          ROTINA  DELAY                            *
;*****************************************************
```

```
DELAY   PUSH    BC
        PUSH    DE
        LD      B,TEMPO1        ;Podem ser passados por
        LD      DE,TEMPO2       ;Software
DELAY1: DEC     DE
        LD      A,E
        OR      D
        JR      NZ,DELAY1
        DEC     B
        LD      A,B
        OR      B
        JR      NZ,DELAY1
        POP     DE
        POP     BC
        RET

*********************DADOS*****************************

        ORG     74FFH
ADCINF: DS      1       ; Byte Inferior Do A/D
ADCSUP: DS      1       ; Byte Superior Do A/D
MFLAGS: DS      1       ; Byte De Teste Interno De Memória RAM
        ORG     7505H
PROGRI: DS      1       ; Variavel de Saida Para PIA
HINT:   DS      1
        ORG     7617H
EW:     DS      1       ;7619H- Habilitação De Escrita De Posição Na Memória
PONT1:  DS      2       ;761AH- Ponterio da Tabela 1
PONT2:  DS      2       ;761CH- Ponterio da Tabela 2
EW1:    DS      1       ;761EH
*********************PILHA*****************************
        ORG     7D00H
STACK:  DS      255

    END
```

As instruções iniciais têm três funções importantes:

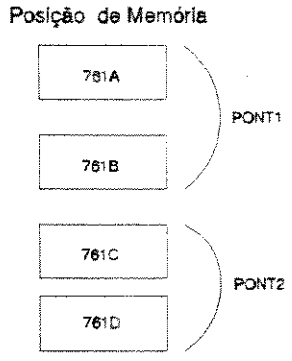
- Programar o Z-80H para interpretar interrupções no modo 1 (IM 1)
- Desabilitar as Interrupções (DI)
- Inicializar o *Stack Pointer* (Aviso para o Z-80H da última posição de memória *RAM* da sua placa - no caso 7FFFh).

```
        LD  HL,STTOP
        LD  SP,HL           * STTOP foi definido no inicio deste programa assim:
STTOP: EQU  7FFFh.
```

As próximas 4 instruções são para preparar duas regiões de memória que servem para armazenamento de dados durante a execução do programa de controle, para posterior análise, por exemplo: desenhar gráficos do comportamento de variáveis.

Designamos para isso, dois registradores de 16bits que contém os endereços destas respectivas regiões de memória (para formar tabelas de dados). Estes registradores chamam-se ITAB1 e ITAB2 (Note que isso em realidade não é essencial para o sistema funcionar, é só para que possamos desenhar as curvas do progresso das variáveis no final de uma execução. Vejam que também estes registradores foram definidos no início do programa, e apontam para as posições de memória 5ØØØh e 6ØØØh respectivamente. Também definimos dois registradores de dois *bytes* cada um que servem para armazenar um valor que aponta para uma posição de cada uma destas tabelas com finalidade de podermos facilmente fazer pesquisa de dados nestas tabelas (Endereçam posição atual

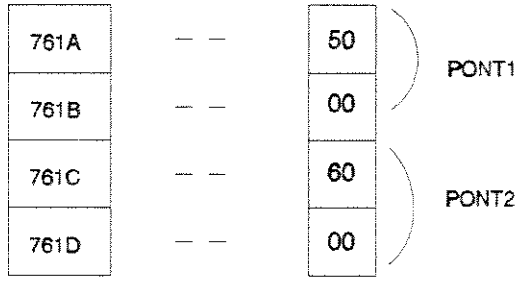
para ler ou escrever nas posições destas tabelas), estes registradores estão definidos no final do programa com os nomes PONT1, e PONT2.



Então as instruções:

```
LD  HL,ITAB1
LD  (PONT1),HL
LD  HL,ITAB2
LD  (PONT2),HL
```

Inicializam estes apontadores com os endereços respectivos dos inícios de cada tabela, isto é:



A seguir definimos dois registradores muito importantes para o controle de execução, são eles: EW e HINT.

O registrador EW, alocado na posição de memória 7619h, serve para indicar se o Z-80H estará habilitado, ou não, para escrever dados nas regiões de memória destinadas para o armazenamento de dados gerados durante a execução de um programa de controle. Se nele estiver armazenado o valor ØØh então o Z-80H não estará habilitado para fazer isso, de modo contrário estará (A alocação de memória para este registro está no final do programa).

O registrador HINT alocado na posição de memória 75Ø6h é o registro que utilizamos como habilitador/desabilitador de execução da subrotina de controle. Se nele estiver armazenado o valor ØØh, o controlador de junta não estará habilitado a executar o programa de controle, em caso contrário estará.

Este registro é que dá a função de Escravo aos controladores de juntas, O Mestre, através dele, pode fazer um controlador executar ou parar a execução do programa

de controle da respectiva junta. (A alocação de memória para este registro está no final do programa).

Então as três próximas instruções:

```
LD  A,ØØh
LD  (EW),A
LD  (HINT),A
```

Inicializam estes registros fazendo com que, neste momento, o controlador não esteja apto a escrever em posições de memória das tabelas, nem a executar o programa de controle.

A seguir a instrução: JP INICIO é utilizada para pular para a região de memória onde o programa continua, deixando livre a área destinada para a subrotina de tratamento de interrupção. Decidimos deixar reservado até a posição de memória Ø4FFh para esta subrotina de interrupção, para que possa ter espaço para um programa que poderá ser bastante complexo, porisso o programa continua a partir da posição de memória Ø5ØØh (a instrução ORG Ø5ØØh é quem controla este salto de uma posição para outra na programação, no caso para a posição Ø5ØØh).

Logo em seguida é necessário fazer a programação dos *chip's* programáveis que compõe o sistema, isto é: *PIA 8255*, controladora de Interrupções *8259A*, e Temporizadores *8253*, de acordo com o interesse do programador.

Verificando nos modos de programação da PIA, e no circuito esquemático dos controladores Escravos, constata-se que toda vez que for executada uma instrução tipo: OUT (Ø3h),A; o valor Ø3h indica a porta que prepara a PIA para ser programada, e o registrador "A" do Z-80H deverá conter o valor que definirá qual é o tipo de programação que será feita. Por exemplo: Se "A" for 8Øh então a PORTA A da PIA será uma Entrada, a PORTA B será Saída, e a PORTA C será Saída. Desse modo, variando o valor de "A" pode-se programar a PIA para qualquer combinação de Entradas e Saídas das respectivas portas que se desejar. Note que nós definimos o *label* CONPIA para conter a palavra de programação (está no início do programa, no setor *Inicialização de Variáveis* - Aí é onde o modo de programação da PIA é definido), e o *label* CONTES definido no início do programa, contém o valor Ø3h, portanto as instruções:

```
LD  A,CONPIA
OUT  (CONTES),A
```

fazem esta programação.

As duas instruções seguintes são para garantir que o motor ficará "parado" neste momento, pois de acordo com o *hardware*, a PORTA A da PIA, envia para o circuito de geração *PWM* a palavra de controle (ou o ângulo de comutação), e a palavra 8Øh equivale a um ângulo de 180° (Neste caso, a largura positiva do ciclo *PWM* é igual a largura negativa, portanto o motor fica parado). Estas instruções são:

```
LD  A,8Øh
OUT (PORT_A),A
```

A seguir vem a programação da controladora de interrupções. Esta é uma programação mais sofisticada, porque o *chip 8259A* permite fazer controles de interrupções

desde simples até muito complexos conforme foi visto. São necessários 5 procedimentos para inicializá-lo com a programação.

Nós procuramos definir *label's* para auxiliar, de acordo com os termos utilizados pela INTEL, portanto os termos: ICW1CI, ICW1TP, ICW2CI, ICW2TP, e ICW4CI, deverão conter as palavras digitais que farão a programação de acordo com os propósitos de controle. Estes *label's* são definidos no início do programa, e no setor *Cont. de Interrupção.*

Note que neste programa monitor nós os definimos assim:

```
ICW1CI = 13h
ICW1TP = 1Øh
ICW2CI = 7Øh
ICW2TP = 11h
ICW4CI = ØFh
```

E as próximas operações realizam sua programação:

```
LD  A,ICW1CI
OUT (ICW1TP),A
LD  A,ICW2CI
OUT (ICW2TP),A
LD  A,ICW4CI
OUT (ICW2TP),A
LD  A,ØBEh
OUT (ICW2TP),A
```

Em seguida vêm as instruções de programação dos contadores (*chip 8253*). Aqui também definimos vários *label's* para auxiliar, eles são: PCONTØ, PCONT1, e PCONT2, que estão definidos no início do programa no setor *Inicialização de Variáveis*. No programa definimos estes *label's* com os seguintes valores:

```
PCONTØ = 34h
PCONT1 = 74h
PCONT2 = ØB4h
```

Aqui procedemos de acordo com as explicações dadas no ítem *Temporizador Programável de Intervalos (8253)* para definir os respectivos canais. Da mesma forma os *label's*: CONTEP, CANTEØ, CANTE1 e CANTE2 também devem ser definidos no início do programa de acordo com as portas impostas pelo *hardware* (como o *hardware* já está definido, esses valores não deverão sofrer alterações).

Eles são:

```
CONTEP = ØFh
CANTEØ = ØCh
CANTE1 = ØDh
CANTE2 = ØEh
```

Depois das instruções de programação dos temporizadores (término da inicialização de cada controlador Escravo) vem a rotina principal, que na realidade é extremamente simples, e serve para bloquear e desbloquear a execução do programa de controle propriamente dito **[7]**.

O fluxograma desta rotina principal pode ser visto na **figura C.1**.

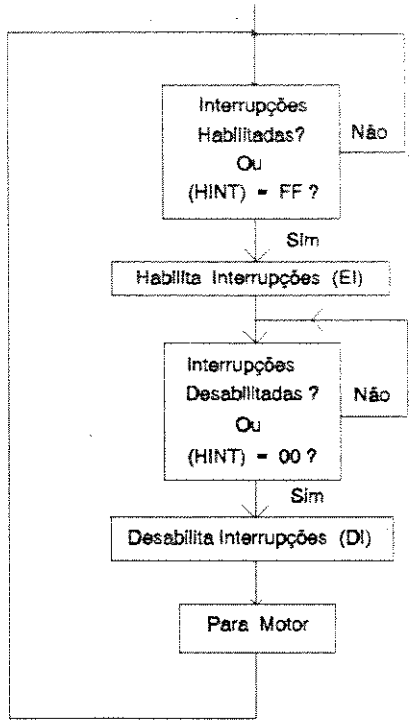


Figura C.1 - Fluxograma da Rotina Principal dos Escravos.

Enquanto o Mestre não injetar o valor FFh na posição de memória 75Ø6h (HINT) dos Escravos, estes controladores não atendem aos pedidos de interrupção, mesmo que sejam feitos estes pedidos (isto é, o programa de controle não será executado, porque ele é executado via interrupção). Uma vez que o Mestre injetar FFh na posição 75Ø6h, as interrupções são habilitadas por uma instrução EI, e então o programa de controle ficará sendo executado até que o Mestre envie novamente o valor ØØh para esta posição de memória (HINT), daí ocorrerá uma instrução DI que novamente desabilitará os atendimentos às interrupções, portanto, o programa de controle parará de ser executado, e para garantir, é enviada uma palavra digital para a porta de saída que controla o motor para fazê-lo parar. Este ciclo é sempre repetido até que seja dado um *-Reset*.

A subrotina de tratamento de interrupções também faz parte do programa monitor, e começa na posição de memória ØØ38h (Modo 1 - Z-80H). De acordo com o *hardware*, toda vez que o *chip 8259A* receber algum pulso em algum de seus 8 pinos de entrada, através de algum dos contadores Ø, 1 e 2, ou da saída de final de conversão do conversor *A/D CA 3310*, e da programação que foi dada a ele na inicialização, este *chip* injetará um pulso no pino -INT do Z-80H, que, se estiver habilitado, atenderá a um pedido de interrupção, parando "momentaneamente" aquilo que estava fazendo, para executar a subrotina que está à partir da posição de memória ØØ38h, e só retornará desta subrotina quando encontrar uma instrução: RETI (retorno de interrupção INT), ou se receber outra interrupção e estiver com o sistema de interrupção habilitado (interrupção dentro de interrupção, terá que retornar "duas vezes", etc..).

Então a estratégia é fazer uma subrotina de interrupção que interprete o que executar em função do pedido feito pelo *chip 8259A*, direcionar a execução para o determinado atendimento de função, esperar o retorno desta função, e devolver o processador para a execução normal (rotina principal) que estava executando antes de entrar na subrotina de interrupção.

Assim que a execução do programa monitor chegar na parte da rotina principal, e o Mestre habilitar as interrupções (HINT= FF), começa este processo. Como os contadores geram pulsos cíclicos de acordo com o modo de contagem e o tempo programados, a cada pulso desses, a subrotina de interrupção é executada. Quando é feito o desvio para a posição de memória ØØ38h primeiramente ocorre a execução de uma instrução DI (desabilita-se "momentaneamente" o atendimento a uma nova interrupção). Em subrotinas mais complexas, pode ser que tenha-se que deixar habilitado, por questões de prioridades, etc..

Logo em seguida fazemos um salvamento de todos os conteúdos atuais dos registradores, para garantir que quando a subrotina de interrupção terminar, o Z-80H continue a processar aquilo que estava processando com as mesmas condições, e as instruções em *Assembler* que realizam isto são:

```
PUSH AF
PUSH BC
PUSH DE
PUSH HL
PUSH IX
PUSH IY
```

Note que poderíamos utilizar uma instrução tipo EXX, fazendo troca entre registradores principais e alternativos, que teria o mesmo efeito, se ao final desta subrotina utilizássemos outra EXX, e durante a execução da subrotina não utilizássemos os registradores alternativos.

Existem muitos procedimentos para se atingir este objetivo, nós sugerimos fazê-lo assim, apenas como uma idéia, mas não há necessidade de se utilizar sempre as mesmas instruções. Inclusive existem muitas outras formas de programar que podem ser eficazes inclusive gastando menos memória, com execução mais rápida, etc.., que os programadores *Assembler* podem fazer, o importante é apenas respeitar a filosofia do projeto, e estar-se atento para os endereços definidos pelas ligações do *hardware*.

Logo em seguida, é necessário interpretar, o tipo de ação a realizar, isto é feito com um pedido para a controladora de interrupções, de qual foi o "periférico" que fez a chamada. Isso é feito com as próximas três instruções:

```
LD  A,ØØh
OUT (INTACP),A      ; Esta instruçao é um artifício para o
IN  A,(INTACP)      ; gerar o 1º pulso necessário para que a 8259A indique qual o byte relativo , ao periférico que fez a chamada
```

Depois desta seqüência, em "A" estará o número correspondente ao "periférico" que fez a chamada de acordo com a programação que foi feita inicialmente, e com as conexões do *hardware*. Assim, são possíveis retornar valores entre 7Øh e 77h.

Se retornar 7Øh foi pedido do contador Ø, se retornar 71h foi pedido do contador 1, se retornar 72h foi pedido do contador 2, se retornar 76h foi pedido do conversor *A/D* (final de conversão *A/D*). Os demais pinos de entrada do *chip 8259A* estão disponíveis no barramento, e atualmente não estão sendo utilizados, eles podem servir para a expansão do controlador Escravo.

No caso particular deste programa monitor sugerido, estamos utilizando somente o contador Ø e o conversor *A/D*, para gerar interrupções, para fazer somente o controle de posição, e através de um potenciômetro que necessita conversão *A/D* para se

interpretar seus sinais. Para processar algoritmos mais sofisticados, basta habilitar-se os outros pedidos de interrupções necessários com pequena mudança na programação do *chip 8259A*.

Então a cada pulso de interrupção gerado, um novo processo de interrupção começa, a subrotina de interrupção interpreta qual é o pedido, e a seguinte seqüência é executada:

```
VERØ:   CP   7Øh
        JP   NZ,VER1
        CALL 4ØØØh
        JP   NFIM

VER1:   CP   71h
        .
        .
        .

NFIM:   POP  IY
        POP  IX
        POP  HL
        POP  DE
        POP  BC
        POP  AF
        RETI
```

Como o primeiro valor que retorna do *chip 8259A* é 7Øh, a primeira comparação será verdadeira, portanto, o Z-80H é forçado a executar uma subrotina que deve começar na posição de memória 4ØØØh (Esta posição foi arbitrariamente escolhida, poderia ser 4ØØ1h, 4Ø2Øh, etc...). Escolhemos esta porque é a primeira posição de memória *RAM* do Escravo, e simplesmente para otimizar seu uso. Então, justamente a partir desta posição 4ØØØh é que deve ser colocado o programa de controle (P, PI, PID,.., etc..), que controlará o servomecanismo pelo qual o determinado controlador é responsável.

O pacote de *software* de alto nível executado pelo Mestre é quem fica encarregado de enviar estes programas de controle, desenvolvidos em *Assembler* para os microprocessadores que baseiam os circuitos dos Escravos, à suas memórias *RAM*, isso após eles estarem devidamente montados e compilados para a execução. Porisso deve-se ter atenção para a posição inicial de memória destes controladores, definida para isto (no caso 4ØØØh), visando os seus corretos desenvolvimentos.

É muito importante que estes programas *Assembler* de controle que serão injetados nos Escravos pelo Mestre, sempre terminem com uma instrução RET, porque em cada interrupção há uma chamada para subrotina, de modo contrário, certamente dará *Overflow* na memória do respectivo controlador por controle incorreto do seu *stack pointer*.

Como deixamos duas áreas reservadas para tabelas de armazenamento de dados que começam a partir da posição de memória 5ØØØh, logicamente o programa de controle que irá ser injetado na memória (baseado nesta filosofia), deverá caber entre as posições 4ØØØh e 4FFFh. Se ele for tão complexo que necessite mais memória que isso, pode-se sacrificar uma, ou até as duas áreas reservadas para estas tabelas, simplesmente retirando-as do programa.

Deve-se lembrar que é importante não utilizar os FFh *bytes* finais da memória *RAM*, para deixá-los reservados como pilha para o microprocessador poder

guardar dados intermediários de processamento.

Como normalmente um programa de controle tem uma estrutura fixa, e apenas variáveis que têm seus valores alterados, sempre que se chegar a uma boa estrutura, pode-se gravá-la em *EPROM* (depois do espaço para o programa monitor), fazendo com que este programa manipule valores de posições de memória estratégicas definidas na *RAM*, que poderia ser utilizada para finalidades de sofisticação de análise de comportamento, se nela forem armazenados de forma adequada os valores que assumem as variáveis do processo durante a execução de uma rotina de controle.

O controle da inserção de dados nas tabelas, se necessário, deve ser feito dentro do programa *Assembler* de controle (aqui o projetista deve levar em consideração que o tamanho da memória é limitado para cada tabela, e dependendo da quantidade de dados gerada, também poderá ocorrer *Overflow*. Deve-se considerar aspectos como o período de amostragem, e calcular quanto tempo pode-se deixar o programa escrever dados na memória. Imagine que um contador esteja programado para gerar pulsos a cada 5ms, e deseja-se gravar em memória *RAM* os valores, para cada período, assumidos pela variável de posição; Como foi reservado para a *Tabela 1 de dados* a área entre 5ØØØh e 5FFFh (*4Kbytes*) → 4096 posições, se os valores forem gravados em posições sucessivas nesta tabela, poderia-se gravar um tempo total de 20,48s (5ms X 4Ø96).

Também deve-se sempre considerar que o programa de controle deve ser totalmente executado antes de um novo pedido de interrupção para um mesmo procedimento (mesma variável), senão, é fácil constatar que haverá problema. Porisso é importante que no desenvolvimento dos programas *Assembler* de controle para os Escravos, sejam realizadas simulações prévias com auxílio de simuladores que avaliam os tempos necessários para executar as rotinas, e com isso encontrar-se os períodos ideais para a programação.

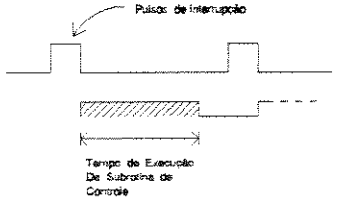


Figura C.2 - Intervalo Para a Execução das Subrotinas de Controle.

Uma vez que a subrotina de controle termine, o controle do Escravo retorna para o procedimento de interrupção para finalizar este ciclo, executa o retorno dos registradores, e então prossegue executando o programa principal. Este processo é "interminável", a menos que o Mestre interrompa, desabilitando o controlador por *software* (HINT = ØØh), ou estejam previstas outras possibilidades na programação.

O pedido para realização de uma conversão *A/D* deve ser feito dentro da própria subrotina de controle, e por isso deve-se ter muito cuidado com a estratégia de habilitação e desabilitação do procedimento de interrupção para não provocar confusão, e o sistema se perder.

C.2 ☞ PROGRAMAÇÃO PARA A TRANSFERÊNCIA DE DADOS ENTRE O MESTRE E OS ESCRAVOS

Para as transferências de dados entre o Mestre e os Escravos desenvolvemos um programa gerenciador de comunicações que tranfere arquivos *byte*-à-*byte* que estejam no formato INTEL INTELLEC 8/MDS conforme pode ser visto na **figura C.3**.

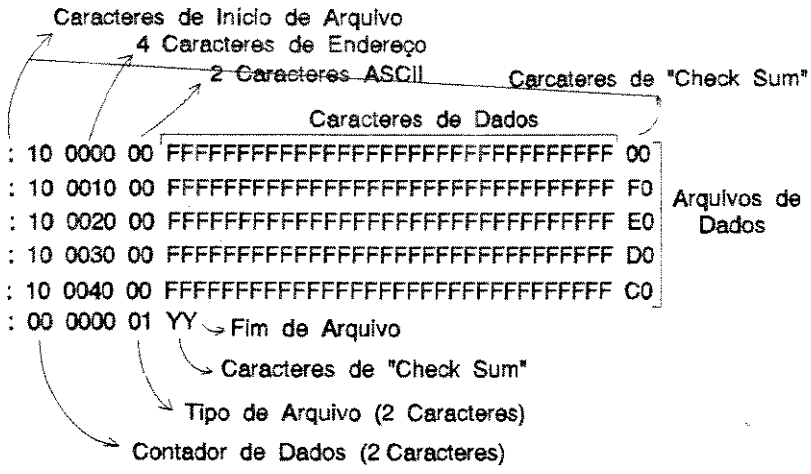


Figura C.3 - Formato INTEL INTELLEC 8/MDS.

Cada arquivo a ser transferido deve ser composto por linhas contendo uma seqüência de informações, com o caracter de início de arquivo ":" que define o início de uma linha do arquivo. Um contador de dados que define o número de caracteres de dados presentes na linha (dois Caracteres). Com dois caracteres de endereços que definem o endereço de memória no qual os dados devem ser alocados, em ordem seqüêncial na memória (quatro caracteres). Com o tipo de arquivo na linha (Ex. o valor "01" indica que a linha em questão é a última linha do arquivo. Com caracteres de dados que compõem o arquivo a ser transmitido propriamente dito (caracteres que devem ser carregados na memória especificada pelos caracteres de endereços). E com caracteres de *Check Sum* que têm a finalidade de verificar a integridade dos dados no arquivo. Este valor é a soma de todos os caracteres anteriores em complemento de um.

Este programa primeiramente faz a verificação da existência do arquivo a ser transmitido. Se o arquivo chamado não existe, o programa é abortado e é enviada uma mensagem de erro para a tela do monitor de vídeo (Erro 0). Depois ele verifica se o primeiro caracter do arquivo é um ":", em caso contrário ele também é abortado e é enviada outra mensagem para a tela do monitor de vídeo (erro 1). Se até aqui tudo estiver correto, então ele realiza uma leitura dos caracteres *ASCII* de dados da respectiva linha e convertê-los para caracteres hexadecimais correspondentes através de duas subrotinas que foram desenvolvidas com esta finalidade especial de converter dois *bytes* em *ASCII* para um *byte* hexadecimal, e quatro *bytes ASCII* para dois *bytes* hexadecimais.

A seguir é feita a geração dos sinais de memória no Mestre, com a formação de uma palavra digital composta por um registrador de segmento e um endereço de deslocamento. A informação contida no registrador de segmento aponta para um bloco de 64Kbytes de memória (16bits), para se ter acesso rápido por blocos de memória, e o endereço

de deslocamento (*offset* - 16bits) é quem dá a localização de um dado dentro do bloco determinado pelo registrador de segmento. O processador constrói o endereço de dados com 20bits, somando o endereço de deslocamento ao conteúdo do registrador de segmento com quatro zeros acrescentados nas posições menos significativas. Então através da programação o Mestre acessa as memórias dos Escravos como se estas fizessem parte da sua própria memória. O endereço utilizado pelo Mestre encontra-se entre 80000h e 8FFFFh, entretando, somente utilizamos 32Kbytes de memória. O endereco de segmento foi escolhido como 80000h. E o endereco de deslocamento (*offset*) é determinado pelo programa a ser transmitido, variando entre 0000h e 7FFFh.

Após a criação do conjunto de dados da linha, é feita a soma, em complemento de um, de todos os seus dados, e sua comparação com o *Check Sum*. Em caso de divergência é apresentado um erro na tela do monitor de vídeo (erro 2). O programa então passa a enviar os dados para o Escravo em questão (ou Escravos) gerando um pedido de requisição de barramento e endereçando a placa Escrava em questão. O envio de dados para estas placas é realizado através de instruções tipo *pokeb* e *peekb* da linguagem de programação C. A instrução *pokeb*(segmento,*offset*,dado) carrega o dado na posição determinada pelo segmento mais o *offset*, e a intrução *peekb*(segmento,*offset*), lê o dado da posição determinada pelo segmento e *offset*. O algoritmo estabelece a comunicação, envia um dado, realiza sua leitura, compara e apresenta mensagem de erro caso haja algum problema na gravação da memória detectado por diferença na comparação entre o dado escrito e o dado lido em uma mesma posição.

Após o envio da linha corrente do arquivo o programa passa à linha seguinte, realiza as mesmas operações comentadas anteriores, e termina quando encontrar um caracter de fim de arquivo.

Podem acontecer falhas devido a problemas de ruídos e de enfraquecimento de sinais nas linhas físicas de transmissão. Ainda continuamos pesquisando sobre o tratamento conveniente dos problemas de ruídos no ambiente, para obtermos respostas melhores, com aterramentos, blindagens, etc. Nós constatamos através de experimentos que ao desenvolvermos esta parte da programação para que seja feita somente uma tentativa de escrita e leitura de um mesmo dado de comunicação, existe chance de acontecer erro. Então permitimos que na tranferência de cada dado, este gerenciador faça tantas tentativas quantas forem necessárias para transferí-lo. A priori poderia acontecer travamento com a utilização de uma técnica assim, mas isso não ocorre. Verificamos que para um *byte* ser transferido com sucesso, raramente são necessárias mais do que três tentativas. Ou seja, para tranferir um *byte* referente a um nível de ajuste para um determinado Escravo, este tempo de tranferência é desprezível perante o tempo necessário para processar uma malha de controle. E quando for necessário tranferir um programa completo de controle do Mestre para os Escravos, portanto com probabilidade maior de acontecerem falhas nas tentativas, pode-se desativar os movimentos do robô, por estes instantes, até que tudo tenha sido feito com sucesso, e então retornar ao funcionamento normal.

Se acaso houver mesmo um problema grave, onde depois de várias tentativas ainda não houver sucesso na tranferência, então pode-se abortar o processo com envio da devida mensagem de erro. Isso pode ser feito pela inserção adequada de uma subrotina que trate este ítem, por exemplo, com utilização de uma interrupção cuja temporização é definida e inicializa em cada início de tranferência de um novo dado.

BIBLIOGRAFIA

[1] Ambrosino G., Celentano G., Garolato F.;"**Adaptive Tracking Control of Industrial Robots**", *Journal of Dynamic System Meas. and Control, Trans. of the ASME*, Vol.110, Sep (1988).

[2] Angeles J.;"**On The Numerical Solution of The Inverse Kinematic Problem**", The *Int. Journal of Robotics Research*, Vol.4, No.2, pp. 21-37, (1985).

[3] APEX; "**A trade Union Strategy For The New Technology**", *APEX - Association of Professional*, Executive, Clerical And Computer Staff, The Microeletronics Revolution, The MIT Press, (1981).

[4] Arimoto S., Kawamura S., Miyazaky F.;"**Bettering Operation of Dynamic Systems By Learning: A New Control Theory for Servomechanism or Mechatronics Systems**", *Proceedings of 23rd Conference on Decision and Control IEEE CDC, Las Vegas*, NV, USA, pp. 1064-1069, Dec (1984).

[5] Arimoto S., Kawamura S., Miyazaki F.; **Bettering Operation of Robots by Learning**, *Journal of Robotic Systems*, 1(2), pp. 123-140, John Wiley & Sons, Inc., (1984).

[6] Arimoto S., Kawamura S., Miyazaki F.;"**Can Mechanical Robots Learn By Themselves?**", *Proc. 2nd Int. Symp. Robotics Res.*, Kyoto, Japan, Chapter 3, pp. 127-134, Aug (1984).

[7] Badan A.G.P., Madrid M.K., Dias M.A., Ohfugi A.S.;"**Software Operational Structure to Robot Manipulator Control Via Parallel Processing**", *Congreso Latino-Americano de Control Automático*, Havana, Cuba, (1992).

[8] Badan A.G.P, Takita K.;"**Geração Autônoma de Trajetória Contínua Espacial Aplicada a Um Robô TRR**", *2º Congreso Latinoamericano de Control Automático -10º Simposio Nacional de Control Automático*, pp. 770-774, Buenos Aires, Argentina, Oct (1986).

[9] Badan A.G.P., Bottura C.P., Burian Jr. Y.;"**Eletrohydraulic System With State Feedback And Pulse Width Modulated Control**", *Joint Automatic Control Conference*, Philadelphia P.A., USA, Vol.1, Oct (1978).

[10] Bhat S.P., Miu D.K.;"**Solution to Point-to-Point Control Problems Using Laplace Transform Technique**", *Journal of Dynamics Systems, Measurement, and Control*, Vol.113, pp. 425-443, Sep (1991).

[11] Bien Z., Huh K.M.;"**Higher-Order Iterative Learning Control Algorithm**", *IEE Proceedings*, Vol.136, No.3, pp. 105-112, May (1989).

[12] Bobrow J.E., Dubowsky S., Gibson J.S.;"**Time Optimal Control of Robotic Manipulators Along Specified Paths**" *Int. J. Robot. Res.*, Vol.4, pp. 3-17, Aug (1985).

[13] Bottura C.P., Burian Y., Badan A.G.P, Rey J.P.;"**Simulation of a PWM Eletrohydraulic System**", *Simulation of Systems '79*, North-Holland Publishing Company, pp. 457-461, (1980).

[14] Boullion T., Odell P.L.; **Generalized Inverse Matrices**, *Wiley Interscience*, (1971).

[15] Chang Y.H., Lee T.T., Liu C.H.;"**On-Line Approximate Cartesian Path Trajectory Planning for Robotic Manipulators**", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.22, No.3, pp. 542-547, May/June (1992).

[16] Chiu S.L.;"**Control of Redundant Manipulators for Task Compatibility**", *Rockwell International Science Center*, Thousand Oaks, CA, USA, (1987).

[17] Craig J.; **Introduction to Robotics: Mechanics and Control**, *Addison-Wesley, Reading*, Mass, USA, (1986).

[18] Craig J., Hsu P., Sastry S.S.;"**Adaptive Control of Mechanical Manipulators**", *Proceedings 1986 IEEE International Conference on Robotics and Automation*, (1986).

[19] Dawson D.M., Qu Z., Carroll J.J.;"**Tracking Control of Rigid-Link Electrically-Driven Robot Manipulators**", *Int. J. Control*, Vol.56, No.5, pp. 991-1006, (1992).

[20] Denavit J., Hartenberg R.S.;"**A Kinematic Notation for Lower Pair Mechanisms Based on Matrices**", *ASME Journal of Applied Mechanics*, June (1955).

[21] Dias M.A., Madrid M.K., Ohfugi A.S., Badan A.G.P.;"**Controle de Robôs Manipuladores Utilizando Processamento Paralelo**", *IX Congreso Chileno de Ingenieria Elétrica*, Arica, Chile, Out (1991).

[22] Dias M.A.;"**Controlador Programável a Microprocessadores Para Controle Hierárquico de Robôs**", *Tese de Mestrado*, Unicamp, SP, (1991).

[23] Dubey R., Luh J.Y.S.;"**Redundant Robot Control For Higher Flexibility**", *Proc. 1987 IEEE Int. Conf. Robotics Automation*, pp. 1066-1072, (1987).

[24] Edan Y., Flash T., Peiper U.M., Shmulevich I., Sarig Y.; "**Near-Minimum-Time Task Planning for Fruit-Picking Robots**", *IEEE Trans. on Robotics and Automation*, Vol.7, No.1, pp. 48-56, Feb (1991).

[25] Eggebrecht L.C.; **Interfacing to The IBM Personal Computer**, *Howard W. Sams & Co.*, Indianapolis, USA, (1986).

[26] Elgazzar S.;"**Efficient Kinematic Transformations for PUMA 560 Robot**", *IEEE J. Robotics and Automat.*, Vol. RA-1, pp. 142-151, Sep (1985).

[27] Fu K.S., Gonzalez R.C., Lee C.S.G.; **Robotics, Control, Sensing, Intelligence**, *McGraw-Hill Book Company*, (1987).

[28] Gu Y.L., Loh N.K.;"**Learning Control in Robotic Systems**", *Proc. IEEE Int. Symp. on Intelligent Control*, pp. 360-364, Philadelphia, PA, USA, (1987).

[29] Hasegawa K., Mizutani T.;"**On the Autonomous Trajectory Generating Servomechanism for Manipulator Control**", *System Science VIII, International Conference on Systems Science*, Wroczaw, Poland, (1983).

[30] Isermann R.; **Digital Control Systems**, *Springer-Verlag*, (1981).

[31] Jayasuniya S., Hwang C.N.;"**Tracking Controllers for Robot Manipulators: A High Gain Perspective**", *Journal of Dyn. Syst. Meas. and Control, Trans. of the ASME*, Vol.110, March (1988).

[32] Jouaneh M.K., Wang Z., Dornfeld D.A.;"**Trajectory Planning for Coordinated Motion of a Robot and a Positioning Table: Part1-Path Specification**", *IEEE Trans. on Robotics and Automation*, Vol.6, No.6, pp. 735-745, Dec (1990).

[33] Jouaneh M.K., Dornfeld D.A., Tomizuka M.;"**Trajectory Planning for Coordinated Motion of a Robot and a Positioning Table: Part2-Optimal Trajectory Specification**", *IEEE Trans. on Robotics and Automation*, Vol.6, No.6, pp. 746-759, Dec (1990).

[34] Kawamura S., Miyazaki F., Arimoto S.;"**Iterative Learning Control for Robotic Systems**", *Proc. of IECON'84 Tokyo*, Japan, pp. 393-398, Oct (1984).

[35] Kawamura S., Miyazaki F., Arimoto S.;"**Applications of Learning Method for Dynamic Control of Robot Manipulators**", *Proceedings of 24th Conference on Decision and Control Ft.Lauderdale*, FL, USA, pp. 1381-1386, Dec (1985).

[36] Klein C.A., Blaho B.E.;"**Dexterity Measures For The Design and Control of Kinematically Redundant Manipulators**", *The International Journal of Robotics Research*, Vol.6, No.2, pp. 72-83, (1987).

[37] Koivo A.J.;"**Self-Tuning Manipulator Control in Cartesian Base Coordinate System**", *Journal of Dyn. Syst., Meas. and Control*, Trans. of *ASME*, Vol.107, Dec (1985).

[38] Korf R.E.;"**Real-Time Heuristic Search: New Results**", *Automated Reasoning*, pp. 139-144, (1988).

[39] Kuc T.Y., Nam K., Lee J.S.;"**An Iterative Control of Robot Manipulators**", *IEEE Trans. Robotics & Automation*, Vol.7, No.6, pp. 835-841, Dec (1991).

[40] Lewis F.L., Abdallah C.T., Dawson D.M.; **Control of Robot Manipulators**, *Macmillan Publishing Company*, N.Y., USA, (1993).

[41] Lim C.M., Hiyama T.;"**Application of Fuzzy Logic Control to a Manipulator**", *IEEE Trans. Robotics and Automation*, Vol.7, No.5, pp. 688-691, Oct (1991).

[42] Lim D.J., Chyung D.H.;"**Robust Optimal Tracking Controller For a Robotic Manipulator**", *Preprints X IFAC World Congress*, IFAC, Vol.4, (1987).

[43] Lin C.S., Chang P.R., Luh J.Y.S.;"**Formulation of Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots**", *IEEE Trans. Automat. Control*, Vol. AC-28, No.12, pp. 1066-1074, Dec (1983).

[44] Lin C.S., Chang P.R.;"**Joint Trajectories of Mechanical Manipulators for Cartesian Path Approximation**", *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-13, pp. 1094-1102, Nov/Dec (1983).

[45] Luh J.Y.S., Lin C.S.;"**Optimum Path Planning for Mechanical Manipulators**", *Journal Dyn. Syst., Meas., Control*, No.102, pp. 142-151, (1981).

[46] Madrid M.K., Monastério F.H.M.;"**Búsqueda Heurística en Tiempo Real Para La Generación de Trayectorias de Robots Manipuladores**", *V Conferencia de La Asociación Española Para La Inteligencia Artificial*, Madrid, España, (1993).

[47] Madrid M.K., Badan A.G.P.;"**Controle de Posição e Velocidade de Manipuladores Mecânicos Com Juntas em Malha Aberta e Malha Fechada**", *8º Congresso Brasileiro de Automática*, SBA, Belém, PA, (1990).

[48] Madrid M.K.;"**Robô-Manipulador Mecânico TRRR para Posicionamento Espacial com Controle Digital Hierárquico a Microprocessadores**", *Dissertação de Mestrado*, FEE/UNICAMP, Campinas SP, Agosto (1988).

[49] Madrid M.K., Badan A.G.P.;"**Um Sistema de Controle Digital para Movimentar Manipuladores Mecânicos**", *7º Congresso Brasileiro de Automática*, SBA, S.J Campos, SP, (1988).

[50] Mahla A.I., Badan A.G.P;"**Modelo Dinámico Del Sistema PLL-DUAL Para Control de Motores C.C. Región de Comportamiento Caótico**", *V Congreso Latino Americano de Control Automático*, Havana-Cuba, Fev (1992).

[51] Marin S.P.;"**Optimal Parametrization of Curves for Robot Trajectory Design**", *IEEE Trans. on Automatic Control*, Vol.33, No.2, pp. 209-214, Feb (1988).

[52] Miller W.T., Glanz F.H., Kraft L.G.;"**Application of a General Learning Algorithm to the Control of Robotic Manipulators**", *The International Journal of Robotics Research*, Vol.6, No.2, pp. 84-98, (1987).

[53] Mizutani T., Hasegawa K.;"**A Survey On Industrial Automation And Robotics**", *3º Congresso de Automação Industrial*", Centro de Convenções Anhembi, SP, Set (1988).

[54] Monastério F.H.M., Madrid M.K.;"**Análisis de Un Método de Búsqueda Heurística en Tiempo Real Para La Generación de Trayectorias de Robots**", *Memoria Interna, Grupo Ingeniería de Control*, ETSI Teleco, Universidad Politécnica de Madrid, Madrid (1993).

[55] Ohfugi A.S.;"**Garras Articuladas Para Robôs Manipuladores: Análise Cinemática e de Forças, Sensoreamento e Controle de Posição e Esforços na Preensão de Objetos**", *Tese de Mestrado*, Unicamp, SP, (1991).

[56] Oliveira A.M., Badan A.G.P, Kamakura A.H., Winnischofer G., Hoshino A.;"**Analysis of Brushless D.C. Motor Performance When Faults Occurrence**", *EPE 91 - European Conference On Power Electronic And Application*, Torino, Italy, Set (1991).

[57] Osório A.F.S, Cunha J.S., Souza M.A.; "**Comentário Sobre o Impacto Social da Automação**", *Comunicação Interna*, Trabalho de Curso, EE-901/Robótica, FEE, Unicamp, (1993).

[58] Qu Z., Dorsey J.;"**Robust Tracking Control of Robots by a Linear Feedback Law**", *IEEE Trans. on Automatic Control*, Vol.36, No.9, pp. 1081-1084, Sep (1991).

[59] Pak H.A., Turner P.J.;"**Optimal Tracking Controller Desing for Invariant Dynamics Direct-Drive Arms**", *Journal of Dyn. Syst. Meas., and Control*, Trans. of the *ASME*, Vol.108, Dec (1986).

[60] Paul R.P.;"**Manipulator Cartesian Path Control**", *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-9, pp. 702-711, Nov (1979).

[61] Paul R.P.; **Robot Manipulators: Mathematics, Programming, and Control**, *MIT Press*, Cambridge, MA, (1981).

[62] Pearl J.; **Heuristics - Intelligent Search Strategies For Computer Problem Solving**, *Addison-Wesley Publishing Company*, Inc, Mass, (1985).

[63] Peliano J.C.;"**Automação, Emprego e Qualificação da Mão-de-Obra na Indústria Brasileira**", *Brasiliense*, São Paulo, (1983).

[64] Rosário J.M., Saramago M.A.P., Messina L.C.P., Niemann H.R., Aust E.;"**Intervention of Advanced Subsea Robots at a 1000 msw Template Manifold**", *3rd International Offshore and Polar Engineering Conference*, Singapore, june (1993).

[65] Rosário J.M.;"**Feasibility Study of Non-Linear Controller for Robotic Manipulators**", *IV DINAME - Symposium on Dynamic Problems of Mechanics*, pp. 105-106, Pouso Alto, Minas Gerais, Mar (1991).

[66] Sadler O.;"**Welcome Back to The Automation Debate**", *The Microeletronics Revolution*, The MIT Press, Cambridge, USA, (1981).

[67] Sahar G., Hollerbach J.M.;"**Planning of Minimum-Time Trajectories for Robot Arms**", *The International Journal of Robotics Research*, Vol.5, No.3, pp. 90-100, (1986).

[68] Sasaki S.;"**New Approaches To Manipulator Arm Solution Via Unconstrained Optimization Theory**", *Robótica*, Vol.11, pp. 253-262, (1993).

[69] Seraji H.;"**A New Approach to Adaptive Control of Manipulators**", *Journal of Dyn. Syst. Meas. and Control*, Trans. of the *ASME*, Vol.109, Sep (1987).

[70] Shin K.G., Mckay N.D.;"**Minimum Time Control of Robotic Manipulator With Geometric Path Constraints**", *IEEE Trans. Automation. Control*, Vol.30, No.6, pp. 532-541, June (1985).

[71] Slotine J.J.E.;"**The Robust Control of Robot Manipulators**", *The International Journal of Robotic Research*, Vol.4, No.2, (1985).

[72] Spong M.W.;"**Modeling And Control of Elastic Joint Robots**", *Trans. ASME, J. Dynamics Systems, Measurement and Control*, Vol.109, pp. 310-319, (1987).

[73] Stone H.W.; **Kinematic Modeling, Identification and Control of Robotic Manipulators**, *Kluwer Academic Publishers*, (1987).

[74] Stonier T.; **Strategies For Change in a Technological Society**, Bradford, England, Batr, (1979).

[75] Takita K., Vieira A., Badan A.G.P.;"**Desenvolvimento de Uma Garra com Sensores Para Manuseio de Peças Frágeis**", *9º Congresso Brasileiro de Automática*, vitória, ES, Set (1992).

[76] Tauile J.R.; "**Robótica: Reflexões Sobre Um Novo Limiar**", *RBT - Revista Brasileira de Tecnologia*, 16(5), Set/Out (1985).

[77] Texas Instruments; **The Power Semiconductor Data Book For Design Engineers**, *Adlard & Son Ltd.*, 2nd Edition, Dorking, Surrey, USA, (1983).

[78] Thomopoulos S.C.A., Tam R.Y.J.;"**An Iterative Solution To The Inverse Kinematic of Robotic Manipulators**", *Mech. Mach. Theory*, Vol.26, No.4, pp. 359-373, (1991).

[79] Vidyasagar M.;"**System Theory And Robotics**", *IEE Control Systems Magazine*, pp. 16-19, April (1987).

[80] Vieira A., Takita K., Badan A.G.P;"**On The Autonomous Tridimensional Trajectory Generating Technique For Articulated Robot Arm**", *3rd International Workshop On Advanced Motion Control*, Berkeley, USA, Mar (1994).

[81] Vieira A.L, Takita K., Mizutani T., Badan A.G.P.;"**Application And Control of The Differential Gear Articulated Robot Arm**", *AMST 90 - 4th International Symposium - Application of Multivariable System Techniques*, Bradford, West Yorkshire, U.K., April (1990).

[82] Vieira A.L., Takita K., Badan A.G.P.;"**O Efeito do Controle de Velocidade Das Juntas no Desempenho da Trajetória de Um Braço Mecânico**", *6º Congresso Brasileiro de Automática*, Belo Horizonte, (1986).

[83] Vukobratovic M., Stonik D.; **Synthesis And Control Algorithms of Manipulation Robots**, *Springer-Verlag*, Berlin, Heidelberg, New York, (1982).

[84] Wang L.Ch.T., Chen Ch.Ch.;"**A Combined Optimization Method For Solving The Inverse Kinematics Problem of Mechanical Manipulators**", *IEEE Trans. on Robotics and Automation*, Vol.7, No.4, pp. 489-499, Aug (1991).

[85] Whitney O.E.;"**The Robust Control of Robot Manipulators**", *The International Journal of Robotic Research*, Vol.4, No.2, (1985).

[86] Wit C.C., Fixot N., Astrom K.J.;"**Tracking in Robot Manipulators via Nonlinear Estimated State Feedback**", *IEEE Trans. Robotics and Automation*, Vol.8, No.1, pp. 138-144, feb (1992).

[87] Yoshikawa T.;"**Manipulability of Robotic Mechanisms**", *The International Journal of Robotics Research*, Vol.4, No.2, pp. 3-9, (1985).

[88] Yoshikawa T.;"**Dynamic Manipulability of Robot Manipulators**", *Journal of Robotic Systems*, John Wiley & Sons, Inc., 2(1), pp. 113-124, (1985).